typst

# Documentación de Typst

Versión: 0.15.1

Derechos de autor © 2019 – 2026 Laurenz Mädje, Martin Haug y The Typst Project Developers.

El contenido de este documento y el compilador de Typst se licencian bajo los términos de la Apache License, Version 2.0.

Esta documentación es de código abierto y se puede contribuir a ella en https://github.com/typst/typst. Fue compuesta con Typst 0.15.1, Reforma 2018, Reforma 1918, Google Sans Code e IBM Plex Math.

La documentación original en inglés es publicada por Typst GmbH, Heidestraße 34, 10557 Berlin, Germany.

https://typst.app/

**Traducción no oficial.** Esta es una traducción al español rioplatense (es-AR) de la documentación de Typst. **No es una publicación oficial** de Typst GmbH ni del equipo de Typst, y no cuenta con su aval. Fue traducida con inteligencia artificial (Claude Sonnet 5.5, de Anthropic), bajo la dirección y revisión de Jassiel Ovando, y puede contener errores o imprecisiones. Ante cualquier duda, prevalece la documentación oficial en inglés:
https://typst.app/docs

**Cambios de diseño respecto del original.** Esta edición usa otras tipografías: la familia Reforma (de PampaType, creada para la Universidad Nacional de Córdoba): Reforma 2018, sin serifa, en lugar de HK Grotesk para el texto, y Reforma 1918, con serifa, para los títulos, que en el original usan la misma tipografía que el texto; Google Sans Code en lugar de Cascadia Mono para el código y IBM Plex Math en lugar de New Computer Modern Math para las fórmulas. Los ejemplos de código compilados se componen con el idioma español de Argentina (es-AR), y el código de los ejemplos, que no se tradujo, se muestra tal como en el original.

# Índice


**1 Descripción general** 9

**2 Tutorial** 10

Introduction ... 10
2.0.1 Cuándo Typst ... 10
2.0.2 Aprendizajes ... 10
2.1 Escribir en Typst ... 11
2.1.1 Figura ... 12
2.1.2 Bibliografía ... 15
2.1.3 Matemática ... 15
2.1.4 Repaso ... 17
2.2 Formato ... 18
2.2.1 Reglas set ... 18
2.2.2 Autocompletado ... 19
2.2.3 Configurar página ... 19
2.2.4 Sofisticación ... 21
2.2.5 Reglas show ... 22
2.2.6 Repaso ... 22
2.3 Estilos avanzados ... 23
2.3.1 Pautas ... 23
2.3.2 Reglas set ... 24
2.3.3 Título y resumen ... 24
2.3.4 Columnas y títulos ... 28
2.3.5 Review ... 31
2.4 Crear una plantilla ... 32
2.4.1 Variables ... 32
2.4.2 Plantilla de juguete ... 33
2.4.3 Reglas set y show ... 34
2.4.4 Argumentos con nombre ... 35
2.4.5 Archivo separado ... 37
2.4.6 Review ... 38

**3 Referencia** 39

Introduction ... 39
3.0.1 Lenguaje ... 39
3.0.2 Biblioteca ... 39
3.0.3 Cómo leer la referencia ... 39
3.1 Sintaxis ... 40
3.1.1 Modos ... 40
3.1.2 Marcado ... 40
3.1.3 Matemática ... 40
3.1.4 Código ... 41
3.1.5 Comentarios ... 42
3.1.6 Escapes ... 42
3.1.7 Identifiers ... 43
3.2 Estilos ... 44
3.2.1 Reglas set ... 44
3.2.2 Reglas show ... 44
3.3 Scripting ... 47
3.3.1 Expresiones ... 47
3.3.2 Bloques ... 47
3.3.3 Enlaces ... 48
3.3.4 Condicionales ... 49

3.3.5 Bucles ... 49
3.3.6 Campos ... 50
3.3.7 Métodos ... 50
3.3.8 Módulos ... 51
3.3.9 Paquetes ... 51
3.3.10 Operadores ... 52
3.4 Contexto ... 53
3.4.1 Contexto de estilo ... 53
3.4.2 Contexto de ubicación ... 53
3.4.3 Contextos anidados ... 55
3.4.4 Iteraciones del compilador ... 56
3.5 Fundamentos ... 57
3.5.1 Arguments ... 58
3.5.2 Array ... 61
3.5.3 Assert ... 71
3.5.4 Auto ... 73
3.5.5 Biblioteca estándar ... 74
3.5.6 Boolean ... 75
3.5.7 Bytes ... 76
3.5.8 Content ... 78
3.5.9 Cálculo ... 80
3.5.10 Datetime ... 96
3.5.11 Decimal ... 101
3.5.12 Dictionary ... 103
3.5.13 Duration ... 106
3.5.14 Evaluate ... 108
3.5.15 Float ... 109
3.5.16 Function ... 113
3.5.17 Integer ... 116
3.5.18 Label ... 122
3.5.19 Module ... 123
3.5.20 None ... 124
3.5.21 Panic ... 125
3.5.22 Path ... 126
3.5.23 Plugin ... 129
3.5.24 Regex ... 132
3.5.25 Representation ... 134
3.5.26 Selector ... 135
3.5.27 Sistema ... 138
3.5.28 String ... 139
3.5.29 Symbol ... 147
3.5.30 Target ... 148
3.5.31 Type ... 149
3.5.32 Version ... 151
3.6 Modelo ... 152
3.6.1 Bibliography ... 153
3.6.2 Bullet List ... 160
3.6.3 Cite ... 163
3.6.4 Divider ... 168
3.6.5 Document ... 169
3.6.6 Emphasis ... 173
3.6.7 Figure ... 174
3.6.8 Footnote ... 181
3.6.9 Heading ... 184
3.6.10 Link ... 189
3.6.11 Numbered List ... 192

3.6.12 Numbering ... 196
3.6.13 Outline ... 198
3.6.14 Paragraph ... 204
3.6.15 Paragraph Break ... 213
3.6.16 Quote ... 214
3.6.17 Reference ... 218
3.6.18 Strong Emphasis ... 221
3.6.19 Table ... 222
3.6.20 Term List ... 232
3.6.21 Title ... 234
3.7 Texto ... 235
3.7.1 Highlight ... 236
3.7.2 Line Break ... 238
3.7.3 Lorem ... 239
3.7.4 Lowercase ... 240
3.7.5 Overline ... 241
3.7.6 Raw Text / Code ... 243
3.7.7 Small Capitals ... 255
3.7.8 Smartquote ... 257
3.7.9 Strikethrough ... 259
3.7.10 Subscript ... 261
3.7.11 Superscript ... 262
3.7.12 Text ... 264
3.7.13 Underline ... 278
3.7.14 Uppercase ... 280
3.8 Matemática ... 281
3.8.1 Variables ... 281
3.8.2 Símbolos ... 281
3.8.3 Saltos de línea ... 281
3.8.4 Llamadas a funciones ... 281
3.8.5 Alineación ... 282
3.8.6 Fuentes matemáticas ... 282
3.8.7 Módulo math ... 283
3.8.8 Accesibilidad ... 283
3.8.9 Accent ... 284
3.8.10 Adjuntos ... 286
3.8.11 Binomial ... 288
3.8.12 Cancel ... 289
3.8.13 Cases ... 291
3.8.14 Class ... 292
3.8.15 Debajo/Encima ... 293
3.8.16 Equation ... 297
3.8.17 Espacios ... 300
3.8.18 Estilos ... 301
3.8.19 Fraction ... 302
3.8.20 Izquierda/Derecha ... 304
3.8.21 Matrix ... 308
3.8.22 Primes ... 311
3.8.23 Raíces ... 312
3.8.24 Stretch ... 313
3.8.25 Tamaños ... 315
3.8.26 Text Operator ... 317
3.8.27 Variantes ... 318
3.8.28 Vector ... 321
3.9 Símbolos ... 323
3.9.1 Atajos ... 323

3.9.2 Símbolos generales ... 324
3.9.3 Emoji ... 336
3.10 Diseño ... 349
3.10.1 Align ... 350
3.10.2 Alignment ... 352
3.10.3 Angle ... 355
3.10.4 Block ... 356
3.10.5 Box ... 360
3.10.6 Column Break ... 363
3.10.7 Columns ... 364
3.10.8 Direction ... 367
3.10.9 Fraction ... 371
3.10.10 Grid ... 372
3.10.11 Hide ... 385
3.10.12 Layout ... 386
3.10.13 Length ... 388
3.10.14 Measure ... 390
3.10.15 Move ... 392
3.10.16 Padding ... 393
3.10.17 Page ... 395
3.10.18 Page Break ... 406
3.10.19 Place ... 407
3.10.20 Ratio ... 411
3.10.21 Relative Length ... 412
3.10.22 Repeat ... 414
3.10.23 Rotate ... 415
3.10.24 Scale ... 417
3.10.25 Skew ... 419
3.10.26 Spacing (H) ... 421
3.10.27 Spacing (V) ... 423
3.10.28 Stack ... 424
3.11 Visualización ... 425
3.11.1 Accesibilidad ... 425
3.11.2 Circle ... 426
3.11.3 Color ... 428
3.11.4 Curve ... 442
3.11.5 Ellipse ... 449
3.11.6 Gradient ... 450
3.11.7 Image ... 459
3.11.8 Line ... 464
3.11.9 Polygon ... 466
3.11.10 Rectangle ... 468
3.11.11 Square ... 471
3.11.12 Stroke ... 473
3.11.13 Tiling ... 478
3.12 Introspección ... 483
3.12.1 Counter ... 484
3.12.2 Here ... 489
3.12.3 Locate ... 490
3.12.4 Location ... 491
3.12.5 Metadata ... 493
3.12.6 Query ... 494
3.12.7 State ... 498
3.13 Carga de datos ... 503
3.13.1 Codificación ... 503
3.13.2 CBOR ... 504

3.13.3 CSV ... 505
3.13.4 JSON ... 506
3.13.5 Read ... 508
3.13.6 TOML ... 509
3.13.7 XML ... 511
3.13.8 YAML ... 513
3.14 Formatos ... 515
3.14.1 Ajustes de exportación ... 515
3.14.2 Bundle ... 516
3.14.3 HTML ... 519
3.14.4 PDF ... 572
3.14.5 PNG ... 587
3.14.6 SVG ... 589

**4 Guías** **590**

4.1 Guía para usuarios de LaTeX ... 591
4.1.1 Instalación ... 591
4.1.2 Cómo empezar ... 591
4.1.3 Elementos ... 592
4.1.4 Comandos ... 593
4.1.5 Plantillas ... 595
4.1.6 Paquetes ... 597
4.1.7 Matemática ... 597
4.1.8 Aspecto LaTeX ... 599
4.1.9 Bibliografías ... 600
4.1.10 Limitaciones ... 600
4.2 Guía de configuración de página ... 601
4.2.1 Personalizar márgenes ... 601
4.2.2 Encabezados y pies ... 604
4.2.3 Números de página ... 605
4.2.4 Columnas ... 609
4.2.5 Modificaciones puntuales ... 611
4.3 Guía de tablas ... 612
4.3.1 Tablas básicas ... 612
4.3.2 Tamaños de columna ... 613
4.3.3 Epígrafes y referencias ... 614
4.3.4 Rellenos ... 615
4.3.5 Trazos ... 619
4.3.6 Alineación ... 625
4.3.7 Combinar celdas ... 627
4.3.8 Rotar una tabla ... 628
4.3.9 Tabla en varias páginas ... 632
4.3.10 Importar datos ... 633
4.3.11 Tabla y grilla ... 638
4.4 Guía de accesibilidad ... 639
4.4.1 Conceptos básicos ... 639
4.4.2 Mantener la semántica ... 639
4.4.3 Orden de lectura ... 640
4.4.4 Contenedores de diseño ... 641
4.4.5 Artefactos ... 641
4.4.6 Uso del color y contraste ... 642
4.4.7 Representaciones textuales ... 644
4.4.8 Idioma natural ... 646
4.4.9 Título del documento y títulos de sección ... 647
4.4.10 Estándares y legislación de accesibilidad ... 647
4.4.11 Pruebas de accesibilidad ... 649

4.4.12 Límites y consideraciones sobre los formatos de exportación ..... 650

**5 Changelog** **652**

0.15.1 ..... 653
0.15.0 ..... 654
0.14.2 ..... 666
0.14.1 ..... 667
0.14.0 ..... 669
0.13.1 ..... 677
0.13.0 ..... 678
0.12.0 ..... 684
0.11.1 ..... 691
0.11.0 ..... 692
0.10.0 ..... 697
0.9.0 ..... 699
0.8.0 ..... 702
0.7.0 ..... 704
0.6.0 ..... 707
0.5.0 ..... 709
0.4.0 ..... 711
0.3.0 ..... 713
0.2.0 ..... 715
0.1.0 ..... 716
Earlier ..... 718

# 1 Descripción general

¡Te damos la bienvenida a la documentación de Typst! Typst es un sistema de composición tipográfica basado en marcado que combina una automatización potente y una tipografía de alta calidad con velocidad y facilidad de uso. Esto lo hace apto para documentos de cualquier complejidad. Typst es una gran alternativa tanto a los procesadores de texto como a LaTeX.

Esta documentación está dividida en varias partes, que cubren distintas necesidades:

- Si sos nuevo en Typst, te recomendamos empezar con nuestro tutorial para principiantes[2]. A lo largo del tutorial, te vamos a presentar Typst a través de un ejemplo práctico.
- Para responder preguntas puntuales sobre Typst y familiarizarte con las funciones avanzadas, usá la referencia[3]. Describe las características fundamentales del lenguaje Typst y tiene secciones para todas las funciones, los tipos y más elementos que vienen con Typst.
- Para instructivos a medida y en profundidad sobre funciones, casos de uso y públicos específicos, fijate en nuestras guías[4]. Incluyen fragmentos de código para copiar y te permiten ganar confianza en un área concreta. Si venís de LaTeX, la Guía para usuarios de LaTeX[4.1] ofrece una introducción alternativa a Typst, que se apoya en conceptos que ya conocés.

El término *Typst* hace referencia a tres conceptos: el lenguaje Typst, el compilador de Typst y la app web de Typst. El lenguaje es lo que escribís, el compilador traduce los archivos en lenguaje Typst a PDF, páginas HTML y otros formatos, y la app web de Typst te permite trabajar de forma colaborativa en proyectos de Typst desde el navegador. El lenguaje Typst y el compilador son de código abierto.

Esta documentación documenta principalmente el lenguaje Typst, aunque el tutorial y varias páginas hacen referencia a la app web y al compilador de Typst por línea de comandos. En la copia de la documentación alojada en https://typst.app/docs/[A], también incluimos documentación sobre la app web. Para aprender a instalar el compilador de Typst por línea de comandos, visitá la página de código abierto[B] de nuestro sitio web. Ahí también podés conocer más sobre la relación entre el compilador de Typst y la app web. Una vez que hayas instalado el compilador, ejecutá `typst help` para obtener más información sobre cómo usarlo.

Nuestro repositorio de GitHub[C] ofrece documentación adicional para desarrolladores sobre cómo contribuir a Typst y cómo integrarlo en tus aplicaciones.

La documentación también contiene un registro de cambios[5], en el que podés seguir la evolución de Typst y ver qué implican los cambios del lenguaje de marcado para tus proyectos. Esta documentación corresponde a Typst 0.15.1.

[A] https://typst.app/docs/
[B] https://typst.app/open-source
[C] https://github.com/typst/typst

# 2 Tutorial

¡Te damos la bienvenida al tutorial de Typst! En este tutorial vas a aprender a escribir y dar formato a documentos en Typst. Vamos a empezar con tareas cotidianas e ir introduciendo de a poco funciones más avanzadas. Este tutorial no supone conocimientos previos de Typst, de otros lenguajes de marcado ni de programación. Sí damos por sentado que sabés editar un archivo de texto.

La mejor forma de empezar es registrarte gratis en la app de Typst y seguir los pasos de abajo. La app te da vista previa instantánea, resaltado de sintaxis y autocompletado útil. Como alternativa, podés seguir el tutorial en tu editor de texto local con la CLI de código abierto[A].

### 2.0.1 Cuándo usar Typst

Antes de empezar, veamos qué es Typst y cuándo conviene usarlo. Typst es un lenguaje de marcado para componer documentos. Está diseñado para ser fácil de aprender, rápido y versátil. Typst toma archivos de texto con marcado y genera PDF.

Typst es una buena opción para escribir cualquier texto extenso, como ensayos, artículos, trabajos científicos, libros, informes y tareas. Además, Typst es ideal para documentos con notación matemática, como los trabajos de matemática, física e ingeniería. Por último, gracias a sus potentes funciones de estilos y automatización, es una excelente opción para cualquier conjunto de documentos que compartan un estilo común, como una colección de libros.

### 2.0.2 Qué vas a aprender

Este tutorial tiene cuatro capítulos. Cada capítulo se apoya en el anterior. Esto es lo que vas a aprender en cada uno:

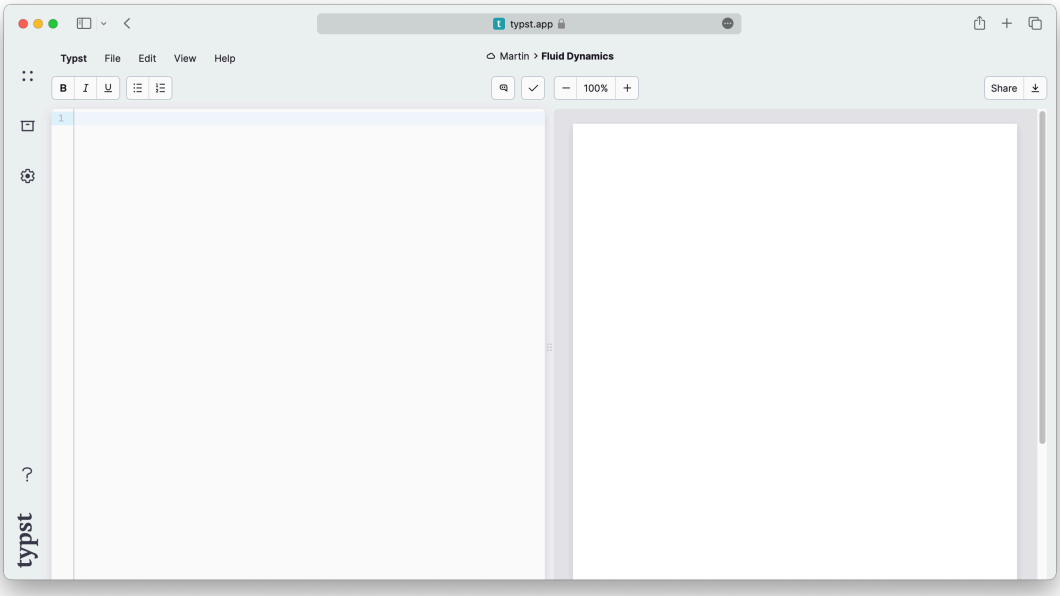
1. Escribir en Typst:[2.1] Aprendé a escribir texto e insertar imágenes, ecuaciones y otros elementos.
2. Formato:[2.2] Aprendé a ajustar el formato de tu documento, incluidos el tamaño de fuente, los estilos de los títulos y más.
3. Estilos avanzados:[2.3] Creá un diseño de página complejo para un trabajo científico, con características tipográficas como una lista de autores y títulos incorporados al párrafo.
4. Crear una plantilla:[2.4] Construí una plantilla reutilizable a partir del trabajo que creaste en el capítulo anterior.

¡Esperamos que disfrutes Typst!

[A] https://github.com/typst/typst

## 2.1 Escribir en Typst

¡Empecemos! Supongamos que te asignaron escribir un informe técnico para la universidad. Va a tener prosa, matemática, títulos y figuras. Para arrancar, creás un proyecto nuevo en la app de Typst. Vas a llegar al editor, donde ves dos paneles: un panel de código, donde componés tu documento, y un panel de vista previa, donde ves el documento renderizado.




Ya tenés un buen enfoque para tu informe. Así que empecemos escribiendo la introducción. Ingresá algo de texto en el panel del editor. Vas a notar que el texto aparece de inmediato en la página de la vista previa.

```
En este informe, exploraremos los
diversos factores que influyen en la
dinámica de fluidos en los glaciares
y cómo contribuyen a la formación y
al comportamiento de estas
estructuras naturales.
```

En este informe, exploraremos los diversos factores que influyen en la dinámica de fluidos en los glaciares y cómo contribuyen a la formación y al comportamiento de estas estructuras naturales.

*A lo largo de este tutorial, vamos a mostrar ejemplos de código como este. Igual que en la app, el primer panel contiene el marcado y el segundo muestra una vista previa. Achicamos la página para que entren los ejemplos y puedas ver lo que pasa.*

El siguiente paso es agregar un título y resaltar algo de texto. Typst usa un marcado simple para las tareas de formato más comunes. Para agregar un título, escribí el carácter =, y para resaltar texto en cursiva, encerralo entre `_underscores_`.

```
= Introducción
En este informe, exploraremos los
diversos factores que influyen en la
_dinámica de fluidos_ en los glaciares
y cómo contribuyen a la formación y
al comportamiento de estas
estructuras naturales.
```

### Introducción

En este informe, exploraremos los diversos factores que influyen en la *dinámica de fluidos* en los glaciares y cómo contribuyen a la formación y al comportamiento de estas estructuras naturales.

¡Qué fácil! Para agregar un párrafo nuevo, simplemente dejá una línea en blanco entre dos líneas de texto. Si ese párrafo necesita un subtítulo, lo generás escribiendo == en lugar de =. La cantidad de caracteres = determina el nivel de anidación del título.

Ahora queremos enumerar algunas de las circunstancias que influyen en la dinámica de los glaciares. Para eso usamos una lista numerada. Para cada ítem de la lista, escribimos un carácter + al principio de la línea. Typst numera los ítems automáticamente.

```
+ El clima
+ La topografía
+ La geología
```

1. El clima
2. La topografía
3. La geología

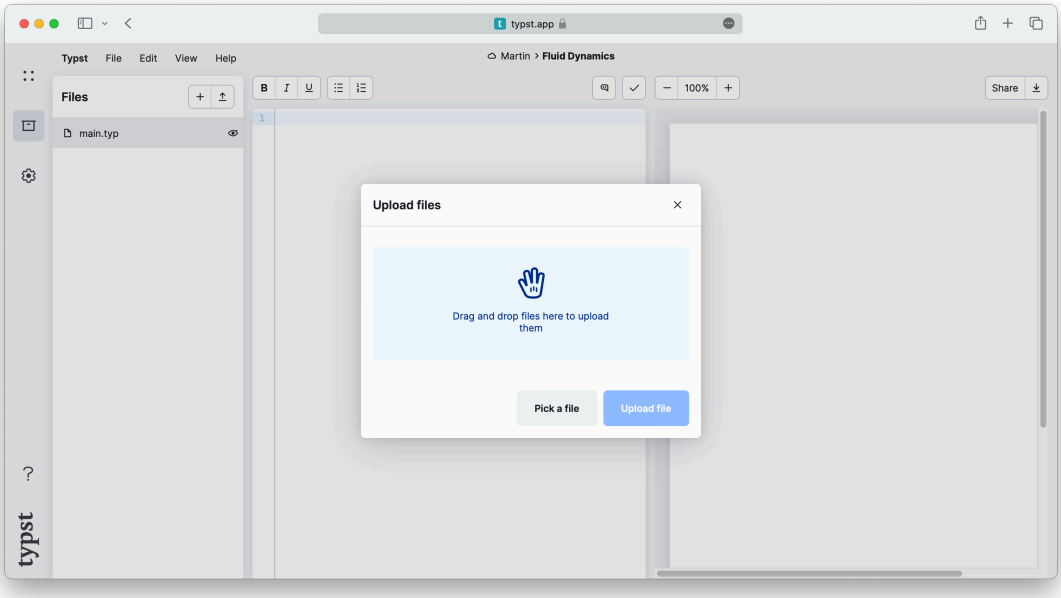
Si quisiéramos agregar una lista con viñetas, usaríamos el carácter – en lugar del carácter +. También podemos anidar listas: por ejemplo, podemos agregar una sublista al primer ítem de la lista anterior indentándola.

```
+ El clima
  - Temperatura
  - Precipitación
+ La topografía
+ La geología
```

1. El clima
   - Temperatura
   - Precipitación
2. La topografía
3. La geología

### 2.1.1 Agregar una figura

Pensás que tu informe se beneficiaría con una figura. Agreguemos una. Typst admite imágenes en los formatos PNG, JPEG, GIF, SVG, PDF y WebP. Para agregar un archivo de imagen a tu proyecto, primero abrí el *panel de archivos* haciendo clic en el ícono de la caja en la barra lateral izquierda. Ahí ves la lista de todos los archivos de tu proyecto. Por ahora hay uno solo: el archivo principal de Typst en el que estás escribiendo. Para subir otro archivo, hacé clic en el botón con la flecha en la esquina superior derecha. Se abre el diálogo de carga, en el que podés elegir los archivos de tu computadora que querés subir. Seleccioná un archivo de imagen para tu informe.



Ya vimos que ciertos símbolos (llamados *marcado*) tienen un significado específico en Typst. Podemos usar =, –, + y _ para crear títulos, listas y texto resaltado, respectivamente. Sin embargo, tener un símbolo especial para todo lo que queremos insertar en el documento se volvería enseguida críptico y engorroso. Por eso, Typst reserva los símbolos de marcado solo para las cosas más comunes. Todo lo demás se inserta con *funciones*. Para que nuestra imagen aparezca en la página, usamos la función `image`[3.11.7] de Typst.

```
#image("glacier.jpg")
```



En general, una función produce algún resultado a partir de un conjunto de *argumentos*. Cuando *llamás* a una función dentro del marcado, le pasás los argumentos y Typst inserta el resultado (el *valor de retorno* de la función) en el documento. En nuestro caso, la función `image` toma un solo argumento: la ruta del archivo de imagen. Para llamar a una función en el marcado, primero escribimos el carácter #, seguido inmediatamente del nombre de la función. Después, encerramos los argumentos entre paréntesis. Typst reconoce muchos tipos de datos distintos en las listas de argumentos. Nuestra ruta de archivo es una cadena de texto[3.5.28] corta, así que tenemos que encerrarla entre comillas dobles.

La imagen insertada ocupa todo el ancho de la página. Para cambiar eso, pasale el argumento `width` a la función `image`. Es un argumento *con nombre* y, por lo tanto, se especifica como un par `name: value`. Si hay varios argumentos, se separan con comas, así que primero tenemos que poner una coma después de la ruta.

```
#image("glacier.jpg", width: 70%)
```



El argumento `width` es una longitud relativa[3.10.21]. En nuestro caso, especificamos un porcentaje, que determina que la imagen ocupe el `70%` del ancho de la página. También podríamos haber especificado un valor absoluto como `1cm` o `0.7in`.

Igual que el texto, la imagen queda alineada a la izquierda de la página por defecto. Además, le falta un epígrafe. Arreglemos eso con la función figure[3.6.7]. Esta función toma el contenido de la figura como argumento posicional y un epígrafe opcional como argumento con nombre.

Dentro de la lista de argumentos de la función `figure`, Typst ya está en modo código. Esto significa que ahora tenés que sacar el numeral antes de la llamada a la función image. El numeral solo hace falta directamente en el marcado (para distinguir el texto de las llamadas a funciones).

El epígrafe consiste en marcado arbitrario. Para pasarle marcado a una función, lo encerramos entre corchetes. Esta construcción se llama *bloque de contenido*.

```
#figure(
  image("glacier.jpg", width: 70%),
```

```
  caption: [
    Los _glaciares_ son una parte importante
    del sistema climático de la Tierra.
  ],
)
```



Figura 1: Los *glaciares* son una parte importante del sistema climático de la Tierra.

Seguís escribiendo tu informe y ahora querés hacer referencia a la figura. Para eso, primero adjuntale una etiqueta a la figura. Una etiqueta identifica de forma única un elemento de tu documento. Agregá una después de la figura encerrando algún nombre entre paréntesis angulares. Después podés referenciar la figura en tu texto escribiendo un símbolo @ seguido de ese nombre. Los títulos y las ecuaciones también se pueden etiquetar para que sean referenciables.

```
¡Glaciares como el que se muestra en
@glaciers dejarán de existir si
no actuamos pronto!

#figure(
  image("glacier.jpg", width: 70%),
  caption: [
    Los _glaciares_ son una parte importante
    del sistema climático de la Tierra.
  ],
) <glaciers>
```

¡Glaciares como el que se muestra en Figura 1 dejarán de existir si no actuamos pronto!



Figura 1: Los *glaciares* son una parte importante del sistema climático de la Tierra.

INFO

Hasta ahora, les pasamos a nuestras funciones bloques de contenido (marcado entre corchetes) y cadenas de texto (texto entre comillas dobles). Los dos parecen contener texto. ¿Cuál es la diferencia?

Un bloque de contenido puede contener texto, pero también cualquier otro tipo de marcado, llamadas a funciones y más, mientras que una cadena es simplemente una *secuencia de caracteres* y nada más.

Por ejemplo, la función image espera una ruta a un archivo de imagen. No tendría sentido pasarle, por ejemplo, un párrafo de texto u otra imagen como ruta de la imagen. Por eso acá solo se permiten cadenas. En cambio, las cadenas funcionan en cualquier lugar donde se espere contenido, porque el texto es un tipo válido de contenido.

### 2.1.2 Agregar una bibliografía

Mientras escribís tu informe, necesitás respaldar algunas de tus afirmaciones. Podés agregar una bibliografía a tu documento con la función `bibliography`[3.6.1]. Esta función espera la ruta de un archivo de bibliografía.

El formato nativo de bibliografía de Typst es Hayagriva[A], pero, por compatibilidad, también podés usar archivos BibLaTeX. Como tu compañero ya hizo un relevamiento bibliográfico y te mandó un archivo `.bib`, vas a usar ese. Subí el archivo desde el panel de archivos para acceder a él en Typst.

Una vez que el documento contiene una bibliografía, podés empezar a citarla. Las citas usan la misma sintaxis que las referencias a una etiqueta. Apenas citás una fuente por primera vez, aparece en la sección de bibliografía de tu documento. Typst admite distintos estilos de citas y de bibliografía. Consultá la referencia[3.6.1/4.4] para más detalles.

```
= Métodos
Seguimos los modelos de derretimiento
de glaciares establecidos en @glacier-melt.

#bibliography("works.bib")
```

## Métodos

Seguimos los modelos de derretimiento de glaciares establecidos en [1].

## Bibliografía

[1] R. Hock, «Glacier melt: a review of processes and their modelling», *Progress in Physical Geography: Earth and Environment*, vol. 29, n.º 3, pp. 362-391, 2005, doi: 10.1191/0309133305pp453ra.

### 2.1.3 Matemática

Después de desarrollar la sección de métodos, pasás a lo central del documento: tus ecuaciones. Typst tiene composición matemática integrada y usa su propia notación matemática. Empecemos con una ecuación simple. La encerramos entre signos $ para que Typst sepa que tiene que esperar una expresión matemática:

[A] https://github.com/typst/hayagriva/blob/main/docs/file-format.md

```
La ecuación $Q = rho A v + C$
define el caudal glacial.
```

La ecuación $Q = \rho A v + C$ define el caudal glacial.

La ecuación se compone en línea, en la misma línea que el texto que la rodea. Si en cambio querés que quede en su propia línea, tenés que insertar un solo espacio al principio y al final:

```
El caudal de un glaciar está
definido por la siguiente ecuación:

$ Q = rho A v + C $
```

El caudal de un glaciar está definido por la siguiente ecuación:

$$Q = \rho A v + C$$

Podemos ver que Typst mostró las letras sueltas `Q`, `A`, `v` y `C` tal cual, mientras que tradujo `rho` a una letra griega. El modo matemático siempre muestra las letras sueltas literalmente. En cambio, las secuencias de varias letras se interpretan como símbolos, variables o nombres de funciones. Para indicar una multiplicación entre letras sueltas, poné espacios entre ellas.

Si querés tener una variable formada por varias letras, podés encerrarla entre comillas:

```
El caudal de un glaciar está dado
por la siguiente ecuación:

$ Q = rho A v + "desfase temporal" $
```

El caudal de un glaciar está dado por la siguiente ecuación:

$$Q = \rho A v + \text{desfase temporal}$$

También vas a necesitar una fórmula de sumatoria en tu trabajo. Podemos usar el símbolo `sum` y después especificar el rango de la sumatoria en subíndices y superíndices:

```
Total de suelo desplazado por el flujo glacial:

$ 7.32 beta +
  sum_(i=0)^nabla Q_i / 2 $
```

Total de suelo desplazado por el flujo glacial:

$$7.32 \beta + \sum_{i=0}^{\nabla} \frac{Q_i}{2}$$

Para agregar un subíndice a un símbolo o variable, escribí un carácter _ y después el subíndice. De manera similar, usá el carácter ^ para un superíndice. Si tu subíndice o superíndice está formado por varias cosas, tenés que encerrarlas entre paréntesis.

El ejemplo anterior también nos mostró cómo insertar fracciones: simplemente poné un carácter / entre el numerador y el denominador, y Typst lo convierte automáticamente en una fracción. Los paréntesis se resuelven de forma inteligente, así que podés ingresar tu expresión como lo harías en una calculadora y Typst reemplaza las subexpresiones entre paréntesis por la notación adecuada.

```
Total de suelo desplazado por el flujo glacial:

$ 7.32 beta +
```

```
sum_(i=0)^nabla
  (Q_i (a_i - epsilon)) / 2 $
```

Total de suelo desplazado por el flujo glacial:

$$7.32\beta + \sum_{i=0}^{\nabla} \frac{Q_i(a_i - \varepsilon)}{2}$$

No todas las construcciones matemáticas tienen una sintaxis especial. En su lugar, usamos funciones, igual que la función `image` que vimos antes. Por ejemplo, para insertar un vector columna, podemos usar la función vec[3.8.28]. Dentro del modo matemático, las llamadas a funciones no necesitan empezar con el carácter #.

```
$ v := vec(x_1, x_2, x_3) $
```

$$v := \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Algunas funciones solo están disponibles dentro del modo matemático. Por ejemplo, la función `cal`[3.8.27/1.6] se usa para componer letras caligráficas, que se usan comúnmente para conjuntos. La sección de matemática de la referencia[3.8] ofrece una lista completa de todas las funciones que el modo matemático pone a disposición.

Una cosa más: muchos símbolos, como la flecha, tienen muchas variantes. Podés elegir entre estas variantes agregando un punto y el nombre de un modificador al nombre del símbolo:

```
$ a arrow.squiggly b $
```

$$a \rightsquigarrow b$$

Esta notación también está disponible en el modo marcado, pero ahí el nombre del símbolo tiene que estar precedido por `#sym.`. Mirá la sección de símbolos[3.9.2] para ver una lista de todos los símbolos disponibles.

### 2.1.4 Repaso

Ya viste cómo escribir un documento básico en Typst. Aprendiste a resaltar texto, escribir listas, insertar imágenes, alinear contenido y componer expresiones matemáticas. También aprendiste sobre las funciones de Typst. Hay muchos más tipos de contenido que Typst te permite insertar en tu documento, como tablas[3.6.19], formas[3.11] y bloques de código[3.7.6]. Podés recorrer la referencia[3] para aprender más sobre estas y otras características.

Por el momento, terminaste de escribir tu informe. Ya guardaste un PDF haciendo clic en el botón de descarga de la esquina superior derecha. Sin embargo, pensás que el informe podría verse un poco menos simple. En la próxima sección, vamos a aprender a personalizar el aspecto de tu documento.

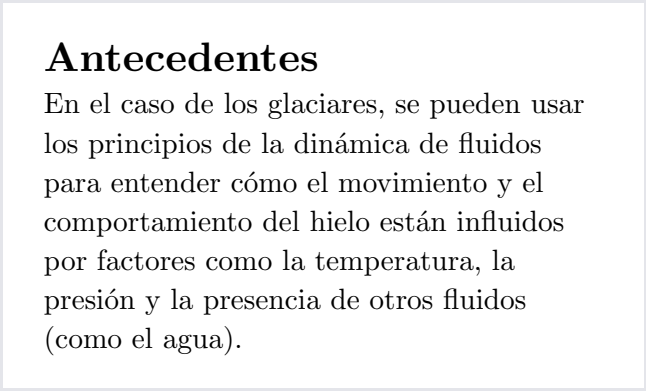
## 2.2 Formato

Hasta ahora escribiste un informe con algo de texto, unas cuantas ecuaciones e imágenes. Sin embargo, todavía se ve muy simple. Tu docente auxiliar aún no sabe que estás usando un nuevo sistema de composición tipográfica, y querés que tu informe encaje con las entregas del resto de los estudiantes. En este capítulo vamos a ver cómo dar formato a tu informe con el sistema de estilos de Typst.

### 2.2.1 Reglas set

Como vimos en el capítulo anterior, Typst tiene funciones que *insertan* contenido (por ejemplo, la función `image`[3.11.7]) y otras que *manipulan* el contenido que reciben como argumentos (por ejemplo, la función `align`[3.10.1]). El primer impulso que podrías tener cuando querés, por ejemplo, cambiar la fuente, es buscar una función que lo haga y envolver el documento completo con ella.

```
#text(font: "New Computer Modern")[
  = Antecedentes
  En el caso de los glaciares, se pueden
  usar los principios de la dinámica de
  fluidos para entender cómo el movimiento
  y el comportamiento del hielo están
  influidos por factores como la
  temperatura, la presión y la presencia
  de otros fluidos (como el agua).
]
```




Un momento: ¿no deberían especificarse todos los argumentos de una función entre paréntesis? ¿Por qué hay un segundo par de corchetes con contenido *después* de los paréntesis? La respuesta es que, como pasarle contenido a una función es algo tan común en Typst, hay una sintaxis especial para eso: en lugar de poner el contenido dentro de la lista de argumentos, podés escribirlo entre corchetes justo después de los argumentos normales, ahorrando puntuación.

Como vimos arriba, eso funciona. Con la función `text`[3.7.12] podemos ajustar la fuente de todo el texto que contiene. Sin embargo, envolver el documento en innumerables funciones y aplicar estilos de forma selectiva y puntual puede volverse engorroso rápidamente.

Por suerte, Typst tiene una solución más elegante. Con las *reglas set*, podés aplicar propiedades de estilo a todas las apariciones de un tipo de contenido. Una regla set se escribe con la palabra clave `set`, seguida del nombre de la función cuyas propiedades querés establecer y una lista de argumentos entre paréntesis.

```
#set text(
  font: "New Computer Modern"
)

= Antecedentes
En el caso de los glaciares, se pueden
usar los principios de la dinámica de
fluidos para entender cómo el movimiento
y el comportamiento del hielo están
influidos por factores como la
temperatura, la presión y la presencia
de otros fluidos (como el agua).
```

## Antecedentes

En el caso de los glaciares, se pueden usar los principios de la dinámica de fluidos para entender cómo el movimiento y el comportamiento del hielo están influidos por factores como la temperatura, la presión y la presencia de otros fluidos (como el agua).

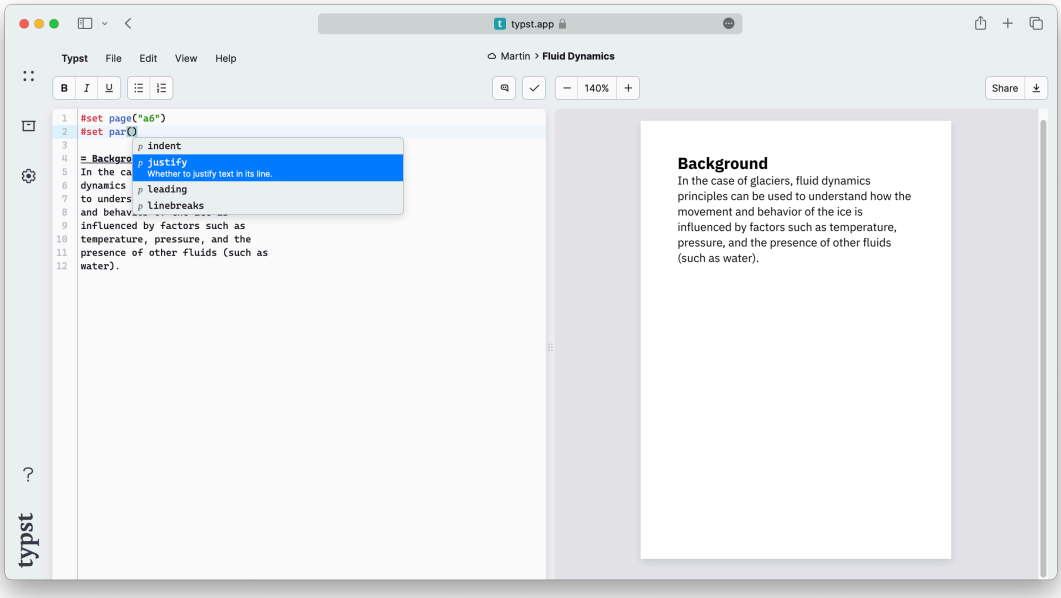
INFO

¿Querés saber en términos más técnicos qué está pasando acá?

Las reglas set se pueden concebir como la definición de valores por defecto para algunos de los parámetros de una función, para todos los usos futuros de esa función.

### 2.2.2 El panel de autocompletado

Si seguiste el tutorial y probaste algunas cosas en la app, es posible que hayas notado que siempre que ingresás un carácter # aparece un panel que te muestra las funciones disponibles y, dentro de una lista de argumentos, los parámetros disponibles. Ese es el panel de autocompletado. Puede ser muy útil mientras escribís tu documento: podés aplicar sus sugerencias presionando la tecla Enter o navegar hasta la opción deseada con las flechas. El panel se puede cerrar presionando la tecla Escape y volver a abrir escribiendo # o presionando `Ctrl` + `Espacio`. Usá el panel de autocompletado para descubrir los argumentos correctos de las funciones. La mayoría de las sugerencias vienen con una pequeña descripción de lo que hacen.




### 2.2.3 Configurar la página

Volviendo a las reglas set: cuando escribís una regla, elegís la función según el tipo de elemento al que querés darle estilo. Esta es una lista de algunas funciones que se usan comúnmente en las reglas set:

- `text`[3.7.12] para establecer la familia de fuente, el tamaño, el color y otras propiedades del texto

- page[3.10.17] para establecer el tamaño de página, los márgenes, los encabezados, activar columnas y los pies de página
- par[3.6.14] para justificar párrafos, establecer el interlineado y más
- heading[3.6.9] para establecer el aspecto de los títulos y activar la numeración
- document[3.6.5] para establecer los metadatos que contiene el PDF de salida, como el título y el autor

No todos los parámetros de una función se pueden establecer. En general, solo se pueden establecer los parámetros que le dicen a una función *cómo* hacer algo, no los que le dicen *con qué* hacerlo. Las páginas de referencia de las funciones indican qué parámetros son configurables.

Agreguemos algunos estilos más a nuestro documento. Queremos márgenes más grandes y una fuente con serifas. A los fines del ejemplo, también vamos a establecer otro tamaño de página.

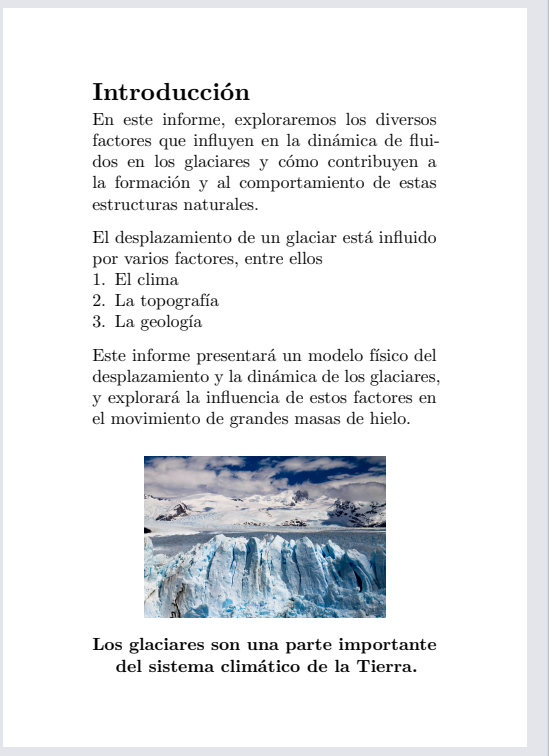
```
#set page(
  paper: "a6",
  margin: (x: 1.8cm, y: 1.5cm),
)
#set text(
  font: "New Computer Modern",
  size: 10pt
)
#set par(
  justify: true,
  leading: 0.52em,
)

= Introducción
En este informe, exploraremos los
diversos factores que influyen en la
dinámica de fluidos en los glaciares
y cómo contribuyen a la formación y
al comportamiento de estas
estructuras naturales.

...

#align(center + bottom)[
  #image("glacier.jpg", width: 70%)

  *Los glaciares son una parte
  importante del sistema
  climático de la Tierra.*
]
```




Hay algunas cosas para destacar acá.

Primero, la regla set de page[3.10.17]. Recibe dos argumentos: el tamaño de la página y los márgenes. El tamaño de página es una cadena de texto. Typst acepta muchos tamaños de página estándar,[3.10.17/4.1] pero también podés especificar un tamaño personalizado. Los márgenes se especifican como un diccionario.[3.5.12] Los diccionarios son colecciones de pares clave-valor. En este caso, las claves son `x` e `y`, y los valores son los márgenes horizontal y vertical, respectivamente. También podríamos haber especificado márgenes separados para cada lado pasando un diccionario con las claves `left`, `right`, `top` y `bottom`.

Lo siguiente es la regla set de text[3.7.12]. Acá establecemos el tamaño de fuente en `10pt` y la familia de fuente en `"New Computer Modern"`. La app de Typst viene con muchas fuentes que podés probar en tu documento. Cuando estás dentro de la lista de argumentos de la función text, podés descubrir las fuentes disponibles en el panel de autocompletado.

También establecimos el espaciado entre líneas (también llamado interlineado o *leading*): se especifica como un valor de longitud[3.10.13], y usamos la unidad `em` para indicar el interlineado relativo al tamaño de la fuente: `1em` equivale al tamaño de fuente actual (que por defecto es `11pt`).

Por último, alineamos la imagen abajo agregando una alineación vertical a nuestra alineación centrada. Las alineaciones vertical y horizontal se pueden combinar con el operador + para obtener una alineación 2D.

### 2.2.4 Un toque de sofisticación

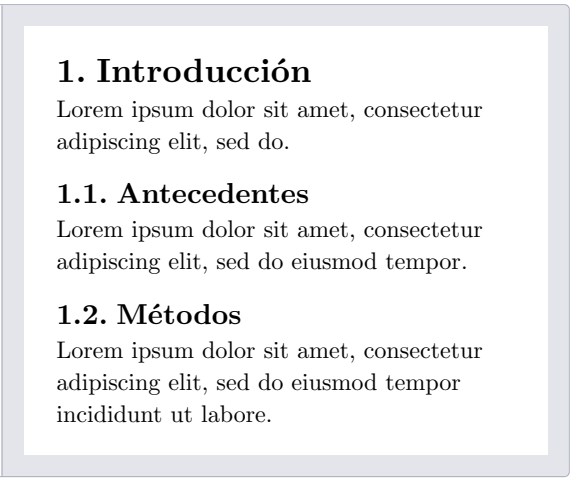
Para estructurar nuestro documento con más claridad, ahora queremos numerar los títulos. Podemos hacerlo estableciendo el parámetro `numbering` de la función `heading`[3.6.9].

```
#set heading(numbering: "1.")

= Introducción
#lorem(10)

== Antecedentes
#lorem(12)

== Métodos
#lorem(15)
```



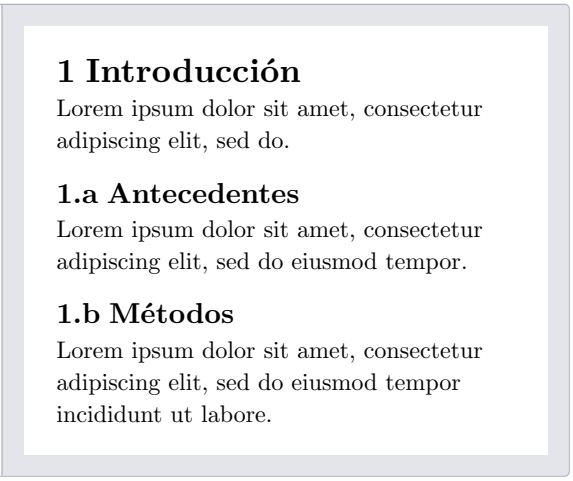

Especificamos la cadena `"1."` como parámetro numbering. Esto le indica a Typst que numere los títulos con números arábigos y que ponga un punto entre el número de cada nivel. También podemos usar letras, números romanos y símbolos[3.6.12] para nuestros títulos:

```
#set heading(numbering: "1.a")

= Introducción
#lorem(10)

== Antecedentes
#lorem(12)

== Métodos
#lorem(15)
```




Este ejemplo también usa la función `lorem`[3.7.3] para generar texto de relleno. Esta función toma un número como argumento y genera esa cantidad de palabras de texto *Lorem Ipsum*.

INFO

¿Te preguntaste por qué las reglas set de text y heading se aplican a todo el texto y a todos los títulos, aunque no se produzcan con las funciones respectivas?

Typst llama internamente a la función `heading` cada vez que escribís `= Conclusion`. De hecho, la llamada a función `#heading[Conclusion]` equivale al marcado de título de arriba. Otros elementos de marcado funcionan de manera similar: son solo *azúcar sintáctico* de las llamadas a funciones correspondientes.

### 2.2.5 Reglas show

Ya estás bastante conforme con cómo quedó. Pero falta arreglar una última cosa: el informe que estás escribiendo es parte de un proyecto más grande, y el nombre de ese proyecto siempre tiene que ir acompañado de un logo, incluso en la prosa.

Evaluás tus opciones. Podrías agregar una llamada `#image("logo.svg")` antes de cada aparición del logo usando buscar y reemplazar. Eso suena muy tedioso. En cambio, quizás podrías definir una función personalizada[3-5.16/4] que siempre devuelva el logo con su imagen. Sin embargo, hay una forma todavía más fácil:

Con las reglas show, podés redefinir cómo Typst muestra ciertos elementos. Especificás qué elementos tiene que mostrar Typst de forma diferente y cómo deben verse. Las reglas show se pueden aplicar a instancias de texto, a muchas funciones e incluso a todo el documento.

```
#show "ArtosFlow": name => box[
  #box(image(
    "logo.svg",
    height: 0.7em,
  ))
  #name
]

Este informe está incluido en el
proyecto ArtosFlow. ArtosFlow es un
proyecto del Instituto Artos.
```

Este informe está incluido en el proyecto ArtosFlow. ArtosFlow es un proyecto del Instituto Artos.

Hay mucha sintaxis nueva en este ejemplo: escribimos la palabra clave `show`, seguida de una cadena de texto que queremos mostrar de forma diferente y de dos puntos. Después, escribimos una función que recibe como argumento el contenido que se va a mostrar. Acá llamamos `name` a ese argumento. Ahora podemos usar la variable `name` en el cuerpo de la función para imprimir el nombre ArtosFlow. Nuestra regla show agrega la imagen del logo delante del nombre y pone el resultado en un box para evitar que se produzcan saltos de línea entre el logo y el nombre. La imagen también se pone dentro de un box, para que no aparezca en su propio párrafo.

Las llamadas a la primera función box y a la función image no necesitaron un # inicial porque no estaban incrustadas directamente en el marcado. Cuando Typst espera código en lugar de marcado, el # inicial no hace falta para acceder a funciones, palabras clave y variables. Esto se puede observar en las listas de parámetros, en las definiciones de funciones y en los bloques de código[3-3].

### 2.2.6 Repaso

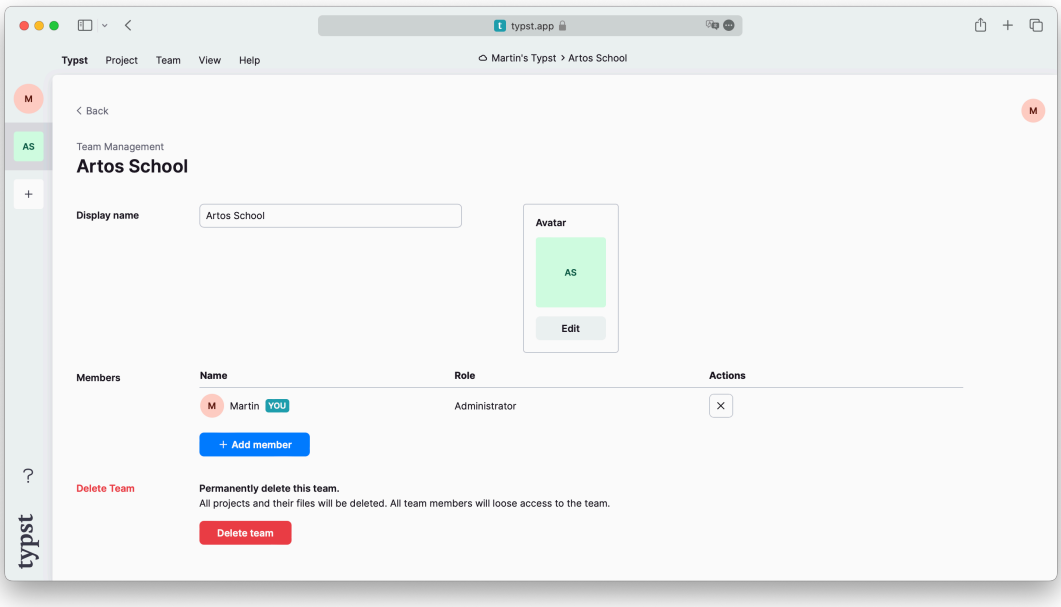
Ahora sabés cómo aplicar formato básico a tus documentos de Typst. Aprendiste a establecer la fuente, justificar los párrafos, cambiar las dimensiones de la página y agregar numeración a los títulos con reglas set. También aprendiste a usar una regla show básica para cambiar cómo aparece el texto en todo tu documento.

Entregaste tu informe. ¡A tu supervisor le gustó tanto que quiere adaptarlo como un trabajo para una conferencia! En la próxima sección, vamos a aprender a dar formato a tu documento como un trabajo científico usando reglas show y funciones más avanzadas.

## 2.3 Estilos avanzados

En los dos capítulos anteriores de este tutorial, aprendiste a escribir un documento en Typst y a cambiar su formato. El informe que escribiste a lo largo de los últimos dos capítulos sacó un diez y tu supervisor quiere basar un trabajo para una conferencia en él. Por supuesto, el trabajo tendrá que cumplir con la guía de estilo de la conferencia. Veamos cómo lograrlo.

Antes de empezar, creemos un equipo, invitemos a tu supervisor y agreguémoslo al equipo. Podés hacerlo volviendo al panel de la app con el ícono de volver de la esquina superior izquierda del editor. Después, elegí el ícono de más en la barra de herramientas izquierda y creá un equipo. Por último, hacé clic en el equipo nuevo y andá a su configuración haciendo clic en "administrar equipo" junto al nombre del equipo. Ahora podés invitar a tu supervisor por correo electrónico.




A continuación, mové tu proyecto al equipo: abrilo, andá a su configuración eligiendo el ícono de engranaje en la barra de herramientas izquierda y seleccioná tu equipo nuevo en el menú desplegable de propietarios. ¡No te olvides de guardar los cambios!

Ahora tu supervisor también puede editar el proyecto y los dos pueden ver los cambios en tiempo real. ¡Podés sumarte a nuestro servidor de Discord[A] para conocer a otros usuarios y probar los equipos con ellos!

### 2.3.1 Las pautas de la conferencia

Las pautas de diseño están disponibles en el sitio web de la conferencia. Veámoslas:

- La fuente debe ser una fuente con serifas de 11pt
- El título debe estar en 17pt y en negrita
- El trabajo contiene un resumen a una columna y un texto principal a dos columnas
- El resumen debe estar centrado
- El texto principal debe estar justificado
- Los títulos de sección de primer nivel deben ser de 13pt, centrados y en versalitas
- Los títulos de segundo nivel van incorporados al párrafo, en cursiva y con el mismo tamaño que el texto del cuerpo
- Por último, las páginas deben ser de tamaño carta de EE. UU., numeradas en el centro del pie de página, y la esquina superior derecha de cada página debe contener el título del trabajo

Ya sabemos hacer muchas de estas cosas, pero para algunas vamos a tener que aprender trucos nuevos.

[A] https://discord.gg/2uDybryKPe

### 2.3.2 Escribir las reglas set correctas

Empecemos escribiendo algunas reglas set para el documento.

```
#set page(
  paper: "us-letter",
  header: align(right)[
    Un modelo de dinámica de fluidos
    para el flujo glaciar
  ],
  numbering: "1",
)
#set par(justify: true)
#set text(
  font: "Libertinus Serif",
  size: 11pt,
)

#lorem(600)
```




Ya conocés la mayor parte de lo que pasa acá. Establecimos el tamaño del texto en `11pt` y la fuente en Libertinus Serif. También activamos la justificación de párrafos y establecimos el tamaño de página en carta de EE. UU.

El argumento `header` es nuevo: con él, podemos proporcionar contenido para llenar el margen superior de cada página. En el encabezado, especificamos el título de nuestro trabajo, como lo pide la guía de estilo de la conferencia. Usamos la función `align` para alinear el texto a la derecha.

Por último, pero no menos importante, el argumento `numbering`. Acá podemos proporcionar un patrón de numeración[3.6.12] que define cómo numerar las páginas. Al establecerlo en `"1"`, Typst muestra solamente el número de página. Establecerlo en `"(1/1)"` habría mostrado la página actual y el total de páginas entre paréntesis. E incluso podríamos haber proporcionado acá una función completamente personalizada para dar formato a nuestro gusto.

### 2.3.3 Crear un título y un resumen

Ahora agreguemos un título y un resumen. Empecemos por el título. Typst trae una función `title`[3.6.21]. Empecemos proporcionando nuestro título como argumento:

```
#title[
  Un modelo de dinámica de fluidos
  para el flujo glaciar
]
```




Podés ver que el título ya está en negrita y tiene algo de espacio alrededor. Sin embargo, está alineado a la izquierda y no mide exactamente 17pt. Por lo tanto, tenemos que ajustar su aspecto. La función title no trae argumentos de fuente o tamaño de texto que podamos establecer. En cambio, esas propiedades se definen en las funciones `text` y `align`.

INFO

¿Cuál es la diferencia entre lo que insertó la función `title` y los títulos que produjimos con signos igual?

Los títulos (headings), incluso los de primer nivel, pueden aparecer varias veces en tu documento, mientras que un título de documento (title) aparece una sola vez, generalmente al principio. Diferenciar entre ambos ayuda a Typst a hacer tu documento accesible para usuarios de tecnologías de asistencia, como los lectores de pantalla.

Cuando queremos personalizar las propiedades de algún elemento dentro de otro tipo de elemento, podemos usar reglas show-set. Primero, usamos `show` para seleccionar qué elemento queremos personalizar. A esto lo llamamos *selector*. Después, escribimos dos puntos. A continuación, escribimos la regla set que debe aplicarse a los elementos que coinciden con el selector. En resumen, la sintaxis se ve así:

```
#show your-selector: set some-element(/* ... */)
```

Recordemos: queremos centrar el título y que mida 17pt. Por lo tanto, necesitamos dos reglas show-set:

- Una con el selector `title` y la regla `set text(size: 17pt)`
- Una con el selector `title` y la regla `set align(center)`

Nuestro ejemplo ahora se ve así:

```
#show title: set text(size: 17pt)
#show title: set align(center)

#title[
  Un modelo de dinámica de fluidos
  para el flujo glaciar
]
```

**Un modelo de dinámica de fluidos para el flujo glaciar**

Esto se ve bien. Agreguemos también la lista de autores: como estamos escribiendo este trabajo junto con nuestro supervisor, vamos a agregar nuestro nombre y el suyo.

```
#grid(
  columns: (1fr, 1fr),
  align(center)[
    Therese Tungsten \
    Instituto Artos \
    #link("mailto:tung@artos.edu")
  ],
  align(center)[
    Dr. John Doe \
    Instituto Artos \
    #link("mailto:doe@artos.edu")
  ]
)
```

**Un modelo de dinámica de fluidos para el flujo glaciar**

| Therese Tungsten | Dr. John Doe |
|---|---|
| Instituto Artos | Instituto Artos |
| tung@artos.edu | doe@artos.edu |

Los dos bloques de autores se disponen uno al lado del otro. Usamos la función `grid`[3.10.10] para crear este diseño. Con una grilla, podemos controlar exactamente el tamaño de cada columna y qué contenido va en cada celda. El argumento `columns` toma un array de longitudes relativas[3.10.21] o fracciones[3.10.9]. En este caso, le pasamos dos tamaños fraccionales iguales, para indicarle que divida el espacio disponible en dos columnas iguales. Después le pasamos dos argumentos de contenido a la función grid: el primero con nuestros datos y el segundo con los de nuestro supervisor. Otra vez usamos la función `align` para centrar el contenido dentro de la columna. La grilla acepta una cantidad arbitraria de argumentos de contenido que especifican las celdas. Las filas se agregan automáticamente, pero también se les puede dar tamaño manualmente con el argumento `rows`.

Si miramos a los autores y el título, están un poco demasiado juntos. Podés resolverlo usando otra regla show-set para configurar el espacio debajo del título. El título, la grilla y todos los demás elementos que Typst dispone de arriba hacia abajo en la página (excepto los párrafos) se llaman *bloques*. Cada bloque se controla con la función `block`[3.10.4]. Ella controla comportamientos como la distancia entre bloques y si un bloque puede contener un salto de página. Eso significa que podemos escribir otra regla show-set que seleccione el título para establecer el espaciado del bloque:

```
#show title: set text(size: 17pt)
#show title: set align(center)
#show title: set block(below: 1.2em)

#title[
  Un modelo de dinámica de fluidos
  para el flujo glaciar
]

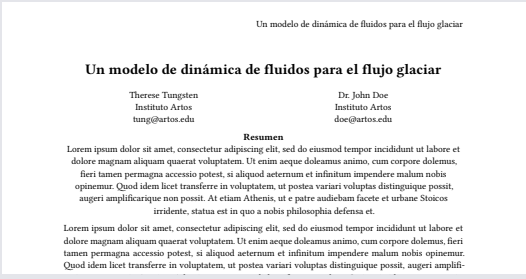
#grid(
  // ...
)
```




Con esta regla show-set, sobrescribimos el espaciado debajo del título. Usamos la unidad `em`: nos permite expresar longitudes como múltiplos del tamaño de la fuente. Acá la usamos para separar el título y la lista de autores exactamente 1,2 veces el tamaño de la fuente. Ahora agreguemos el resumen. Acordate de que la conferencia quiere el resumen con alineación centrada y sin justificar.

```
...

#align(center)[
  #set par(justify: false)
  *Resumen* \
  #lorem(80)
]
```




¡Bien hecho! Una cosa notable es que usamos una regla set dentro del argumento de contenido de `align` para desactivar la justificación del resumen. Esto no afecta el resto del documento, aunque se haya especificado después de la primera regla set, porque los bloques de contenido *delimitan* los estilos. Todo lo que se establece dentro de un bloque de contenido solo afecta al contenido de ese bloque.

Otro ajuste podría ser eliminar la duplicación entre el encabezado y el argumento del elemento title. Como comparten el título, sería conveniente guardarlo en un lugar pensado para contener metadatos del documento. Después necesitaríamos una forma de recuperar el título en ambos lugares. El elemento `document` nos puede ayudar con lo primero: al usarlo en una regla set, podemos guardar metadatos del documento como el título, la descripción y las palabras clave.

```
#set document(title: [Un modelo de dinámica de fluidos para el flujo glaciar])
```

Al exportar un PDF, el título establecido acá aparece en la barra de título de tu lector de PDF. Tu sistema operativo también usa este título para que el archivo se pueda encontrar en las búsquedas. Por último, contribuye a que tu documento sea más accesible y es obligatorio si elegís cumplir con PDF/UA, un estándar de PDF centrado en la accesibilidad.

Ahora necesitamos una forma de recuperar el valor que establecimos en el título principal y en el encabezado. Como la función `title` está diseñada para trabajar junto con el elemento `document`, llamarla sin argumentos simplemente imprime el título. Para el encabezado, vamos a tener que ser más explícitos: como Typst no tiene forma de saber que queremos insertar el título ahí, tenemos que decírselo manualmente.

Con *context*, podemos recuperar el contenido de cualquier valor que hayamos establecido antes en elementos. Cuando usamos la palabra clave `context`, podemos acceder a cualquier propiedad de cualquier elemento, incluida la propiedad title del elemento document. Se usa así:

```
#set document(title: [
  Un modelo de dinámica de fluidos
  para el flujo glaciar
])

...

#set page(
  header: align(
    right + horizon,
    // Obtener la propiedad title
    // del elemento document.
    context document.title,
  ),
  ...
)

#title()

...
```


Un modelo de dinámica de fluidos para el flujo glaciar

**Un modelo de dinámica de fluidos para el flujo glaciar**

Therese Tungsten
Instituto Artos
tung@artos.edu

Dr. John Doe
Instituto Artos
doe@artos.edu

**Resumen**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae sine metu degendae praesidia firmissima. – Filium morte multavit. – Si sine causa, nollem me ab eo delectari, quod ista Platonis, Aristoteli, Theophrasti orationis ornamenta neglexerit. Nam illud quidem physici, credere aliquid esse minimum, quod profecto numquam putavisset, si a Polyaeno, familiari suo, geometrica discere maluisset quam illum etiam ipsum dedocere. Sol Democrito magnus videtur, quippe homini erudito in geometriaque perfecto, huic pedalis fortasse; tantum enim esse omnino in nostris poetis aut inertissimae segnitiae est aut fastidii delicatissimi. Mihi quidem videtur, inermis ac nudus est. Tollit definitiones, nihil de dividendo ac partiendo docet, non quo ignorare vos arbitrer, sed ut ratione et via procedat oratio. Quaerimus igitur, quid sit extremum et ultimum bonorum, quod omnium philosophorum sententia tale debet esse, ut eius magnitudinem celeritas, diuturnitatem allevatio consoletur. Ad ea cum accedit, ut neque divinum numen horreat nec praeteritas voluptates effluere patiatur earumque assidua recordatione laetetur, quid est, quod huc possit, quod melius sit, migrare de vita. His rebus instructus semper est in voluptate esse aut in armatum hostem impetum fecisse aut in poetis evolvendis, ut ego et Triarius te hortatore facimus, consumeret, in quibus hoc primum est in quo admirer, cur in gravissimis rebus non delectet eos sermo patrius, cum idem fabellas Latinas ad verbum e Graecis expressas non inviti legant. Quis enim tam inimicus paene nomini Romano est, qui Ennii Medeam aut Antiopam Pacuvii spernat aut reiciat, quod se isdem Euripidis fabulis delectari dicat, Latinas litteras oderit? Synephebos ego, inquit, potius Caecilii aut Andriam Terentii quam utramque Menandri legam? A quibus tantum dissentio, ut, cum Sophocles vel optime scripserit Electram, tamen male conversam Atilii mihi legendam putem, de quo Lucilius: 'ferreum scriptorem', verum, opinor, scriptorem tamen, ut legendus

1

Primero, fijate cómo llamamos a la función title con paréntesis redondos vacíos. Como no se pasó ningún argumento, usó por defecto lo que establecimos arriba para el elemento document. La distinción entre paréntesis redondos vacíos y corchetes vacíos es importante: mientras que los paréntesis redondos vacíos indican que no pasás nada, los corchetes vacíos significan que pasás un argumento: un bloque de contenido vacío. Si se la llama de esa manera, el título no tendría contenido visible.

A continuación, mirá el encabezado. En lugar del título entre corchetes, usamos la palabra clave context para acceder al título del documento. Esto insertó exactamente lo que establecimos arriba. El rol de context no se limita a acceder a propiedades: con él podés verificar si hay ciertos elementos en el documento, medir las dimensiones físicas de otros y más. Con context, podés construir plantillas potentes que reaccionan a las preferencias del usuario final.

INFO

*¿Por qué hace falta la palabra clave context para acceder a las propiedades de un elemento?*

Normalmente, cuando accedemos a una variable, sabemos exactamente cuál va a ser su valor:

- La variable podría ser una constante incorporada en Typst, como `#sym.pi`
- La variable podría estar definida por un argumento
- La variable podría estar definida o sobrescrita en el ámbito actual

Sin embargo, a veces eso no alcanza. En este capítulo del tutorial, insertamos un encabezado de página con el título. Aunque pasamos una sola pieza de contenido para el encabezado, podríamos querer que distintas páginas tengan encabezados diferentes. Por ejemplo, podríamos querer imprimir el nombre del capítulo o usar el número de página. Cuando usamos context, podemos escribir un único bloque de contexto que le indica a Typst que mire dónde está insertado, busque el último título, el número de página actual o cualquier otra cosa, y siga desde ahí. Eso significa que el mismo bloque de contexto, insertado en páginas distintas, puede producir resultados diferentes.

Para más información, leé sobre context en su documentación[3,4] después de completar este tutorial.

### 2.3.4 Agregar columnas y títulos

Lamentablemente, el trabajo de arriba parece una pared de plomo. Para arreglarlo, agreguemos algunos títulos y pasemos nuestro trabajo a un diseño de dos columnas. Por suerte, es fácil: solo tenemos que ampliar nuestra regla set de `page` con el argumento `columns`.

Al agregar `columns: 2` a la lista de argumentos, envolvimos todo el documento en dos columnas. Sin embargo, eso también afectaría al título y a la lista de autores. Para que sigan ocupando todo el ancho de la página, podemos envolverlos en una llamada a la función `place`[3.10.19]. Place espera como argumentos posicionales una alineación y el contenido que debe ubicar. Con el argumento con nombre `scope`, podemos decidir si los elementos se ubican en relación con la columna actual o con su contenedor padre (la página). Hay una cosa más para configurar: si no se proporcionan otros argumentos, `place` saca su contenido del flujo del documento y lo posiciona sobre el resto del contenido, sin afectar el diseño del otro contenido de su contenedor:

```
#place(
  top + center,
  rect(fill: black),
)
#lorem(30)
```

Lorem ipsum dolo[redacted] consectetur adipiscing elit, sed[redacted]od tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

Si no hubiéramos usado `place` acá, el cuadrado estaría en su propia línea, pero así se superpone con las pocas líneas de texto que le siguen. Del mismo modo, ese texto actúa como si no hubiera cuadrado. Para cambiar este comportamiento, podemos pasar el argumento `float: true` para asegurarnos de que el espacio que ocupa el elemento ubicado en la parte superior o inferior de la página no sea ocupado por ningún otro contenido.

```
#set page(
  paper: "us-letter",
  header: align(
    right + horizon,
    context document.title,
  ),
  numbering: "1",
  columns: 2,
)

#place(
  top + center,
  float: true,
  scope: "parent",
  clearance: 2em,
)[
  ...

  #par(justify: false)[
    *Resumen* \
    #lorem(80)
  ]
]

= Introducción
#lorem(300)

= Trabajos relacionados
#lorem(200)
```



Un modelo de dinámica de fluidos para el flujo glaciar

**Un modelo de dinámica de fluidos para el flujo glaciar**

Therese Tungsten
Instituto Artos
tung@artos.edu

Dr. John Doe
Instituto Artos
doe@artos.edu

**Resumen**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae sine metu degendae praesidia firmissima. – Filium morte multavit. – Si sine causa, nollem me ab eo delectari, quod ista Platonis, Aristoteli, Theophrasti orationis ornamenta neglexerit. Nam illud quidem physici, credere aliquid esse minimum, quod profecto numquam putavisset, si a Polyaeno, familiari suo, geometrica discere maluisset quam illum etiam ipsum dedocere. Sol Democrito magnus videtur, quippe homini erudito in geometriaque perfecto, huic pedalis fortasse; tantum enim esse omnino in nostris poetis aut inertissimae segnitiae est aut fastidii delicatissimi. Mihi quidem videtur, inermis ac nudus est. Tollit definitiones, nihil de dividendo ac partiendo docet, non quo ignorare vos arbitrer, sed ut ratione et via.

**Introducción**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae sine metu degendae praesidia firmissima. – Filium morte multavit. – Si sine causa,

**Trabajos relacionados**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptа-

1

En este ejemplo, también usamos el argumento `clearance` de la función `place` para dar el espacio entre ella y el cuerpo, en lugar de usar la función v[3.10.27]. También podemos eliminar las llamadas explícitas a `align(center, ..)` alrededor de las distintas partes, ya que heredan la alineación centrada de la ubicación.

Ahora solo queda una cosa por hacer: darle estilo a los títulos. Tenemos que centrarlos y usar versalitas. Estas propiedades no están disponibles en la función `heading`, así que vamos a tener que escribir algunas reglas show-set y una regla show:

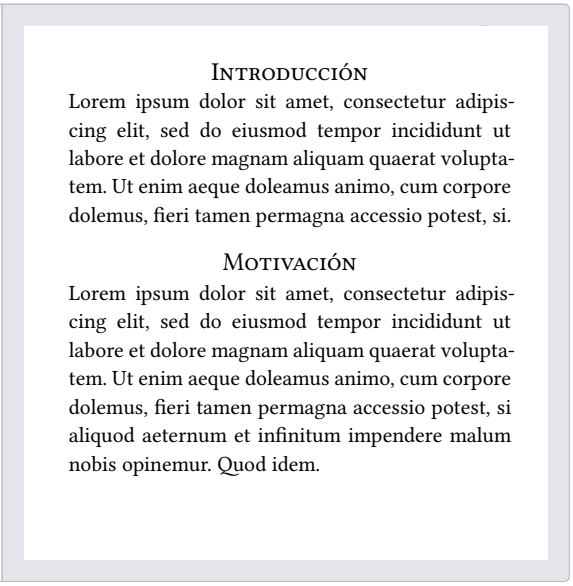
- Una regla show-set para que los títulos estén centrados
- Una regla show-set para que los títulos midan 13pt y usen el peso regular
- Una regla show para envolver los títulos en una llamada a la función `smallcaps`

```
#show heading: set align(center)
#show heading: set text(
  size: 13pt,
  weight: "regular",
)
#show heading: smallcaps

...

= Introducción
...

== Motivación
...
```

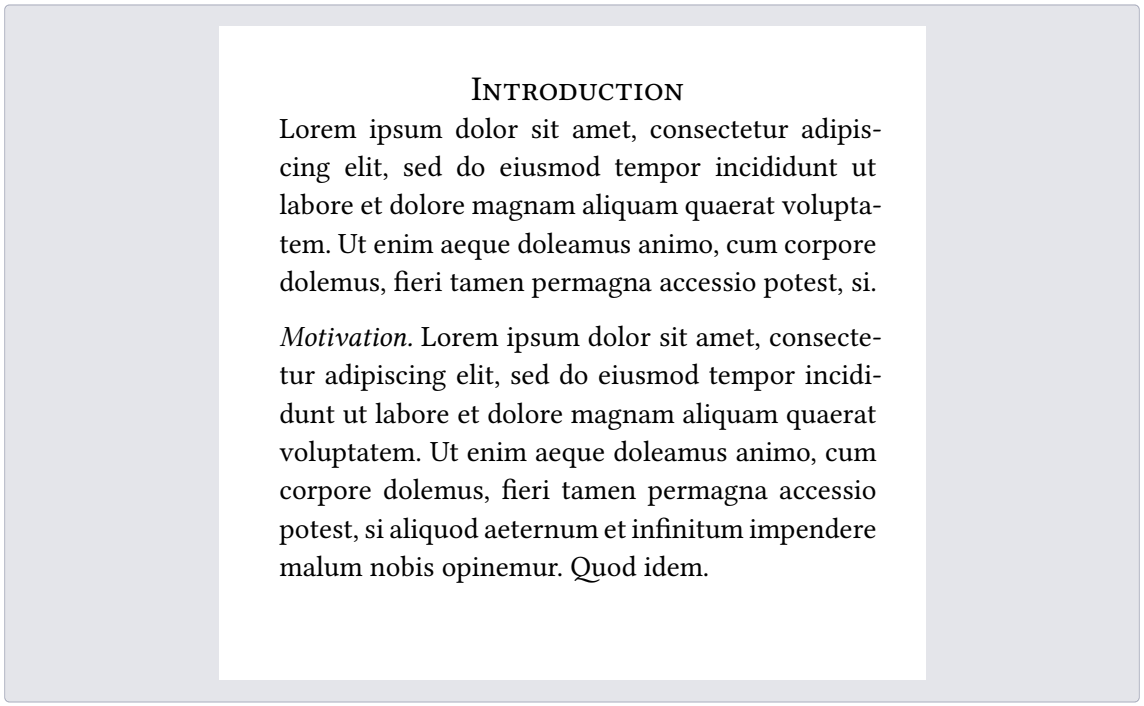



¡Esto se ve genial! Usamos reglas show que se aplican a todos los títulos. En la última regla show, aplicamos la función `smallcaps` al título completo. Como vamos a ver en el próximo ejemplo, también podemos proporcionar una regla personalizada para sobrescribir por completo el aspecto por defecto de los títulos.

El único problema que queda es que todos los títulos se ven iguales ahora. Las subsecciones "Motivación" y "Planteo del problema" deberían ser títulos en cursiva incorporados al párrafo, pero ahora mismo son indistinguibles de los títulos de sección. Lo podemos arreglar usando un selector `where` en nuestra regla show: es un método[3.3.7] que podemos llamar sobre los títulos (y otros elementos) y que nos permite filtrarlos por sus propiedades. Podemos usarlo para diferenciar entre los títulos de sección y de subsección:

```
#show heading.where(level: 1): set align(center)
#show heading.where(level: 1): set text(size: 13pt, weight: "regular")
#show heading.where(level: 1): smallcaps

#show heading.where(level: 2): set text(
  size: 11pt,
  weight: "regular",
  style: "italic",
)
#show heading.where(level: 2): it => {
  it.body + [.]
}
```




En este ejemplo, primero acotamos nuestras reglas anteriores a los títulos de primer nivel usando `.where(level: 1)` para hacer el selector más específico. Después, agregamos una regla show-set para el segundo nivel de títulos. Por último, necesitamos una regla show con una función personalizada: los títulos encierran su contenido en un bloque por defecto. Esto hace que el título tenga su propia línea. Sin embargo, queremos que quede incorporado al texto, así que tenemos que proporcionar nuestra propia regla show para deshacernos de este bloque.

A la regla le damos una función que toma el título como parámetro. Por convención, este parámetro se llama `it`, pero puede tener otro nombre. El parámetro se puede usar como contenido y simplemente muestra el título por defecto completo. Como alternativa, cuando queremos construir nuestro propio título, podemos usar sus campos como `body`, `numbering` y `level` para componer un aspecto personalizado. Acá simplemente imprimimos el cuerpo del título con un punto al final y omitimos el bloque que produce la regla show incorporada. Tené en cuenta que este título ya no va a reaccionar a las reglas set de numeración de títulos y similares, porque no usamos explícitamente `it.numbering` en la regla show. Si escribís reglas show como esta y querés que el documento siga siendo personalizable, vas a tener que tener en cuenta estos campos.

¡Esto se ve genial! Escribimos reglas show que se aplican selectivamente a los títulos de primer y segundo nivel. Usamos un selector `where` para filtrar los títulos por su nivel. Después mostramos los títulos de subsección incorporados al párrafo. También agregamos automáticamente un punto al final de los títulos de subsección.

Repasemos la guía de estilo de la conferencia:
- ✓ La fuente debe ser una fuente con serifas de 11pt
- ✓ El título debe estar en 17pt y en negrita
- ✓ El trabajo contiene un resumen a una columna y un texto principal a dos columnas
- ✓ El resumen debe estar centrado
- ✓ El texto principal debe estar justificado
- ✓ Los títulos de sección de primer nivel deben estar centrados, en versalitas y en 13pt
- ✓ Los títulos de segundo nivel van incorporados al párrafo, en cursiva y con el mismo tamaño que el texto del cuerpo
- ✓ Por último, las páginas deben ser de tamaño carta de EE. UU., numeradas en el centro, y la esquina superior derecha de cada página debe contener el título del trabajo

Ahora cumplimos con todos estos estilos y ¡podemos enviar el trabajo a la conferencia! El trabajo terminado se ve así:



A fluid dynamic model for glacier flow

**A fluid dynamic model for glacier flow**

Therese Tungsten
Artos Institue
tungsten@artos.edu

Dr. John Doe
Artos Institue
doe@artos.edu

**Abstract**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et.

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis.

*Motivation.* Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere.

*Problem Statement.* Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Related Work

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eve-

1

## 2.3.5 Review

Ya aprendiste a crear títulos, encabezados y pies de página, a usar funciones, reglas show-set y ámbitos para sobrescribir estilos localmente, a crear diseños más complejos con la función `grid`[3.10.10], a acceder a las propiedades de los elementos con context y a escribir reglas show para funciones individuales y para todo el documento. También aprendiste a usar el selector `where`[3.2.2] para filtrar los títulos por su nivel.

¡El trabajo fue un gran éxito! Conociste a muchos investigadores con intereses afines en la conferencia y estás planeando un proyecto que esperás publicar en el mismo lugar el año que viene. Sin embargo, vas a tener que escribir un nuevo trabajo con la misma guía de estilo, así que quizás ahora quieras crear una plantilla que les ahorre tiempo a vos y a tu equipo.

En la próxima sección, vamos a aprender a crear plantillas que se pueden reutilizar en varios documentos. Es un tema más avanzado, así que sentite libre de volver a él más tarde si ahora no tenés ganas.

## 2.4 Crear una plantilla

En los tres capítulos anteriores de este tutorial, aprendiste a escribir un documento en Typst, a aplicar estilos básicos y a personalizar su aspecto en profundidad para cumplir con la guía de estilo de una editorial. Como el trabajo que escribiste en el capítulo anterior fue un éxito rotundo, te pidieron que escribas un artículo de seguimiento para la misma conferencia. Esta vez querés tomar el estilo que creaste en el capítulo anterior y convertirlo en una plantilla reutilizable. En este capítulo vas a aprender a crear una plantilla que vos y tu equipo puedan usar con una sola regla show. ¡Empecemos!

### 2.4.1 Reutilizar datos con variables

En los capítulos anteriores, la mayor parte del contenido del documento se ingresó a mano. En el tercer capítulo usamos el elemento `document` y context para reducir la repetición e ingresar el título una sola vez. Pero en la práctica puede haber muchas más cosas que aparecen varias veces en tu documento. Hay varias buenas razones para definir estos valores repetidos una sola vez:

1. Hace que cambiarlos más adelante sea más fácil
2. Te permite encontrar rápidamente todos los lugares donde usaste algo
3. Facilita mantener la coherencia en todo el documento
4. Para segmentos repetidos largos o difíciles de escribir, suele ser más cómodo tipear un nombre de variable más corto

Si estuvieras usando un procesador de texto convencional, podrías recurrir a un valor provisorio para buscarlo más tarde. En Typst, en cambio, podés usar variables para guardar contenido de forma segura y reutilizarlo en todo tu documento mediante el nombre de la variable.

La técnica de usar context para reproducir la propiedad de un elemento que aprendimos antes no siempre es la más apropiada para esto: los elementos incorporados de Typst se centran en propiedades semánticas, como el título y la descripción de un documento, o en cosas que se relacionan directamente con la composición tipográfica, como el tamaño del texto.

Para nuestro ejemplo, queremos ver la pronunciación de Typst. Una de las mejores formas de transcribir la pronunciación es el Alfabeto Fonético Internacional (AFI, o IPA por sus siglas en inglés). Pero como usa caracteres que no se encuentran en los teclados comunes, escribir IPA repetidamente puede volverse engorroso. Entonces, definamos una variable a la que podamos hacer referencia varias veces.

```
#let ipa = [taɪpst]
```

Acá usamos una palabra clave nueva, `let`, para indicar la definición de una variable. Después ponemos el nombre de nuestra variable, en este caso, `ipa`. Por último, escribimos un signo igual y el valor de nuestra variable. Va entre corchetes porque es contenido, igual que cuando llamás a una función que acepta contenido. En otras palabras, esta sintaxis refleja la frase "*Sea la variable* `ipa` *con el valor* `[taɪpst]`."

Ahora podemos usar la variable en nuestro documento:

```
#let ipa = [taɪpst]

La forma canónica de
pronunciar Typst es #ipa.

#table(
  columns: (1fr, 1fr),
  [Nombre], [Typst],
  [Pronunciación], ipa,
)
```

La forma canónica de pronunciar Typst es taɪpst.

| Nombre | Typst |
|---|---|
| Pronunciación | taɪpst |

En el ejemplo podés ver que la variable se puede usar tanto en el marcado (con el prefijo #) como en una llamada a función (escribiendo solo su nombre). Por supuesto, podemos cambiar el valor de la variable y todas sus apariciones van a cambiar automáticamente. Hagamos un poco más claro qué es IPA y qué es prosa normal, mostrando el IPA en cursiva. También usamos barras, que por convención suelen encerrar el IPA.

```
#let ipa = text(
  style: "italic",
)[/taɪpst/]

La forma canónica de
pronunciar Typst es #ipa.

#table(
  columns: (1fr, 1fr),
  [Nombre], [Typst],
  [Pronunciación], ipa,
)
```

La forma canónica de pronunciar Typst es */taɪpst/*.

| Nombre | Typst |
|---|---|
| Pronunciación | */taɪpst/* |

Acá llamamos a la función text y asignamos su *valor de retorno* a la variable. Cuando llamás a una función, ella procesa sus argumentos y después devuelve otro valor (a menudo, contenido). Hasta ahora, en este tutorial llamamos a la mayoría de las funciones directamente en el marcado, así: `#text(fill: red)[CRIMSON!]`. Esta llamada a la función text devuelve el texto rojo como valor de retorno. Como la pusimos en el marcado, su valor de retorno se insertó de inmediato en el contenido que escribimos. Con las variables, en cambio, podemos guardarlo para usarlo más tarde o componerlo con otros valores.

Las variables no se limitan a guardar contenido: pueden guardar cualquier tipo de dato que Typst conozca. A lo largo de este tutorial usaste muchos tipos de datos cuando se los pasaste a las funciones incorporadas de Typst. Este es un ejemplo que asigna cada uno de ellos a una variable:

```
// Contenido con marcado adentro
#let blind-text = [_Lorem ipsum_ dolor sit amet]

// Cadenas de texto sin formato
#let funny-font = "MS Comic Sans"

// Longitudes absolutas (mirá también pt, in, ...)
#let mile = 160934cm

// Longitudes relativas al tamaño de la fuente
#let double-space = 2em

// Proporciones
#let progress = 80%

// Números enteros
#let answer = 42

// Booleanos
#let truth = false

// Alineación horizontal y vertical
#let focus = center
```

En este capítulo del tutorial vas a aprovechar las variables y tus propias funciones para construir plantillas que se puedan reutilizar en varios documentos.

### 2.4.2 Una plantilla de juguete

En Typst, las plantillas son funciones en las que podés envolver todo tu documento. Para aprender a hacerlo, primero repasemos cómo escribir tus propias funciones. Pueden hacer lo que quieras, así que ¿por qué no volvernos un poco locos?

```
#let amazed(term) = box[✨ #term ✨]

Sos #amazed[genial]!
```

Sos ✨ genial ✨!

Si comparás esto con la sección anterior, quizás notaste que se parece mucho a una definición de variable con `let`. Tu instinto es correcto: las funciones son simplemente otro tipo de dato. Acá definimos la variable `amazed` y le asignamos una función que toma un solo argumento, `term`, y devuelve contenido con el `term` rodeado de destellos. También pusimos todo dentro de un box[3.10.5] para que el término que nos asombra no pueda separarse de sus destellos por un salto de línea. La sintaxis especial de definición de funciones hace la definición más corta y legible, pero también podés usar la sintaxis normal de definición de variables (mirá la referencia de scripting[3.3.3] para más detalles). Después de su definición, podemos llamar a la función igual que a todas las funciones incorporadas.

Muchas de las funciones que vienen con Typst tienen parámetros con nombre opcionales. Nuestras funciones también pueden tenerlos. Agreguemos a nuestra función un parámetro que nos permita elegir el color del texto. Tenemos que proporcionar un color por defecto por si no se indica el parámetro.

```
#let amazed(term, color: blue) = {
  text(color, box[✨ #term ✨])
}

Sos #amazed[genial]!
Esto es #amazed(color: purple)[asombroso]!
```

Sos ✨ genial ✨! Esto es ✨ asombroso ✨!

Las plantillas ahora funcionan envolviendo todo nuestro documento en una función personalizada como `amazed`. ¡Pero envolver un documento entero en una llamada a una función gigante sería engorroso! En cambio, podemos usar una regla show de "todo" para lograr lo mismo con un código más limpio. Para escribir una regla show así, poné dos puntos justo después de la palabra clave show y proporcioná una función. A esta función se le pasa el resto del documento como parámetro. La función puede hacer entonces lo que quiera con ese contenido. Como la función `amazed` se puede llamar con un único argumento de contenido, podemos simplemente pasarla por nombre a la regla show. Probémoslo:

```
#show: amazed
Elijo concentrarme en lo bueno
de mi vida y soltar cualquier
pensamiento o creencia negativa.
De hecho, ¡soy increíble!
```

✨ Elijo concentrarme en lo bueno de mi vida y soltar cualquier pensamiento o creencia negativa. De hecho, ¡soy increíble! ✨

Ahora todo nuestro documento se va a pasar a la función `amazed`, como si la hubiéramos envuelto alrededor de él. Por supuesto, esto no es especialmente útil con esta función en particular, pero combinado con reglas set y argumentos con nombre puede ser muy potente.

### 2.4.3 Incorporar reglas set y show

Para aplicar algunas reglas set y show a nuestra plantilla, podemos usar `set` y `show` dentro de un bloque de contenido en nuestra función y después insertar el documento en ese bloque de contenido.

```
#let template(doc) = [
  #set text(font: "Inria Serif")
  #show "algo genial": [Typst]
  #doc
]
```

Hoy estoy aprendiendo Typst. ¡Hasta ahora viene yendo muy bien!

```
#show: template
Hoy estoy aprendiendo algo genial.
¡Hasta ahora viene yendo muy bien!
```

Igual que ya descubrimos en el capítulo anterior, las reglas set se aplican a todo lo que está dentro de su bloque de contenido. Como la regla show de "todo" le pasa nuestro documento completo a la función `template`, la regla set de text y la regla show de cadena de nuestra plantilla se aplican a todo el documento. Usemos este conocimiento para crear una plantilla que reproduzca el estilo del cuerpo del trabajo que escribimos en el capítulo anterior.

```
#let conf(title, doc) = {
  set page(
    paper: "us-letter",
    header: align(
      right + horizon,
      title
    ),
    columns: 2,
    ...
  )
  set par(justify: true)
  set text(
    font: "Libertinus Serif",
    size: 11pt,
  )

  // Reglas show de los títulos.
  ...

  doc
}

#show: doc => conf(
  [Título del trabajo],
  doc,
)

= Introducción
...
```




Copiamos y pegamos la mayor parte de ese código del capítulo anterior. Las dos diferencias son estas:

1. Envolvimos todo en la función `conf` usando una regla show de "todo". La función aplica algunas reglas set y show y, al final, devuelve el contenido que recibió.
2. Además, usamos un bloque de código entre llaves en lugar de un bloque de contenido. De esta manera, no necesitamos poner el prefijo # a todas las reglas set y llamadas a funciones. A cambio, ya no podemos escribir marcado directamente en el bloque de código.

Fijate también de dónde viene el título: antes lo teníamos dentro de una variable. Ahora lo recibimos como primer parámetro de la función de la plantilla. Para hacerlo, le pasamos una clausura (es decir, una función sin nombre que se usa de inmediato) a la regla show de "todo". Lo hicimos porque la función `conf` espera dos argumentos posicionales, el título y el cuerpo, pero la regla show solo pasa el cuerpo. Por lo tanto, agregamos una nueva definición de función que nos permite establecer el título del trabajo y usar el único parámetro de la regla show.

### 2.4.4 Plantillas con argumentos con nombre

Nuestro trabajo del capítulo anterior tenía un título y una lista de autores. Podemos mantener el título como metadato de `document`[3.6.5], pero nuestra plantilla también debería aceptar una lista de autores con sus

filiaciones y el resumen del trabajo. Los vamos a agregar como argumentos con nombre. Al final, queremos que funcione así:

```
#set document(title: [
  Un modelo de dinámica de fluidos para
  el flujo glaciar
])

#show: doc => conf(
  authors: (
    (
      name: "Theresa Tungsten",
      affiliation: "Instituto Artos",
      email: "tung@artos.edu",
    ),
    (
      name: "Eugene Deklan",
      affiliation: "Honduras State",
      email: "e.deklan@hstate.hn",
    ),
  ),
  abstract: lorem(80),
  doc,
)

...
```

Construyamos esta nueva función de plantilla. El título se puede mostrar con la función `title`[3.6.21] y se accede a él mediante `document.title`, así que la plantilla solo necesita los parámetros con nombre `authors` y `abstract`, con valores por defecto vacíos. A continuación, copiamos en la plantilla el código que genera el título, el resumen y los autores del capítulo anterior, reemplazando los datos fijos por los parámetros.

El nuevo parámetro `authors` espera un array[3.5.2] de diccionarios[3.5.12] con las claves `name`, `affiliation` y `email`. Como podemos tener una cantidad arbitraria de autores, determinamos dinámicamente si necesitamos una, dos o tres columnas para la lista de autores. Primero, determinamos la cantidad de autores usando el método `.len()`[3.5.2/3.1] sobre el array `authors`. Después, establecemos la cantidad de columnas como el mínimo entre esa cantidad y tres, de modo que nunca creemos más de tres columnas. Si hay más de tres autores, se inserta una fila nueva. Para eso, también agregamos un parámetro `row-gutter` a la función `grid`. De lo contrario, las filas quedarían demasiado juntas. Para extraer los datos de los autores del diccionario, usamos la sintaxis de acceso a campos[3.3.6].

Todavía tenemos que proporcionar un argumento a la grilla por cada autor: acá es donde resulta útil el método `map`[3.5.2/3.15] del array. Toma como argumento una función que se llama con cada elemento del array. Le pasamos una función que da formato a los datos de cada autor y devuelve un nuevo array con valores de contenido. Ahora tenemos un array de valores que queremos usar como múltiples argumentos de la grilla. Podemos hacerlo con el operador `spread`[3.5.1]. Toma un array y aplica cada uno de sus elementos como un argumento separado de la función.

La función de plantilla resultante se ve así:

```
#let conf(
  authors: (),
  abstract: [],
  doc,
) = {
  // Las reglas set y show de antes.
  // ...

  place(
    top + center,
    float: true,
```

```
    scope: "parent",
    clearance: 2em,
    {
      title()

      let count = authors.len()
      let ncols = calc.min(count, 3)
      grid(
        columns: (1fr,) * ncols,
        row-gutter: 24pt,
        ..authors.map(author => [
          #author.name \
          #author.affiliation \
          #link("mailto:" + author.email)
        ]),
      )

      par(justify: false)[
        *Resumen* \
        #abstract
      ]

    }
  )

  doc
}
```

### 2.4.5 Un archivo separado

La mayoría de las veces, una plantilla se especifica en otro archivo y luego se importa en el documento. De este modo, el archivo principal en el que escribís se mantiene ordenado y tu plantilla se reutiliza fácilmente. Creá un archivo de texto nuevo en el panel de archivos haciendo clic en el botón de más y llamalo `conf.typ`. Mové la definición de la función `conf` dentro de ese archivo nuevo. Ahora podés acceder a ella desde tu archivo principal agregando una importación antes de la regla show. Especificá la ruta del archivo entre la palabra clave `import` y dos puntos, y después nombrá la función que querés importar.

Otra cosa que podés hacer para que aplicar plantillas sea un poco más elegante es usar el método `.with`[3.5.16/8.1] de las funciones para completar de antemano todos los argumentos con nombre. De esta manera, evitás escribir una clausura y agregar el argumento de contenido al final de la lista de tu plantilla. Las plantillas de Typst Universe[A] están diseñadas para funcionar con este estilo de llamada a función.

[A] https://typst.app/universe

```
#import "conf.typ": conf

#set document(title: [
  Un modelo de dinámica de fluidos para
  el flujo glaciar
])

#show: conf.with(
  authors: (
    (
      name: "Theresa Tungsten",
      affiliation: "Instituto Artos",
      email: "tung@artos.edu",
    ),
    (
      name: "Eugene Deklan",
      affiliation: "Honduras State",
      email: "e.deklan@hstate.hn",
    ),
  ),
  abstract: lorem(80),
)

= Introduction
#lorem(90)

== Motivación
#lorem(140)

== Planteo del problema
#lorem(50)

= Trabajos relacionados
#lorem(200)
```



Un modelo de dinámica de fluidos para el flujo glaciar

**Un modelo de dinámica de fluidos para el flujo glaciar**

Theresa Tungsten
Instituto Artos
tung@artos.edu

Eugene Deklan
Honduras State
e.deklan@hstate.hn

**Resumen**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et.

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis.

*Motivación.* Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae.

Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere.

*Planteo del problema.* Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Trabajos relacionados

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae.

1

¡Ya convertimos el trabajo de la conferencia en una plantilla reutilizable para esa conferencia! ¿Por qué no compartirla en el foro[A] o en el servidor de Discord de Typst[B] para que otros también puedan usarla?

### 2.4.6 Review

¡Felicitaciones, completaste el tutorial de Typst! En esta sección aprendiste a definir tus propias funciones y a crear y aplicar plantillas que definen estilos de documento reutilizables. Llegaste lejos y aprendiste mucho. Ahora podés usar Typst para escribir tus propios documentos y compartirlos con otros.

Todavía somos un proyecto muy joven y estamos buscando comentarios. Si tenés preguntas, sugerencias o encontraste un error, contanos en el foro[C], en nuestro servidor de Discord[D], en GitHub[E] o mediante el formulario de comentarios de la app web (siempre disponible en el menú Ayuda).

¿Y entonces qué estás esperando? Registrate[F] ¡y escribí algo!

[A] https://forum.typst.app/
[B] https://discord.gg/2uDybryKPe
[C] https://forum.typst.app/
[D] https://discord.gg/2uDybryKPe
[E] https://github.com/typst/typst/
[F] https://typst.app

# 3 Referencia

Esta documentación de referencia es una guía completa de toda la sintaxis, los conceptos, las funciones, los tipos y otras definiciones de Typst. Usá la referencia para responder preguntas específicas sobre Typst y para ampliar tu comprensión de las características disponibles.

Si sos completamente nuevo en Typst, te recomendamos empezar con el tutorial[2] y después volver a la referencia para aprender más sobre las características de Typst a medida que las necesites.

### 3.0.1 Lenguaje

La referencia empieza cubriendo los fundamentos del lenguaje Typst. Primero, damos un panorama de la sintaxis de Typst.[3.1] Las siguientes secciones cubren conceptos centrales del lenguaje Typst, como dar estilo a los documentos,[3.2] usar las capacidades de scripting de Typst[3.3] y razonar sobre el contenido de tu documento.[3.4]

### 3.0.2 Biblioteca

A partir de Fundamentos[3.5], la referencia incluye secciones sobre todas las funciones, los tipos y otras definiciones que ofrece la *biblioteca estándar* del lenguaje Typst.

Las secciones de definiciones están agrupadas por tema. Por ejemplo, si querés explorar todas las herramientas que ofrece Typst para ajustar dónde caen los elementos en la página, deberías empezar en la sección Diseño[3.10]. Si en cambio preferís saber más sobre los formatos a los que puede exportar Typst, deberías recorrer la sección de formatos[3.14].

### 3.0.3 Cómo leer la referencia

Esta referencia usa algunas convenciones gráficas y etiquetas para que puedas recorrer rápidamente sus secciones.

`str` Estas píldoras indican que un valor es de un tipo particular. El capítulo de cada tipo usa la píldora respectiva como título. Los tipos similares comparten un color. Por ejemplo, todos los tipos numéricos tienen el mismo color.

*Element* Algunas funciones están etiquetadas como elementos. Esto significa que se pueden usar con reglas set y show. Algunos elementos se pueden ubicar[3.12.3] y usar con la función `query`[3.12.6]. Los elementos generalmente producen salida visible en el documento. Es posible que uses elementos aunque no llames a funciones, ya que hay marcado dedicado para algunos elementos.

*Contextual* Estas funciones pueden razonar sobre el contenido de tu documento. Solo se pueden usar cuando hay *contexto* disponible, por ejemplo mediante un bloque de contexto. Consultá la sección Contexto[3.4] para más información.

*Required* Aparece en un parámetro de una función si llamar a la función sin ese parámetro produciría un error.

*Positional* Aparece en un parámetro de una función que se especifica sin nombre de parámetro ni dos puntos. En cambio, Typst usará el orden de los parámetros para determinar qué argumento es cuál. Los parámetros que no están marcados como posicionales son parámetros *con nombre*.

*Variadic* Aparece en los parámetros de funciones que se pueden especificar varias veces.

*Settable* Aparece en los parámetros de funciones de elementos que se pueden personalizar con una regla set.

## 3.1 Sintaxis

Typst es un lenguaje de marcado. Esto significa que podés usar una sintaxis simple para resolver tareas comunes de diseño. La sintaxis de marcado liviana se complementa con reglas set y show, que te permiten dar estilo a tu documento de forma fácil y automática. Todo esto está respaldado por un lenguaje de scripting estrechamente integrado, con funciones incorporadas y definidas por el usuario.

### 3.1.1 Modos

Typst tiene tres modos sintácticos: marcado, matemático y código. El modo marcado es el que se usa por defecto en un documento de Typst, el modo matemático te permite escribir fórmulas matemáticas y el modo código te permite usar las funciones de scripting de Typst.

Podés cambiar a un modo específico en cualquier momento consultando la siguiente tabla:

| Modo nuevo | Sintaxis | Ejemplo |
|---|---|---|
| Código | Anteponé # al código | `Number: #(1 + 2)` |
| Matemática | Rodeá la ecuación con `$..$` | `$-x$ is the opposite of $x$` |
| Marcado | Rodeá el marcado con `[..]` | `let name = [*Typst!*]` |

Una vez que entraste al modo código con #, no necesitás usar más numerales, a menos que hayas vuelto al modo marcado o matemático en el medio.

### 3.1.2 Marcado

Typst ofrece marcado incorporado para los elementos de documento más comunes. La mayoría de los elementos de sintaxis son simples atajos de la función correspondiente. La tabla de abajo enumera todo el marcado disponible y enlaza al mejor lugar para aprender más sobre su sintaxis y uso.

| Nombre | Ejemplo | Ver |
|---|---|---|
| Salto de párrafo | Línea en blanco | parbreak[3.6.15] |
| Énfasis fuerte | `*strong*` | strong[3.6.18] |
| Énfasis | `_emphasis_` | emph[3.6.6] |
| Texto sin formato (raw) | `` `print(1)` `` | raw[3.7.6] |
| Enlace | `https://typst.app/` | link[3.6.10] |
| Etiqueta | `<intro>` | label[3.5.18] |
| Referencia | `@intro` | ref[3.6.17] |
| Título | `= Heading` | heading[3.6.9] |
| Lista con viñetas | `- item` | list[3.6.2] |
| Lista numerada | `+ item` | enum[3.6.11] |
| Lista de términos | `/ Term: description` | terms[3.6.20] |
| Matemática | `$x^2$` | Matemática[3.8] |
| Salto de línea | `\` | linebreak[3.7.2] |
| Comillas inteligentes | `'single' or "double"` | smartquote[3.7.8] |
| Atajo de símbolo | `~`, `---` | Símbolos[3.9.1] |
| Expresión de código | `#rect(width: 1cm)` | Scripting[3.3.1] |
| Escape de carácter | `Tweet at us \#ad` | Más abajo[3.1.6] |
| Comentario | `/* block */`, `// line` | Más abajo[3.1.5] |

### 3.1.3 Modo matemático

El modo matemático es un modo de marcado especial que se usa para componer fórmulas matemáticas. Se activa envolviendo una ecuación entre caracteres $. Funciona tanto en marcado como en código. La ecuación

se compone en su propio bloque si empieza y termina con al menos un espacio (por ejemplo, `$ x^2 $`). Se puede producir matemática en línea omitiendo los espacios (por ejemplo, `$x^2$`). A continuación, una descripción general de la sintaxis específica del modo matemático:

| Nombre | Ejemplo | Ver |
|---|---|---|
| Matemática en línea | `$x^2$` | Matemática[3.8] |
| Matemática en bloque | `$ x^2 $` | Matemática[3.8] |
| Adjunto inferior | `$x_1$` | attach[3.8.10] |
| Adjunto superior | `$x^2$` | attach[3.8.10] |
| Fracción | `$1 + (a+b)/5$` | frac[3.8.19] |
| Salto de línea | `$x \ y$` | linebreak[3.7.2] |
| Punto de alineación | `$x &= 2 \ &= 3$` | Matemática[3.8] |
| Acceso a variable | `$#x$, $pi$` | Matemática[3.8] |
| Acceso a campo | `$arrow.r.long$` | Scripting[3.3.6] |
| Multiplicación implícita | `$x y$` | Matemática[3.8] |
| Atajo de símbolo | `$->$, $!=$` | Símbolos[3.9.1] |
| Texto/cadena en matemática | `$a "is natural"$` | Matemática[3.8] |
| Llamada a función matemática | `$floor(x)$` | Matemática[3.8] |
| Expresión de código | `$#rect(width: 1cm)$` | Scripting[3.3.1] |
| Escape de carácter | `$x\^2$` | Más abajo[3.1.6] |
| Comentario | `$/* comment */$` | Más abajo[3.1.5] |

### 3.1.4 Modo código

Dentro de los bloques y las expresiones de código, se pueden iniciar expresiones nuevas sin un carácter # inicial. Muchos elementos sintácticos son específicos de las expresiones. A continuación, una tabla con toda la sintaxis disponible en el modo código:

| Nombre | Ejemplo | Ver |
|---|---|---|
| Ninguno | `none` | none[3.5.20] |
| Auto | `auto` | auto[3.5.4] |
| Booleano | `false`, `true` | bool[3.5.6] |
| Entero | `10`, `0xff` | int[3.5.17] |
| Número de punto flotante | `3.14`, `1e5` | float[3.5.15] |
| Longitud | `2pt`, `3mm`, `1em`, .. | length[3.10.13] |
| Ángulo | `90deg`, `1rad` | angle[3.10.3] |
| Fracción | `2fr` | fraction[3.10.9] |
| Proporción | `50%` | ratio[3.10.20] |
| Cadena de texto | `"hello"` | str[3.5.28] |
| Etiqueta | `<intro>` | label[3.5.18] |
| Matemática | `$x^2$` | Matemática[3.8] |
| Texto sin formato (raw) | `` `print(1)` `` | raw[3.7.6] |
| Acceso a variable | `x` | Scripting[3.3.2] |
| Bloque de código | `{ let x = 1; x + 2 }` | Scripting[3.3.2] |
| Bloque de contenido | `[*Hello*]` | Scripting[3.3.2] |
| Expresión entre paréntesis | `(1 + 2)` | Scripting[3.3.2] |
| Array | `(1, 2, 3)` | Array[3.5.2] |
| Diccionario | `(a: "hi", b: 2)` | Diccionario[3.5.12] |
| Operador unario | `-x` | Scripting[3.3.10] |

| Nombre | Ejemplo | Ver |
|---|---|---|
| Operador binario | `x + y` | Scripting[3.3.10] |
| Asignación | `x = 1` | Scripting[3.3.10] |
| Acceso a campo | `x.y` | Scripting[3.3.6] |
| Llamada a método | `x.flatten()` | Scripting[3.3.7] |
| Llamada a función | `min(x, y)` | Función[3.5.16] |
| Expansión de argumentos | `min(..nums)` | Argumentos[3.5.1] |
| Función sin nombre | `(x, y) => x + y` | Función[3.5.16/6] |
| Enlace let | `let x = 1` | Scripting[3.3.3] |
| Función con nombre | `let f(x) = 2 * x` | Función[3.5.16] |
| Regla set | `set text(14pt)` | Estilos[3.2.1] |
| Regla set-if | `set text(..) if ..` | Estilos[3.2.1] |
| Regla show-set | `show heading: set block(..)` | Estilos[3.2.2] |
| Regla show con función | `show raw: it => {..}` | Estilos[3.2.2] |
| Regla show de todo | `show: template` | Estilos[3.2.2] |
| Expresión de contexto | `context text.lang` | Contexto[3.4] |
| Condicional | `if x == 1 {..} else {..}` | Scripting[3.3.4] |
| Bucle for | `for x in (1, 2, 3) {..}` | Scripting[3.3.5] |
| Bucle while | `while x < 10 {..}` | Scripting[3.3.5] |
| Control de flujo del bucle | `break, continue` | Scripting[3.3.5] |
| Retorno de una función | `return x` | Función[3.5.16] |
| Incluir un módulo | `include "bar.typ"` | Scripting[3.3.8] |
| Importar un módulo | `import "bar.typ"` | Scripting[3.3.8] |
| Importar elementos de un módulo | `import "bar.typ": a, b, c` | Scripting[3.3.8] |
| Comentario | `/* block */`, `// line` | Más abajo[3.1.5] |

### 3.1.5 Comentarios

Typst ignora los comentarios y no los incluye en la salida. Esto es útil para excluir versiones viejas o para agregar anotaciones. Para comentar una sola línea, empezala con `//`:

```
// our data barely supports
// this claim

We show with $p < 0.05$
that the difference is
significant.
```

We show with $p < 0.05$ that the difference is significant.

Los comentarios también se pueden envolver entre `/*` y `*/`. En ese caso, el comentario puede abarcar varias líneas:

```
Our study design is as follows:
/* Somebody write this up:
   - 1000 participants.
   - 2x2 data design. */
```

Our study design is as follows:

### 3.1.6 Secuencias de escape

Las secuencias de escape se usan para insertar caracteres especiales que son difíciles de escribir o que tienen un significado especial en Typst. Para escapar un carácter, anteponele una barra invertida. Para insertar

cualquier punto de código Unicode, podés escribir una secuencia de escape hexadecimal: `\u{1f600}`. El mismo tipo de secuencias de escape también funciona en las cadenas de texto[3.5.28].

```
I got an ice cream for
\$1.50! \u{1f600}
```

I got an ice cream for $1.50! 😀

### 3.1.7 Identifiers

Los nombres de variables, funciones, etcétera (*identificadores*) pueden contener letras, números, guiones (-) y guiones bajos (_). Tienen que empezar con una letra o un guion bajo.

Más específicamente, la sintaxis de los identificadores en Typst se basa en el Anexo N° 31 del Estándar Unicode[A], con dos extensiones: permitir _ como carácter inicial y permitir tanto _ como - como caracteres de continuación.

Para los identificadores de varias palabras, la convención de mayúsculas y minúsculas recomendada es el kebab case[B]. En kebab case, las palabras se escriben en minúsculas y separadas por guiones (como en `top-edge`). Esto es especialmente relevante cuando desarrollás módulos y paquetes para que otros los usen, ya que mantiene las cosas predecibles.

```
#let kebab-case = [Using hyphen]
#let _schön = "😊"
#let 始料不及 = "😱"
#let π = calc.pi

#kebab-case
#if -π < 0 { _schön } else { 始料不及 }
// -π means -1 * π,
// so it's not a valid identifier
```

Using hyphen 😊

[A] https://www.unicode.org/reports/tr31/
[B] https://en.wikipedia.org/wiki/Letter_case#Kebab_case

## 3.2 Estilos

Typst incluye un sistema de estilos flexible que aplica automáticamente a tu documento los estilos que elijas. Con las *reglas set*, podés configurar las propiedades básicas de los elementos. De esta manera creás la mayoría de los estilos comunes. Sin embargo, puede que no haya una propiedad incorporada para todo lo que querés hacer. Por eso, Typst también admite las *reglas show*, que pueden redefinir por completo el aspecto de los elementos.

### 3.2.1 Reglas set

Con las reglas set, podés personalizar el aspecto de los elementos. Se escriben como una llamada a función[3.5.16] de una función de elemento[3.5.16/2] precedida de la palabra clave `set` (o `#set` en marcado). A la regla set solo se le pueden proporcionar los parámetros opcionales de esa función. Consultá la documentación de cada función para ver cuáles parámetros son opcionales. En el siguiente ejemplo, usamos dos reglas set para cambiar la familia de fuente[3.7.12/2.1] y la numeración de los títulos[3.6.9/5.4].

```
#set heading(numbering: "I.")
#set text(
  font: "New Computer Modern"
)

= Introduction
With set rules, you can style
your document.
```

I. Introduction

With set rules, you can style your document.

Una regla set de nivel superior sigue vigente hasta el final del archivo. Cuando está anidada dentro de un bloque de código o de contenido, solo está vigente hasta el final de ese bloque. Con un bloque, podés entonces restringir el efecto de una regla a un segmento particular de tu documento. A continuación, usamos un bloque de contenido para limitar el estilo de lista a una lista en particular.

```
This list is affected: #[
  #set list(marker: [--])
  - Dash
]

This one is not:
- Bullet
```

This list is affected:
– Dash

This one is not:
• Bullet

A veces vas a querer aplicar una regla set de forma condicional. Para eso, podés usar una regla *set-if*.

```
#let task(body, critical: false) = {
  set text(red) if critical
  [- #body]
}

#task(critical: true)[Food today?]
#task(critical: false)[Work deadline]
```

- Food today?
- Work deadline

### 3.2.2 Reglas show

Con las reglas show, podés personalizar en profundidad el aspecto de un tipo de elemento. La forma más básica de regla show es una *regla show-set*. Una regla así se escribe con la palabra clave `show` seguida de un selector[3.5.26], dos puntos y luego una regla set. La forma más básica de selector es una función de elemento[3.5.16/2]. Esto hace que la regla set se aplique solo al elemento seleccionado. En el siguiente ejemplo, los títulos se vuelven azul oscuro mientras que todo el resto del texto sigue en negro.

```
#show heading: set text(navy)

= This is navy-blue
But this stays black.
```

**This is navy-blue**
But this stays black.

Con las reglas show-set podés combinar propiedades de distintas funciones para lograr muchos efectos diferentes. Pero igual te limitan a lo que está predefinido en Typst. Para lograr la máxima flexibilidad, podés escribir en cambio una regla show *de transformación* que defina cómo dar formato a un elemento desde cero. Para escribir una regla show así, reemplazá la regla set que va después de los dos puntos por una función[3.5.16] arbitraria. Esta función recibe el elemento en cuestión y puede devolver contenido arbitrario. La función suele definirse en línea como `it => ..` usando la sintaxis de función sin nombre[3.5.16/6]. Por convención, el parámetro de la función suele llamarse `it`.

Los campos[3.3.6] disponibles en el elemento que se le pasa a la función coinciden con los parámetros de la función de elemento respectiva. A continuación, definimos una regla show que da formato a los títulos de una enciclopedia de fantasía.

La regla show en sí agrega caracteres de tilde alrededor del título (hay que escaparlos con una barra invertida porque, de lo contrario, indicarían un espacio de no separación), resalta el título en cursiva y después muestra el contador de títulos a continuación del título.

Para este ejemplo, también queríamos alineación centrada y una fuente diferente. Si bien podríamos haber agregado estas reglas set dentro de la regla show existente, las agregamos como reglas show-set separadas. Esta es una buena práctica porque así estas reglas todavía pueden ser sobrescritas por reglas show-set posteriores del documento, lo que mantiene los estilos componibles. En cambio, las reglas set dentro de una regla show de transformación ya no se podrían sobrescribir.

```
#set heading(numbering: "(I)")
#show heading: set align(center)
#show heading: set text(font: "Inria Serif")
#show heading: it => block[
  \~
  #emph(it.body)
  #counter(heading).display()
  \~
]

= Dragon
With a base health of 15, the dragon is the most
powerful creature.

= Manticore
While less powerful than the dragon, the manticore
gets extra style points.
```

~ *Dragon* (I) ~

With a base health of 15, the dragon is the most powerful creature.

~ *Manticore* (II) ~

While less powerful than the dragon, the manticore gets extra style points.

Al igual que las reglas set, las reglas show están vigentes hasta el final del bloque o archivo actual.

En lugar de una función, el lado derecho de una regla show también puede tomar una cadena de texto literal o un bloque de contenido que se sustituya directamente por el elemento. Y además de una función, el lado izquierdo de una regla show también puede tomar otros *selectores* que definen a qué aplicar la transformación:

- **Todo:** `show: rest => ..`
  Transforma todo lo que sigue a la regla show. Es útil para aplicar un diseño más complejo a todo tu documento sin envolverlo todo en una llamada a una función gigante.
- **Texto:** `show "Text": ..`
  Da estilo, transforma o reemplaza texto.
- **Regex:** `show regex("\w+"): ..`
  Selecciona y transforma texto con una expresión regular para tener aún más flexibilidad. Mirá la documentación del tipo `regex`[3.5.24] para más detalles.
- **Función con campos:** `show heading.where(level: 1): ..`
  Transforma solo los elementos que tienen los campos especificados. Por ejemplo, podrías querer cambiar solo el estilo de los títulos de nivel 1.
- **Etiqueta:** `show <intro>: ..`
  Selecciona y transforma los elementos que tienen la etiqueta especificada. Mirá la documentación del tipo `label`[3.5.18] para más detalles.

```
#show "Project": smallcaps
#show "badly": "great"

We started Project in 2019
and are still working on it.
Project is progressing badly.
```

We started Project in 2019 and are still working on it. Project is progressing great.

## 3.3 Scripting

Typst incorpora un potente lenguaje de scripting. Podés automatizar tus documentos y crear estilos más sofisticados con código. A continuación, una descripción general de los conceptos de scripting.

### 3.3.1 Expresiones

En Typst, el marcado y el código están fusionados. Todos los elementos, salvo los más comunes, se crean con *funciones*. Para que esto sea lo más cómodo posible, Typst ofrece una sintaxis compacta para incrustar una expresión de código en el marcado: una expresión se introduce con un numeral (#) y el análisis normal del marcado se retoma cuando la expresión termina. Si un carácter continuaría la expresión pero debe interpretarse como texto, se puede terminar la expresión a la fuerza con un punto y coma (;). Podés escapar un # o un ; literal con una barra invertida[3.1.6].

```
#emph[Hello] \
#emoji.face \
#"hello".len()
```

*Hello*
😀
5

El ejemplo anterior muestra algunas de las expresiones disponibles, incluidas las llamadas a funciones[3.5.16], los accesos a campos[3.3.6] y las llamadas a métodos[3.3.7]. En el resto de este capítulo se tratan más tipos de expresiones. Algunos tipos de expresiones no son compatibles con la sintaxis del numeral (por ejemplo, las expresiones con operadores binarios). Para incrustarlas en el marcado, podés usar paréntesis, como en `#(1 + 2)`.

### 3.3.2 Bloques

Para estructurar tu código e incrustarle marcado, Typst ofrece dos tipos de *bloques*:

- **Bloque de código:** `{ let x = 1; x + 2 }`
  Cuando escribís código, probablemente quieras dividir tu cómputo en varias sentencias, crear algunas variables intermedias, etcétera. Los bloques de código te permiten escribir varias expresiones donde se espera una. Las expresiones individuales de un bloque de código deben separarse con saltos de línea o punto y coma. Los valores de salida de las expresiones individuales de un bloque de código se unen para determinar el valor del bloque. Las expresiones sin salida útil, como los enlaces `let`, devuelven `none`, que se puede unir a cualquier valor sin efecto.
- **Bloque de contenido:** `[*Hey* there!]`
  Con los bloques de contenido, podés manejar marcado/contenido como un valor programático, guardarlo en variables y pasarlo a funciones[3.5.16]. Los bloques de contenido están delimitados por corchetes y pueden contener marcado arbitrario. Un bloque de contenido da como resultado un valor de tipo contenido[3.5.8]. Se puede pasar una cantidad arbitraria de bloques de contenido como argumentos finales de funciones. Es decir, `list([A], [B])` equivale a `list[A][B]`.

Los bloques de contenido y de código se pueden anidar arbitrariamente. En el siguiente ejemplo, `[hello ]` se une con la salida de `a + [ the ] + b`, lo que da `[hello from the *world*]`.

```
#{
  let a = [from]
  let b = [*world*]
  [hello ]
  a + [ the ] + b
}
```

hello from the **world**

### 3.3.3 Enlaces y desestructuración

Como ya se demostró arriba, las variables se pueden definir con enlaces `let`. A la variable se le asigna el valor de la expresión que sigue al signo =. Un nombre de variable válido[3.1.7] puede contener –, pero no puede empezar con –. La asignación de un valor es opcional: si no se asigna ningún valor, la variable se inicializa como `none`. La palabra clave `let` también se puede usar para crear una función personalizada con nombre[3.5.16/4]. Se puede acceder a las variables durante el resto del bloque que las contiene (o durante el resto del archivo, si no hay un bloque que las contenga).

```
#let name = "Typst"
This is #name's documentation.
It explains #name.

#let my-add(x, y) = x + y
Sum is #my-add(2, 3).
```

This is Typst's documentation. It explains Typst.

Sum is 5.

Los enlaces let también se pueden usar para desestructurar arrays[3.5.2] y diccionarios[3.5.12]. En ese caso, la estructura del lado izquierdo de la asignación debe reflejar el array o el diccionario: con enlaces que se corresponden por posición en los arrays y por nombre de clave en los diccionarios. El operador `..` se puede usar una vez en el patrón para recolectar el resto de los elementos del array o del diccionario.

```
#let (x, y) = (1, 2)
The coordinates are #x, #y.

#let (a, .., b) = (1, 2, 3, 4)
The first element is #a.
The last element is #b.

#let books = (
  Shakespeare: "Hamlet",
  Homer: "The Odyssey",
  Austen: "Persuasion",
)

#let (Austen,) = books
Austen wrote #Austen.

#let (Homer: h) = books
Homer wrote #h.

#let (Homer, ..other) = books
#for (author, title) in other [
  #author wrote #title.
]
```

The coordinates are 1, 2.

The first element is 1. The last element is 4.

Austen wrote Persuasion.

Homer wrote The Odyssey.

Shakespeare wrote Hamlet. Austen wrote Persuasion.

Podés usar el guion bajo para descartar elementos en un patrón de desestructuración:

```
#let (_, y, _) = (1, 2, 3)
The y coordinate is #y.
```

The y coordinate is 2.

La desestructuración también funciona en las listas de argumentos de funciones …

```
#let left = (2, 4, 5)
#let right = (3, 2, 6)
#left.zip(right).map(
  ((a,b)) => a + b
)
```

```
(5, 6, 11)
```

... y en el lado izquierdo de las asignaciones normales. Esto puede ser útil, entre otras cosas, para intercambiar variables.

```
#{
  let a = 1
  let b = 2
  (a, b) = (b, a)
  [a = #a, b = #b]
}
```

a = 2, b = 1

### 3.3.4 Condicionales

Con un condicional, podés mostrar o calcular cosas distintas según se cumpla o no una condición. Typst admite las expresiones `if`, `else if` y `else`. Cuando la condición se evalúa como `true`, el condicional devuelve el valor que resulta del cuerpo del if. De lo contrario, devuelve el valor que resulta del cuerpo del else.

```
#if 1 < 2 [
  This is shown
] else [
  This is not.
]
```

This is shown

Cada rama puede tener como cuerpo un bloque de código o de contenido.

- `if condition {..}`
- `if condition [..]`
- `if condition [..] else {..}`
- `if condition [..] else if condition {..} else [..]`

### 3.3.5 Bucles

Con los bucles, podés repetir contenido o calcular algo de forma iterativa. Typst admite dos tipos de bucles: `for` y `while`. Los primeros iteran sobre una colección especificada, mientras que los segundos iteran mientras se siga cumpliendo una condición. Igual que los bloques, los bucles *unen* los resultados de cada iteración en un solo valor.

En el siguiente ejemplo, las tres oraciones creadas por el bucle for se unen en un único valor de contenido y los arrays de longitud 1 del bucle while se unen en un array más grande.

```
#for c in "ABC" [
  #c is a letter.
]

#let n = 2
#while n < 10 {
  n = (n * 2) - 1
  (n,)
}
```

A is a letter. B is a letter. C is a letter.

`(3, 5, 9, 17)`

Los bucles for pueden iterar sobre una variedad de colecciones:

- `for value in array {..}`
  Itera sobre los elementos del array[3.5.2]. Acá también se puede usar la sintaxis de desestructuración descrita en Enlace let[3.3.3].
- `for pair in dict {..}`
  Itera sobre los pares clave-valor del diccionario[3.5.12]. Los pares también se pueden desestructurar usando `for (key, value) in dict {..}`. Es más eficiente que `for pair in dict.pairs() {..}` porque no crea un array temporal con todos los pares clave-valor.

- `for letter in "abc" {..}`
  Itera sobre los caracteres de la cadena de texto[3.5.28]. Técnicamente, itera sobre los grupos de grafemas de la cadena. La mayoría de las veces, un grupo de grafemas es un solo punto de código. Sin embargo, un grupo de grafemas puede contener varios puntos de código, como un emoji de bandera.
- `for byte in bytes("😀") {..}`
  Itera sobre los bytes[3.5.7], que se pueden convertir a partir de una cadena de texto[3.5.28] o leer[3.13.5] de un archivo sin codificación. Cada valor de byte es un entero[3.5.17] entre `0` y `255`.

Para controlar la ejecución del bucle, Typst ofrece las sentencias `break` y `continue`. La primera sale anticipadamente del bucle, mientras que la segunda salta a la siguiente iteración.

```
#for letter in "abc nope" {
  if letter == " " {
    break
  }

  letter
}
```

abc

El cuerpo de un bucle puede ser un bloque de código o de contenido:

- `for .. in collection {..}`
- `for .. in collection [..]`
- `while condition {..}`
- `while condition [..]`

### 3.3.6 Campos

Podés usar la *notación con punto* para acceder a los campos de un valor. Para los valores de tipo content[3.5.8], también podés usar la función `fields`[3.5.8/3.4] para listar los campos.

El valor en cuestión puede ser:
- un diccionario[3.5.12] que tiene la clave especificada,
- un símbolo[3.5.29] que tiene el modificador especificado,
- un módulo[3.5.19] que contiene la definición especificada,
- contenido[3.5.8] que consiste en un elemento que tiene el campo especificado. Los campos disponibles coinciden con los argumentos de la función de elemento[3.5.16/2] que se dieron cuando se construyó el elemento.

```
#let it = [= Heading]
#it.body \
#it.depth \
#it.fields()

#let dict = (greet: "Hello")
#dict.greet \
#emoji.face
```

Heading
1
`(depth: 1, body: [Heading])`
Hello
😀

### 3.3.7 Métodos

Una *llamada a método* es una forma cómoda de llamar a una función que pertenece al ámbito del tipo[3.5.31] de un valor. Por ejemplo, podemos llamar a la función `str.len`[3.5.28/4.1] de las dos maneras equivalentes siguientes:

```
#str.len("abc") is the same as
#"abc".len()
```

3 is the same as 3

La estructura de una llamada a método es `value.method(..args)` y su llamada a función completa equivalente es `type(value).method(value, ..args)`. La documentación de cada tipo enumera las funciones de su ámbito. Actualmente no podés definir tus propios métodos.

```
#let values = (1, 2, 3, 4)
#values.pop() \
#values.len() \

#("a, b, c"
    .split(", ")
    .join[ --- ])

#"abc".len() is the same as
#str.len("abc")
```

4
3

a — b — c

3 is the same as 3

Hay algunas funciones especiales que modifican el valor sobre el que se las llama (por ejemplo, `array.push`[3.5.2/3.5]). Estas funciones *deben* llamarse en forma de método. En algunos casos, cuando el método se llama solo por su efecto secundario, su valor de retorno se debe ignorar (y no participar en la unión). La forma canónica de descartar un valor es con un enlace let: `let _ = array.remove(1)`.

### 3.3.8 Módulos

Podés dividir tus proyectos de Typst en varios archivos llamados *módulos*. Un módulo puede referirse al contenido y a las definiciones de otro módulo de varias maneras:

- **Inclusión:** `include "bar.typ"`
  Evalúa el archivo en la ruta[3.5.22] `bar.typ` y devuelve el contenido[3.5.8] resultante.
- **Importación:** `import "bar.typ"`
  Evalúa el archivo en la ruta[3.5.22] `bar.typ` e inserta el módulo[3.5.19] resultante en el ámbito actual como `bar` (nombre de archivo sin extensión). Podés usar la palabra clave `as` para renombrar el módulo importado: `import "bar.typ" as baz`. Podés importar elementos anidados con la notación con punto: `import "bar.typ": baz.a`.
- **Importación de elementos:** `import "bar.typ": a, b`
  Evalúa el archivo en la ruta[3.5.22] `bar.typ`, extrae los valores de las variables `a` y `b` (que tienen que estar definidas en `bar.typ`, por ejemplo mediante enlaces `let`) y los define en el archivo actual. Reemplazar `a, b` por `*` carga todas las variables definidas en un módulo. Podés usar la palabra clave `as` para renombrar cada elemento: `import "bar.typ": a as one, b as two`

En lugar de una cadena de texto o una ruta[3.5.22], también podés usar un valor de módulo[3.5.19], como se muestra en el siguiente ejemplo:

```
#import emoji: face
#face.grin
```

😀

### 3.3.9 Paquetes

Para reutilizar componentes entre proyectos, también podés crear e importar *paquetes* de Typst. La importación de un paquete se especifica como una tripla formada por un espacio de nombres, un nombre y una versión.

```
#import "@preview/example:0.1.0": add
#add(2, 7)
```

9

El espacio de nombres `preview` contiene paquetes compartidos por la comunidad. Podés encontrar todos los

paquetes de la comunidad disponibles en Typst Universe[A].

Si usás Typst de forma local, también podés crear tus propios paquetes locales del sistema. Para más detalles, mirá el repositorio de paquetes[B].

### 3.3.10 Operadores

La siguiente tabla enumera todos los operadores unarios y binarios disponibles, con su efecto, su aridad (unario, binario) y su nivel de precedencia (los más altos se asocian con más fuerza). Algunas operaciones, como el módulo[3.5.9/2.39], no tienen una sintaxis especial y se pueden lograr con funciones del módulo `calc`[3.5.9].

| Operador | Efecto | Aridad | Precedencia |
|---|---|---|---|
| `-` | Negación | Unario | 7 |
| `+` | Sin efecto (existe por simetría) | Unario | 7 |
| `*` | Multiplicación | Binario | 6 |
| `/` | División | Binario | 6 |
| `+` | Suma | Binario | 5 |
| `-` | Resta | Binario | 5 |
| `==` | Verifica igualdad | Binario | 4 |
| `!=` | Verifica desigualdad | Binario | 4 |
| `<` | Verifica menor que | Binario | 4 |
| `<=` | Verifica menor o igual que | Binario | 4 |
| `>` | Verifica mayor que | Binario | 4 |
| `>=` | Verifica mayor o igual que | Binario | 4 |
| `in` | Verifica si está en una colección | Binario | 4 |
| `not in` | Verifica si no está en una colección | Binario | 4 |
| `not` | “No” lógico | Unario | 3 |
| `and` | “Y” lógico con cortocircuito | Binario | 3 |
| `or` | “O” lógico con cortocircuito | Binario | 2 |
| `=` | Asignación | Binario | 1 |
| `+=` | Asignación con suma | Binario | 1 |
| `-=` | Asignación con resta | Binario | 1 |
| `*=` | Asignación con multiplicación | Binario | 1 |
| `/=` | Asignación con división | Binario | 1 |

[A] https://typst.app/universe
[B] https://github.com/typst/packages

## 3.4 Contexto

A veces queremos crear contenido que reaccione a su ubicación en el documento. Puede ser una frase localizada que dependa del idioma de texto configurado o algo tan simple como el número de un título que imprime el valor correcto según cuántos títulos vinieron antes. Sin embargo, el código de Typst no es directamente consciente de su ubicación en el documento. Un código al principio del texto fuente podría producir contenido que termine al final del documento.

Para producir contenido que reaccione a su entorno, tenemos entonces que indicárselo específicamente a Typst. Lo hacemos con la palabra clave `context`, que precede a una expresión y asegura que se calcule con conocimiento de su entorno. A cambio, la expresión de contexto en sí queda opaca. No podemos acceder directamente en nuestro código a lo que resulta de ella, precisamente porque es contextual: no hay un único resultado correcto, puede haber varios resultados en distintos lugares del documento. Por eso, todo lo que dependa de datos contextuales tiene que ocurrir dentro de la expresión de contexto.

Además de las expresiones de contexto explícitas, el contexto también se establece de forma implícita en algunos lugares que también son conscientes de su ubicación en el documento: las reglas show[3.2.2] proporcionan contexto[A] y, por ejemplo, las numeraciones del índice también proporcionan el contexto adecuado para resolver contadores.

### 3.4.1 Contexto de estilo

Con las reglas set, podemos ajustar las propiedades de estilo de partes de nuestro documento o de todo él. No podemos acceder a ellas sin un contexto conocido, ya que pueden cambiar a lo largo del documento. Cuando hay un contexto disponible, podemos recuperarlas simplemente accediendo a ellas como campos de la función de elemento respectiva.

```
#set text(lang: "de")
#context text.lang
```

de

Como se explicó antes, una expresión de contexto reacciona a los distintos entornos en los que se la coloca. En el siguiente ejemplo, creamos una sola expresión de contexto, la guardamos en la variable `value` y la usamos varias veces. Cada uso reacciona correctamente al entorno actual.

```
#let value = context text.lang
#value

#set text(lang: "de")
#value

#set text(lang: "fr")
#value
```

es

de

fr

Un punto clave: al crearse, `value` se convierte en contenido[3.5.8] opaco al que no podemos echarle un vistazo. Solo se puede resolver cuando se lo coloca en algún lugar, porque solo entonces se conoce el contexto. El cuerpo de una expresión de contexto puede evaluarse cero, una o varias veces, según en cuántos lugares distintos se lo ponga.

### 3.4.2 Contexto de ubicación

Ya vimos que el contexto nos da acceso a los valores de las reglas set. Pero puede hacer más: también nos permite saber *dónde* del documento estamos, en relación con otros elementos y de forma absoluta en las páginas. Podemos usar esta información para crear interacciones muy flexibles entre distintas partes del documento. Esto es lo que sustenta funciones como la numeración de títulos, el índice o los encabezados de página que dependen de los títulos de sección.

[A] Actualmente, todas las reglas show proporcionan un contexto de estilo, pero solo las reglas show sobre elementos localizables proporcionan un contexto de ubicación.

Algunas funciones, como `counter.get`[3.12.1/11.1], acceden implícitamente a la ubicación actual. En el siguiente ejemplo, queremos recuperar el valor del contador de títulos. Como cambia a lo largo del documento, primero tenemos que entrar en una expresión de contexto. Después, usamos `get` para recuperar el valor actual del contador. Esta función accede a la ubicación actual desde el contexto para resolver el valor del contador. Los contadores tienen varios niveles y `get` devuelve un array con los números resueltos. Así obtenemos el siguiente resultado:

```
#set heading(numbering: "1.")

= Introduction
#lorem(5)

#context counter(heading).get()

= Background
#lorem(5)

#context counter(heading).get()
```




Para tener más flexibilidad, también podemos usar la función `here`[3.12.2] para extraer directamente la ubicación[3.12.4] actual del contexto. El siguiente ejemplo lo demuestra:

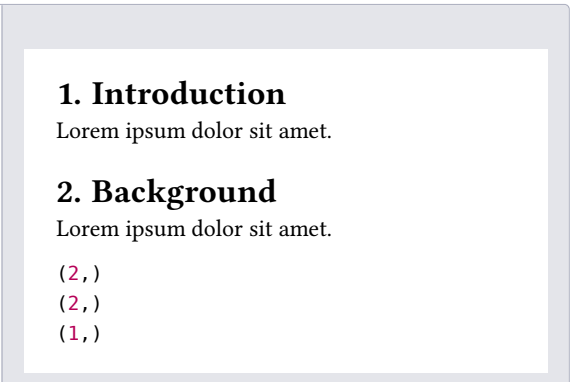
- Primero tenemos `counter(heading).get()`, que se resuelve en `(2,)` como antes.
- Después usamos la más potente `counter.at`[3.12.1/11.3] con `here`[3.12.2], que en combinación equivale a `get`, y así obtenemos `(2,)`.
- Por último, usamos `at` con una etiqueta[3.5.18] para recuperar el valor del contador en una ubicación *diferente* del documento, en nuestro caso la del título de la introducción. Esto da `(1,)`. El sistema de contexto de Typst nos da capacidades de viaje en el tiempo y nos permite recuperar los valores de cualquier contador y estado en *cualquier* ubicación del documento.

```
#set heading(numbering: "1.")

= Introduction <intro>
#lorem(5)

= Background <back>
#lorem(5)

#context [
  #counter(heading).get() \
  #counter(heading).at(here()) \
  #counter(heading).at(<intro>)
]
```

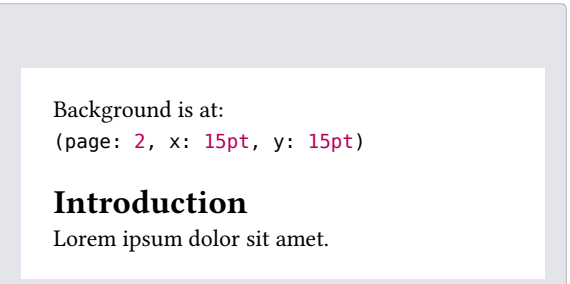


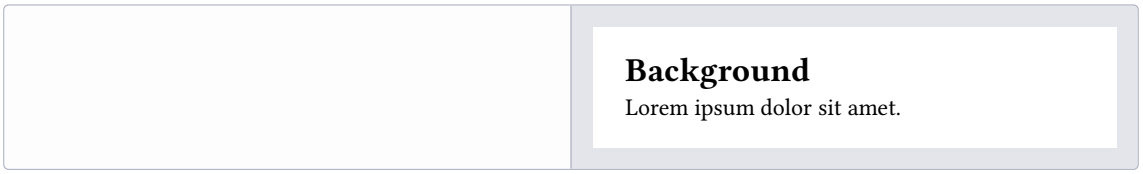

Como mencionamos antes, también podemos usar context para obtener la posición física de los elementos en las páginas. Lo hacemos con la función `locate`[3.12.3], que funciona de forma similar a `counter.at`: toma una ubicación u otro selector[3.5.26] que se resuelva en un elemento único (también puede ser una etiqueta) y devuelve la posición de ese elemento en las páginas.

```
Background is at: \
#context locate(<back>).position()

= Introduction <intro>
#lorem(5)
#pagebreak()

= Background <back>
#lorem(5)
```






Hay otras funciones que usan el contexto de ubicación, la más destacada es query[3.12.6]. Mirá la categoría de introspección[3.12] para más detalles sobre ellas.

### 3.4.3 Contextos anidados

El contexto también es accesible desde dentro de llamadas a funciones anidadas en bloques de contexto. En el siguiente ejemplo, `foo` en sí se convierte en una función contextual, igual que lo es to-absolute[3.10.13/3.5].

```
#let foo() = 1em.to-absolute()
#context {
  foo() == text.size
}
```

```
true
```

Los bloques de contexto se pueden anidar. El código contextual siempre accede entonces al contexto más interno. El siguiente ejemplo lo demuestra: el primer `text.lang` accede a los estilos del bloque de contexto externo y, por lo tanto, **no** ve el efecto de `set text(lang: "fr")`. El bloque de contexto anidado alrededor del segundo `text.lang`, en cambio, empieza después de la regla set y por eso muestra su efecto.

```
#set text(lang: "de")
#context [
  #set text(lang: "fr")
  #text.lang \
  #context text.lang
]
```

```
de
fr
```

Quizás te preguntes por qué Typst ignora la regla set del francés al calcular el primer `text.lang` del ejemplo anterior. La razón es que, en el caso general, Typst no puede conocer todos los estilos que se van a aplicar, ya que las reglas set se pueden aplicar al contenido después de haberlo construido. A continuación, `text.lang` ya está calculado cuando se aplica la función de plantilla. Por lo tanto, no puede ser consciente del cambio de idioma a francés en la plantilla.

```
#let template(body) = {
  set text(lang: "fr")
  upper(body)
}

#set text(lang: "de")
#context [
  #show: template
  #text.lang \
  #context text.lang
]
```

```
DE
FR
```

El segundo `text.lang`, en cambio, *sí* reacciona al cambio de idioma, porque la evaluación de su bloque de contexto circundante se difiere hasta que se conocen los estilos que le corresponden. Esto ilustra la importancia de elegir el punto de inserción correcto para un contexto, de modo de acceder precisamente a los estilos correctos.

Lo mismo vale para el contexto de ubicación. A continuación, la primera llamada a `c.display()` accede al bloque de contexto externo y, por lo tanto, no ve el efecto de `c.update(2)`, mientras que la segunda `c.display()` accede al contexto interno y, por lo tanto, sí lo ve.

```
#let c = counter("mycounter")
#c.update(1)
#context [
  #c.update(2)
  #c.display() \
  #context c.display()
]
```

1
2

### 3.4.4 Iteraciones del compilador

Para resolver las interacciones contextuales, el compilador de Typst procesa tu documento varias veces. Por ejemplo, para resolver una llamada a `locate`, Typst primero proporciona una posición provisoria, compone tu documento y luego vuelve a compilar con la posición conocida a partir del diseño terminado. Se usa el mismo enfoque para resolver contadores, estados y consultas. En ciertos casos, Typst puede incluso necesitar más de dos iteraciones para resolverlo todo. Si bien a veces es una necesidad, también puede ser señal de un uso indebido de las funciones contextuales (por ejemplo, del estado[3.12.7/4]). Si Typst no puede resolver todo en cinco intentos, se detiene y muestra la advertencia "document did not converge within five attempts."

Un lector muy atento podría haber notado que no todas las funciones presentadas arriba usan realmente la ubicación actual. Si bien `counter(heading).get()` definitivamente depende de ella, `counter(heading).at(<intro>)`, por ejemplo, no. Sin embargo, igual requiere contexto. Aunque su valor es siempre el mismo *dentro* de una iteración de compilación, puede cambiar a lo largo de varias iteraciones del compilador. Si se la pudiera llamar directamente en el nivel superior de un módulo, todo el módulo y sus exportaciones podrían cambiar a lo largo de varias iteraciones del compilador, lo cual no sería deseable.

# 3.5 Fundamentos

Tipos y funciones fundamentales.

Acá vas a encontrar documentación de los tipos de datos básicos, como los enteros[3.5.17] y las cadenas de texto[3.5.28], así como detalles sobre las funciones de cálculo centrales.

## Definiciones


3.5.1 Arguments Captured arguments to a function. . . . 58
3.5.2 Array A sequence of values. . . . 61
3.5.3 Assert Ensures that a condition is fulfilled. . . . 71
3.5.4 Auto A value that indicates a smart default. . . . 73
3.5.5 Biblioteca estándar Un módulo que contiene todos los elementos accesibles globalmente. . . . 74
3.5.6 Boolean A type with two states. . . . 75
3.5.7 Bytes A sequence of bytes. . . . 76
3.5.8 Content A piece of document content. . . . 78
3.5.9 Cálculo Módulo para cálculos y procesamiento de valores numéricos. . . . 80
3.5.10 Datetime Represents a date, a time, or a combination of both. . . . 96
3.5.11 Decimal A fixed-point decimal number type. . . . 101
3.5.12 Dictionary A map from string keys to values. . . . 103
3.5.13 Duration Represents a positive or negative span of time. . . . 106
3.5.14 Evaluate Evaluates a string as Typst code. . . . 108
3.5.15 Float A floating-point number. . . . 109
3.5.16 Function A mapping from argument values to a return value. . . . 113
3.5.17 Integer An integer: a positive whole number, a negative whole number, or zero. . . . 116
3.5.18 Label A label for an element. . . . 122
3.5.19 Module A collection of variables and functions that are commonly related to a single theme. . . . 123
3.5.20 None A value that indicates the absence of any other value. . . . 124
3.5.21 Panic Fails with an error. . . . 125
3.5.22 Path A file system path. . . . 126
3.5.23 Plugin Loads a WebAssembly module. . . . 129
3.5.24 Regex A regular expression. . . . 132
3.5.25 Representation Returns the string representation of a value. . . . 134
3.5.26 Selector A filter for selecting elements within the document. . . . 135
3.5.27 Sistema Módulo para las interacciones con el sistema . . . . 138
3.5.28 String A sequence of Unicode codepoints. . . . 139
3.5.29 Symbol A Unicode symbol. . . . 147
3.5.30 Target Returns the current export target. . . . 148
3.5.31 Type Describes a kind of value. . . . 149
3.5.32 Version A version with an arbitrary number of components. . . . 151

### 3.5.1 `arguments` Desde: siempre

Argumentos capturados de una función.

Los argumentos son *posicionales* o *con nombre*, y se puede acceder a ellos mediante los métodos pos[3.5.1/4.3], named[3.5.1/4.4] y at[3.5.1/4.2].

Además, se puede acceder a los argumentos con nombre con la sintaxis de campos[3.5.1/4.2], de forma similar a los diccionarios[3.5.12].

#### /1 Sumideros de argumentos

Al igual que las funciones incorporadas, las funciones personalizadas también pueden tomar una cantidad variable de argumentos. Podés especificar un *sumidero de argumentos* que recolecta todos los argumentos sobrantes como `..sink`. El valor `sink` resultante es del tipo `arguments`. Expone métodos para acceder a los argumentos posicionales y con nombre.

```
#let format(title, ..authors) = {
  let by = authors
    .pos()
    .join(", ", last: " and ")

  [*#title* \ _Written by #by;_]
}

#format("ArtosFlow", "Jane", "Joe")
```

**ArtosFlow**
*Written by Jane and Joe*

#### /2 Expansión

De manera inversa a un sumidero de argumentos, podés *expandir* argumentos, arrays y diccionarios en una llamada a función con el operador `..spread`:

```
#let array = (2, 3, 5)
#calc.min(..array)
#let dict = (fill: blue)
#text(..dict)[Hello]
```

2 Hello

#### /3 Constructor

Construye argumentos expandibles en el lugar.

Esta función se comporta como `let args(..sink) = sink`.

```
#let args = arguments(stroke: red, inset: 1em, [Body])
#box(..args)
```

Body

```
arguments(.. any ) → arguments
```

##### /3.1 **arguments** `any` *Obligatorio* *Posicional* *Variádico*

Los argumentos que se construyen.

## /4 Funciones

### /4.1 len Desde: 0.15.0

La cantidad de argumentos, posicionales o con nombre.

```
self.len() → int
```

### /4.2 at Desde: 0.12.0

Devuelve el argumento posicional del índice especificado o el argumento con nombre del nombre especificado.

Si la clave es un entero[3.5.17], esto equivale a llamar primero a pos[3.5.1/4.3] y luego a array.at[3.5.2/3.4]. Si es una cadena de texto[3.5.28], equivale a llamar primero a named[3.5.1/4.4] y luego a dictionary.at[3.5.12/3.2].

También se puede acceder a los argumentos con nombre con la sintaxis de campos (por ejemplo, `arguments(key: 42).key`) si no hace falta un valor por defecto. A diferencia de los diccionarios[3.5.12], los campos de los argumentos no se pueden modificar.

```
self.at(
  int str,
  default: any,
) → any
```

#### /4.2.1 key int or str *Obligatorio* *Posicional*

El índice o el nombre del argumento que se quiere obtener.

#### /4.2.2 default any

Un valor por defecto para devolver si la clave no es válida.

### /4.3 pos Desde: siempre

Devuelve los argumentos posicionales capturados como un array.

```
self.pos() → array
```

### /4.4 named Desde: siempre

Devuelve los argumentos con nombre capturados como un diccionario.

```
self.named() → dictionary
```

### /4.5 filter Desde: 0.15.0

Produce un nuevo `arguments` solo con los argumentos cuyo valor pasa la prueba.

```
#{
  arguments(-1, a: 0, b: 1, 2)
    .filter(v => v > 0)
}
```

```
arguments(b: 1, 2)
```

```
self.filter(function) → arguments
```

#### /4.5.1 test function *Obligatorio* *Posicional*

La función que se aplica a cada valor. Debe devolver un booleano.

### /4.6 map Desde: 0.15.0

Produce un nuevo arguments transformando el valor de cada argumento con la función indicada.

```
#{
  arguments(0, a: 1, 2)
    .map(v => v + 1)
}
```

```
arguments(1, a: 2, 3)
```

```
self.map(function) → arguments
```

#### /4.6.1 mapper function *Obligatorio* *Posicional*

La función que se aplica a cada valor.

### 3.5.2 `array` Desde: siempre

Una secuencia de valores.

Podés construir un array encerrando entre paréntesis una secuencia de valores separados por comas. Los valores no tienen por qué ser del mismo tipo.

Podés acceder a los elementos de un array y actualizarlos con el método `.at()`. Los índices empiezan en cero y los índices negativos dan la vuelta hasta el final del array. Podés iterar sobre un array con un bucle for[3.3.5]. Los arrays se pueden sumar con el operador +, unir[3.3.2] y multiplicar por enteros.

**Nota:** Un array de longitud uno necesita una coma final, como en `(1,)`. Esto es para distinguirlo de una simple expresión entre paréntesis como `(1 + 2) * 3`. Un array vacío se escribe `()`.

#### /1 Ejemplo

```
#let values = (1, 7, 4, -3, 2)

#values.at(0) \
#(values.at(0) = 3)
#values.at(-1) \
#values.find(calc.even) \
#values.filter(calc.odd) \
#values.map(calc.abs) \
#values.rev() \
#(1, (2, 3)).flatten() \
#(("A", "B", "C")
    .join(", ", last: " and "))
```

```
1
2
4
(3, 7, -3)
(3, 7, 4, 3, 2)
(2, -3, 4, 7, 3)
(1, 2, 3)
A, B and C
```

#### /2 Constructor

Convierte un valor en un array.

Tené en cuenta que esta función solo está pensada para convertir en array un valor similar a una colección, no para crear un array a partir de elementos individuales. Para eso, usá la sintaxis de array `(1, 2, 3)` (o `(1,)` para un array de un solo elemento).

```
#let hi = "Hello 😃"
#array(bytes(hi))
```

```
(72, 101, 108, 108, 111, 32, 240, 159,
152, 131)
```

```
array( bytes  array  version ) → array
```

##### /2.1 **`value`** `bytes` or `array` or `version` *Obligatorio* *Posicional*

El valor que se debe convertir en array.

#### /3 Funciones

##### /3.1 **`len`** Desde: siempre

La cantidad de valores del array.

```
self.len() → int
```

##### /3.2 **`first`** Desde: siempre

Devuelve el primer ítem del array. Se puede usar en el lado izquierdo de una asignación. Devuelve el valor por defecto si el array está vacío, o falla con un error si no se especificó ningún valor por defecto.

```
self.first(default: any) → any
```

/3.2.1 **default** any

Un valor por defecto para devolver si el array está vacío.

## /3.3 last Desde: siempre

Devuelve el último ítem del array. Se puede usar en el lado izquierdo de una asignación. Devuelve el valor por defecto si el array está vacío, o falla con un error si no se especificó ningún valor por defecto.

```
self.last(default: any) → any
```

/3.3.1 **default** any

Un valor por defecto para devolver si el array está vacío.

## /3.4 at Desde: siempre

Devuelve el elemento del índice especificado en el array. Se puede usar en el lado izquierdo de una asignación. Devuelve el valor por defecto si el índice está fuera de rango, o falla con un error si no se especificó ningún valor por defecto.

```
self.at(
  int,
  default: any,
) → any
```

/3.4.1 **index** int *Obligatorio* *Posicional*

El índice del elemento que se quiere obtener. Si es negativo, se indexa desde el final.

/3.4.2 **default** any

Un valor por defecto para devolver si el índice está fuera de rango.

## /3.5 push Desde: siempre

Agrega un valor al final del array.

```
self.push(any) → none
```

/3.5.1 **value** any *Obligatorio* *Posicional*

El valor que se inserta al final del array.

## /3.6 pop Desde: siempre

Elimina el último elemento del array y lo devuelve. Falla con un error si el array está vacío.

```
self.pop() → any
```

## /3.7 insert Desde: siempre

Inserta un valor en el array en el índice especificado y desplaza a la derecha todos los elementos siguientes. Falla con un error si el índice está fuera de rango.

Para reemplazar un elemento de un array, usá at[3.5.2/3.4].

```
self.insert(
  int,
  any,
) → none
```

/3.7.1 **index** int *Obligatorio* *Posicional*

El índice en el que se inserta el elemento. Si es negativo, se indexa desde el final.

/3.7.2 **value** any *Obligatorio* *Posicional*

El valor que se inserta en el array.

## /3.8 remove Desde: siempre

Elimina del array el valor del índice especificado y lo devuelve.

```
self.remove(
  int,
  default: any,
) -> any
```

/3.8.1 **index** int *Obligatorio* *Posicional*

El índice del elemento que se quiere eliminar. Si es negativo, se indexa desde el final.

/3.8.2 **default** any

Un valor por defecto para devolver si el índice está fuera de rango.

## /3.9 slice Desde: siempre

Extrae una subsecuencia del array. Falla con un error si el índice de inicio o de fin está fuera de rango.

```
self.slice(
  int,
  none int,
  count: int,
) -> array
```

/3.9.1 **start** int *Obligatorio* *Posicional*

El índice inicial (inclusivo). Si es negativo, se indexa desde el final.

/3.9.2 **end** none or int *Posicional* Por defecto: none

El índice final (exclusivo). Si se omite, se extrae toda la porción hasta el final del array. Si es negativo, se indexa desde el final.

/3.9.3 **count** int

La cantidad de elementos que se extraen. Equivale a pasar `start + count` como posición end. Es mutuamente excluyente con end.

## /3.10 contains Desde: siempre

Si el array contiene el valor especificado.

Este método también tiene una sintaxis dedicada: podés escribir `2 in (1, 2, 3)` en lugar de `(1, 2, 3).contains(2)`.

```
self.contains( any ) -> bool
```

/3.10.1 **value** any *Obligatorio* *Posicional*

El valor que se busca.

## /3.11 find Desde: siempre

Busca un elemento para el cual la función indicada devuelva `true` y devuelve la primera coincidencia, o none si no hay ninguna.

```
self.find( function ) -> none any
```

/3.11.1 **searcher** function *Obligatorio* *Posicional*

La función que se aplica a cada elemento. Debe devolver un booleano.

## /3.12 position Desde: siempre

Busca un elemento para el cual la función indicada devuelva `true` y devuelve el índice de la primera coincidencia, o `none` si no hay ninguna.

```
self.position(function) → none int
```

/3.12.1 **searcher** function *Obligatorio* *Posicional*

La función que se aplica a cada elemento. Debe devolver un booleano.

```
#let values = (1, 7, 4, 6, 9)
#values.position(x => calc.even(x)) \
// Or equivalently:
#values.position(calc.even)
```

```
2
2
```

## /3.13 range Desde: siempre

Crea un array formado por una secuencia de números.

Si pasás un solo parámetro posicional, se interpreta como el end del rango. Si pasás dos, describen el `start` y el end del rango.

Esta función está disponible tanto en el ámbito de la función array como de forma global.

```
#range(5) \
#range(2, 5) \
#range(20, step: 4) \
#range(21, step: 4) \
#range(5, 2, step: -1)
```

```
(0, 1, 2, 3, 4)
(2, 3, 4)
(0, 4, 8, 12, 16)
(0, 4, 8, 12, 16, 20)
(5, 4, 3)
```

```
array.range(
  int,
  int,
  inclusive: bool,
  step: int,
) → array
```

/3.13.1 **start** int *Posicional* Por defecto: `0`

El inicio del rango (inclusivo).

/3.13.2 **end** int *Obligatorio* *Posicional*

El final del rango.

/3.13.3 **inclusive** bool Por defecto: `false`

Si end es inclusivo.

```
#range(0, inclusive: true) \
#range(7, 10, inclusive: true) \
#range(-8, -4, inclusive: true) \
#range(-6, step: -2, inclusive: true)
```

```
(0,)
(7, 8, 9, 10)
(-8, -7, -6, -5, -4)
(0, -2, -4, -6)
```

/3.13.4 **step** `int` Por defecto: `1`

La distancia entre los números generados.

/3.14 **filter** Desde: siempre

Produce un array nuevo solo con los elementos del original para los cuales la función indicada devuelve `true`.

```
self.filter(function) → array
```

/3.14.1 **test** `function` *Obligatorio* *Posicional*

La función que se aplica a cada elemento. Debe devolver un booleano.

/3.15 **map** Desde: siempre

Produce un array nuevo en el que todos los elementos del original se transformaron con la función indicada.

```
self.map(function) → array
```

/3.15.1 **mapper** `function` *Obligatorio* *Posicional*

La función que se aplica a cada elemento.

/3.16 **enumerate** Desde: 0.2.0

Devuelve un array nuevo con los valores junto a sus índices.

El array devuelto consiste en pares `(index, value)` en forma de arrays de longitud 2. Estos se pueden desestructurar[3-3-3] con un enlace let o un bucle for.

```
#for (i, value) in ("A", "B", "C").enumerate() {
  [#i: #value \ ]
}

#("A", "B", "C").enumerate(start: 1)
```

0: A
1: B
2: C

```
((1, "A"), (2, "B"), (3, "C"))
```

```
self.enumerate(start: int) → array
```

/3.16.1 **start** `int` Por defecto: `0`

El índice que se devuelve para el primer par de la lista resultante.

/3.17 **zip** Desde: 0.3.0

Combina el array con otros arrays elemento a elemento.

Devuelve un array de arrays, donde el `i`-ésimo array interno contiene todos los `i`-ésimos elementos de cada array original.

Si los arrays que se combinan tienen longitudes distintas, se combinan hasta el último elemento del array más corto y se ignoran todos los elementos restantes.

Esta función es variádica, es decir, podés combinar varios arrays a la vez: `(1, 2).zip(("A", "B"), (10, 20))` da `((1, "A", 10), (2, "B", 20))`.

```
self.zip(
  exact: bool,
  ..array,
) → array
```

/3.17.1 **exact** `bool` Por defecto: `false`

Si todos los arrays deben tener la misma longitud. Por ejemplo, `(1, 2).zip((1, 2, 3), exact: true)` produce un error.

/3.17.2 **others** `array` *Obligatorio* *Posicional* *Variádico*

Los arrays con los que se combina.

## /3.18 fold Desde: siempre

Pliega todos los elementos en un único valor usando una función acumuladora.

```
#let array = (1, 2, 3, 4)
#array.fold(0, (acc, x) => acc + x)
```

10

```
self.fold(
  any,
  function,
) → any
```

/3.18.1 **init** `any` *Obligatorio* *Posicional*

El valor inicial con el que se empieza.

/3.18.2 **folder** `function` *Obligatorio* *Posicional*

La función de plegado. Debe tener dos parámetros: uno para el valor acumulado y otro para un elemento.

## /3.19 sum Desde: 0.3.0

Suma todos los elementos (funciona con todos los tipos que se pueden sumar).

```
self.sum(default: any) → any
```

/3.19.1 **default** `any`

Qué devolver si el array está vacío. Hay que establecerlo si el array puede estar vacío.

## /3.20 product Desde: 0.3.0

Calcula el producto de todos los elementos (funciona con todos los tipos que se pueden multiplicar).

```
self.product(default: any) → any
```

/3.20.1 **default** `any`

Qué devolver si el array está vacío. Hay que establecerlo si el array puede estar vacío.

## /3.21 any Desde: siempre

Si la función indicada devuelve `true` para algún elemento del array.

```
self.any(function) → bool
```

/3.21.1 **test** `function` *Obligatorio* *Posicional*

La función que se aplica a cada elemento. Debe devolver un booleano.

### /3.22 all Desde: siempre

Si la función indicada devuelve true para todos los elementos del array.

```
self.all(function) → bool
```

#### /3.22.1 test function *Obligatorio* *Posicional*

La función que se aplica a cada elemento. Debe devolver un booleano.

### /3.23 flatten Desde: siempre

Combina todos los arrays anidados en un único array plano.

```
self.flatten() → array
```

### /3.24 rev Desde: siempre

Devuelve un array nuevo con los mismos elementos, pero en orden inverso.

```
self.rev() → array
```

### /3.25 split Desde: siempre

Divide el array en las apariciones del valor especificado.

```
#(1, 1, 2, 3, 2, 4, 5).split(2)
```

```
((1, 1), (3,), (4, 5))
```

```
self.split( any ) → array
```

#### /3.25.1 at any *Obligatorio* *Posicional*

El valor por el que se divide.

### /3.26 join Desde: siempre

Combina todos los elementos del array en uno solo.

```
self.join(
  none any ,
  last: any ,
  default: none any ,
) → any
```

#### /3.26.1 separator none or any *Posicional* Por defecto: none

Un valor que se inserta entre cada elemento del array.

#### /3.26.2 last any

Un separador alternativo entre los dos últimos elementos.

#### /3.26.3 default none or any Por defecto: none

Qué devolver si el array está vacío.

### /3.27 intersperse Desde: 0.8.0

Devuelve un array con una copia del valor separador colocada entre los elementos adyacentes.

```typ
#("A", "B", "C").intersperse("-")
```

```
("A", "-", "B", "-", "C")
```

```
self.intersperse( any ) → array
```

/3.27.1 **separator** any *Obligatorio* *Posicional*

El valor que se coloca entre cada par de elementos adyacentes.

## /3.28 chunks Desde: 0.11.0

Divide un array en fragmentos que no se superponen, empezando por el principio y terminando con un único fragmento restante.

Todos los fragmentos menos el último tienen `chunk-size` elementos. Si `exact` es `true`, el fragmento restante se descarta si contiene menos de `chunk-size` elementos.

```typ
#let array = (1, 2, 3, 4, 5, 6, 7, 8)
#array.chunks(3) \
#array.chunks(3, exact: true)
```

```
((1, 2, 3), (4, 5, 6), (7, 8))
((1, 2, 3), (4, 5, 6))
```

```
self.chunks(
  int ,
  exact: bool ,
) → array
```

/3.28.1 **chunk-size** int *Obligatorio* *Posicional*

Cuántos elementos puede contener como máximo cada fragmento.

/3.28.2 **exact** bool Por defecto: `false`

Si se descarta el fragmento restante cuando su tamaño es menor que `chunk-size`.

## /3.29 windows Desde: 0.12.0

Devuelve ventanas deslizantes de `window-size` elementos sobre un array.

Si la longitud del array es menor que `window-size`, devuelve un array vacío.

```typ
#let array = (1, 2, 3, 4, 5, 6, 7, 8)
#array.windows(5)
```

```
(
  (1, 2, 3, 4, 5),
  (2, 3, 4, 5, 6),
  (3, 4, 5, 6, 7),
  (4, 5, 6, 7, 8),
)
```

```
self.windows( int ) → array
```

/3.29.1 **window-size** int *Obligatorio* *Posicional*

Cuántos elementos contendrá cada ventana.

## /3.30 sorted Desde: siempre

Devuelve una versión ordenada de este array, opcionalmente según una función de clave. El algoritmo de ordenamiento usado es estable.

Devuelve un error si no se pudo comparar un par de valores seleccionados para la comparación, o si la función de clave o de comparación (si se indica) produce un error.

Para ordenar según varios criterios a la vez, por ejemplo en caso de igualdad entre algunos criterios, la función de clave puede devolver un array. Los resultados se ordenan lexicográficamente.

```
#let array = (
  (a: 2, b: 4),
  (a: 1, b: 5),
  (a: 2, b: 3),
)
#array.sorted(key: it => (it.a, it.b))
```

```
((a: 1, b: 5), (a: 2, b: 3), (a: 2, b: 4))
```

```
self.sorted(
  key: function,
  by: function,
) → array
```

/3.30.1 **key** function

Si se indica, aplica esta función a cada elemento del array para determinar las claves por las que se ordena.

/3.30.2 **by** function

Si se indica, usa esta función para comparar cada par de elementos del array.

La función recibirá dos elementos del array para compararlos y debería devolver un booleano que indique su orden: `true` indica que los elementos están en orden, mientras que `false` indica que hay que intercambiarlos. Para mantener la estabilidad del ordenamiento, si los dos elementos son iguales, la función debería devolver `true`.

Si esta función no ordena correctamente los elementos (por ejemplo, al devolver `false` tanto para `(x, y)` como para `(y, x)`, o para `(x, x)`), el array resultante tendrá un orden no especificado.

Cuando se usa junto con `key`, a `by` se le pasan las claves en lugar de los elementos.

```
#(
  "sorted",
  "by",
  "decreasing",
  "length",
).sorted(
  key: s => s.len(),
  by: (l, r) => l >= r,
)
```

```
("decreasing", "sorted", "length", "by")
```

/3.31 **dedup** Desde: 0.7.0

Elimina los elementos duplicados del array.

Devuelve un array nuevo sin los elementos duplicados. Solo se conserva el primer elemento de cada duplicado.

```
#(3, 3, 1, 2, 3).dedup()
```

```
(3, 1, 2)
```

```
self.dedup(key: function) → array
```

#### /3.31.1 key function

Si se indica, aplica esta función a cada elemento del array para determinar las claves por las que se elimina la duplicación.

```
#("apple", "banana", " apple ").dedup(key: s => s.trim())
```

```
("apple", "banana")
```

### /3.32 to-dict Desde: 0.12.0

Convierte un array de pares en un diccionario. El primer valor de cada par es la clave y el segundo, el valor.

Si la misma clave aparece varias veces, se selecciona el último valor.

```
#(
  ("apples", 2),
  ("peaches", 3),
  ("apples", 5),
).to-dict()
```

```
(apples: 5, peaches: 3)
```

```
self.to-dict() → dictionary
```

### /3.33 reduce Desde: 0.12.0

Reduce los elementos a uno solo, aplicando repetidamente una operación de reducción.

Si el array está vacío, devuelve none; de lo contrario, devuelve el resultado de la reducción.

La función de reducción es una clausura con dos argumentos: un "acumulador" y un elemento.

Para arrays con al menos un elemento, esto equivale a array.fold[3.5.2/3.18] con el primer elemento del array como valor inicial del acumulador, plegando en él cada elemento siguiente.

```
#let array = (2, 1, 4, 3)
#array.reduce((acc, x) => calc.max(acc, x))
```

```
4
```

```
self.reduce(function) → any
```

#### /3.33.1 reducer function *Obligatorio* *Posicional*

La función de reducción. Debe tener dos parámetros: uno para el valor acumulado y otro para un elemento.

### 3.5.3 `assert` Desde: siempre

Asegura que se cumpla una condición.

Falla con un error si la condición no se cumple. No produce ninguna salida en el documento.

Si querés comprobar la igualdad entre dos valores, mirá `assert.eq`[3.5.3/3.1] y `assert.ne`[3.5.3/3.2].

#### /1 Ejemplo

```
#assert(1 < 2, message: "math broke")
```

#### /2 Parámetros

```
assert(
  bool,
  message: str,
) → none
```

##### /2.1 **condition** `bool` *Obligatorio* *Posicional*

La condición que debe ser verdadera para que la aserción pase.

##### /2.2 **message** `str`

El mensaje de error cuando la aserción falla.

#### /3 Funciones

##### /3.1 **eq** Desde: 0.4.0

Asegura que dos valores sean iguales.

Falla con un error si el primer valor no es igual al segundo. No produce ninguna salida en el documento.

```
#assert.eq(10, 10)
```

```
assert.eq(
  any,
  any,
  message: str,
) → none
```

###### /3.1.1 **left** `any` *Obligatorio* *Posicional*

El primer valor que se compara.

###### /3.1.2 **right** `any` *Obligatorio* *Posicional*

El segundo valor que se compara.

###### /3.1.3 **message** `str`

Un mensaje opcional que se muestra en caso de error en lugar de las representaciones de los valores comparados.

##### /3.2 **ne** Desde: 0.4.0

Asegura que dos valores no sean iguales.

Falla con un error si el primer valor es igual al segundo. No produce ninguna salida en el documento.

```
#assert.ne(3, 4)
```

```
assert.ne(
  any,
  any,
  message: str,
) → none
```

/3.2.1 **left** `any` *Obligatorio* *Posicional*

El primer valor que se compara.

/3.2.2 **right** `any` *Obligatorio* *Posicional*

El segundo valor que se compara.

/3.2.3 **message** `str`

Un mensaje opcional que se muestra en caso de error en lugar de las representaciones de los valores comparados.

### 3.5.4 `auto`

Un valor que indica un valor por defecto inteligente.

El tipo auto tiene exactamente un valor: `auto`.

Los parámetros que admiten el valor `auto` tienen algún valor por defecto inteligente o un comportamiento contextual. Un buen ejemplo es el parámetro de dirección del texto[3.7.12/2.19]. Establecerlo en `auto` permite que Typst determine automáticamente la dirección a partir del idioma del texto[3.7.12/2.16].

### 3.5.5 Biblioteca estándar

Un módulo que contiene todos los elementos accesibles globalmente.

#### /1 Usar definiciones "ocultas" (shadowed)

El módulo `std` es útil siempre que hayas reemplazado un nombre del ámbito global (esto se llama *shadowing*). Por ejemplo, podrías haber usado el nombre `text` para un parámetro. Para poder seguir accediendo al elemento `text`, escribí `std.text`.

```
#let par = [My special paragraph.]
#let special(text) = {
  set std.text(style: "italic")
  set std.par.line(numbering: "1")
  text
}

#special(par)

#lorem(10)
```

1 *My special paragraph.*

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

#### /2 Acceso condicional

También podés usar esto en combinación con el constructor de diccionarios[3.5.12/2] para acceder condicionalmente a definiciones globales. Esto puede ser útil, por ejemplo, para usar funcionalidad nueva o experimental cuando está disponible, recurriendo a una implementación alternativa si se usa en una versión más antigua de Typst. En particular, esto nos permite crear polyfills[A].

Puede ser tan simple como crear un alias para evitar mensajes de advertencia, por ejemplo, usar condicionalmente `pattern` en la versión 0.12 de Typst, pero usar `tiling`[3.11.13] en las versiones más nuevas. Como los parámetros que acepta la función `tiling` coinciden con los de la antigua función `pattern`, usar la función `tiling` cuando esté disponible y recurrir a `pattern` en caso contrario unificará el uso en todas las versiones. Tené en cuenta que, al crear un polyfill, `sys.version`[3.5.27/1.1] también puede ser muy útil.

```
#let tiling = if "tiling" in std { tiling } else { pattern }

...
```

[A] https://en.wikipedia.org/wiki/Polyfill_(programming)

### 3.5.6 `bool` Desde: siempre

Un tipo con dos estados.

El tipo booleano tiene dos valores: `true` y `false`. Indica si algo está activo o habilitado.

#### /1 Ejemplo

```
#false \
#true \
#(1 < 2)
```

```
false
true
true
```

## 3.5.7 `bytes` Desde: 0.7.0

Una secuencia de bytes.

Conceptualmente es similar a un array de enteros[3.5.17] entre `0` y `255`, pero se representa de forma mucho más eficiente. Podés iterar sobre ella con un bucle for[3.3.5].

Podés convertir
- una cadena de texto[3.5.28] o un array[3.5.2] de enteros en bytes con el constructor de `bytes`[3.5.7]
- bytes en una cadena de texto con el constructor de `str`[3.5.28], con codificación UTF-8
- bytes en un array de enteros con el constructor de `array`[3.5.2]

Al leer[3.13.5] datos de un archivo, podés decidir si cargarlos como cadena de texto o como bytes sin procesar.

```
#bytes((123, 160, 22, 0)) \
#bytes("Hello 😃")

#let data = read(
  "rhino.png",
  encoding: none,
)

// Magic bytes.
#array(data.slice(0, 4)) \
#str(data.slice(1, 4))
```

```
bytes(4)
bytes(10)
(137, 80, 78, 71)
PNG
```

### /1 Constructor

Convierte un valor en bytes.

- Las cadenas de texto se codifican en UTF-8.
- Los arrays de enteros entre `0` y `255` se convierten directamente. La representación dedicada de bytes es mucho más eficiente que la de array y, por eso, se suele usar para buffers de bytes grandes (por ejemplo, datos de imágenes).

```
#bytes("Hello 😃") \
#bytes((123, 160, 22, 0))
```

```
bytes(10)
bytes(4)
```

```
bytes( str bytes array ) → bytes
```

#### /1.1 **value** `str` or `bytes` or `array` *Obligatorio* *Posicional*

El valor que se debe convertir en bytes.

### /2 Funciones

#### /2.1 `len` Desde: 0.7.0

La longitud en bytes.

```
self.len() → int
```

#### /2.2 `at` Desde: 0.7.0

Devuelve el byte del índice especificado. Devuelve el valor por defecto si el índice está fuera de rango, o falla con un error si no se especificó ningún valor por defecto.

```
self.at(
  int,
  default: any,
) → any
```

/2.2.1 **index** `int` *Obligatorio* *Posicional*

El índice del byte que se quiere obtener.

/2.2.2 **default** `any`

Un valor por defecto para devolver si el índice está fuera de rango.

### /2.3 slice Desde: 0.7.0

Extrae una subsecuencia de los bytes. Falla con un error si el índice de inicio o de fin está fuera de rango.

```
self.slice(
  int,
  none int,
  count: int,
) → bytes
```

/2.3.1 **start** `int` *Obligatorio* *Posicional*

El índice inicial (inclusivo).

/2.3.2 **end** `none` or `int` *Posicional* Por defecto: none

El índice final (exclusivo). Si se omite, se extrae toda la porción hasta el final.

/2.3.3 **count** `int`

La cantidad de elementos que se extraen. Equivale a pasar `start + count` como posición `end`. Es mutuamente excluyente con `end`.

### 3.5.8 `content` Desde: siempre

Un fragmento de contenido del documento.

Este tipo está en el corazón de Typst. Todo el marcado que escribís y la mayoría de las funciones[3.5.16] que llamás producen valores de contenido. Podés crear un valor de contenido encerrando marcado entre corchetes. Así es también como le pasás contenido a las funciones.

#### /1 Ejemplo

```
Type of *Hello!* is
#type([*Hello!*])
```

Type of **Hello!** is `content`

El contenido se puede sumar con el operador +, unir[3.3.2] y multiplicar por enteros. Dondequiera que se espere contenido, también podés pasar una cadena de texto[3.5.28] o none.

#### /2 Representación

El contenido consiste en elementos con campos. Al construir un elemento con su *función de elemento*, le proporcionás estos campos como argumentos y, cuando tenés un valor de contenido, podés acceder a sus campos con la sintaxis de acceso a campos[3.3.6].

Algunos campos son obligatorios: hay que proporcionarlos al construir un elemento y, en consecuencia, siempre están disponibles mediante el acceso a campos en el contenido de ese tipo. Los campos obligatorios se marcan como tales en la documentación.

La mayoría de los campos son opcionales: igual que los obligatorios, se pueden pasar a la función de elemento para configurarlos en un único elemento. Sin embargo, también se pueden configurar con reglas set[3.2.1] para aplicarlos a todos los elementos de un ámbito. Los campos opcionales solo están disponibles con la sintaxis de acceso a campos cuando se pasaron explícitamente a la función de elemento, no cuando provienen de una regla set.

Cada elemento tiene un aspecto por defecto. Sin embargo, también podés personalizar por completo su aspecto con una regla show[3.2.2]. A la regla show se le pasa el elemento. Puede acceder a los campos del elemento y producir contenido arbitrario a partir de ellos.

En la app web, podés pasar el cursor sobre una variable de contenido para ver exactamente de qué elementos se compone el contenido y qué campos tienen. Alternativamente, podés inspeccionar la salida de la función repr[3.5.25].

#### /3 Funciones

##### /3.1 `func` Desde: siempre

La función de elemento del contenido. Esta función se puede usar para crear el elemento que contiene este contenido. Se puede usar en reglas set y show del elemento. Se puede comparar con funciones globales para comprobar si tenés un tipo específico de elemento.

```
self.func() → function
```

##### /3.2 `has` Desde: siempre

Si el contenido tiene el campo especificado.

```
self.has(str) → bool
```

###### /3.2.1 `field` `str` *Obligatorio* *Posicional*

El campo que se busca.

### /3.3 at Desde: siempre

Accede al campo especificado del contenido. Devuelve el valor por defecto si el campo no existe, o falla con un error si no se especificó ningún valor por defecto.

```
self.at(
  str,
  default: any,
) → any
```

#### /3.3.1 field str *Obligatorio* *Posicional*

El campo al que se accede.

#### /3.3.2 default any

Un valor por defecto para devolver si el campo no existe.

### /3.4 fields Desde: 0.5.0

Devuelve los campos de este contenido.

```
#rect(
  width: 10cm,
  height: 10cm,
).fields()
```

```
(width: 0% + 283.46pt, height: 0% +
283.46pt)
```

```
self.fields() → dictionary
```

### /3.5 location Desde: siempre

La ubicación del contenido. Solo está disponible en el contenido devuelto por query[3.12.6] o proporcionado por una regla show[3.2.2]; para otro contenido será none. La ubicación resultante se puede usar con contadores[3.12.1], estado[3.12.7] y consultas[3.12.6].

```
self.location() → none location
```

## 3.5.9 Cálculo

Módulo para cálculos y procesamiento de valores numéricos.

Estas definiciones forman parte del módulo `calc` y no se importan por defecto.

### /1 Constantes

#### /1.1 `inf` `float` Desde: siempre

Infinito ($+\infty$) como valor.

Es el mismo valor que `float.inf`[3.5.15/3.1]. Consultá la documentación correspondiente para más detalles.

#### /1.2 `pi` `float` Desde: siempre

Constante de Arquímedes ($\pi$).

Es la razón entre la circunferencia de un círculo y su diámetro.

```
#calc.pi
```

3.141592653589793

#### /1.3 `tau` `float` Desde: 0.8.0

La circunferencia de un círculo unitario ($\tau$).

La constante $\tau$ se define por la relación $\tau = 2\pi$.

```
#calc.tau \
#(2 * calc.pi)
```

6.283185307179586
6.283185307179586

#### /1.4 `e` `float` Desde: siempre

Número de Euler ($e$).

Es la base del logaritmo natural y de la función exponencial.

```
#calc.e
```

2.718281828459045

### /2 Funciones

#### /2.1 `abs` Desde: siempre

Calcula el valor absoluto de un valor numérico.

```
#calc.abs(-5) \
#calc.abs(5pt - 2cm) \
#calc.abs(2fr) \
#calc.abs(decimal("-342.440"))
```

5
51.69pt
2fr
342.440

`calc.abs(` `int` `float` `length` `angle` `ratio` `fraction` `decimal` `)` → `any`

/2.1.1 **value** `int` or `float` or `length` or `angle` or `ratio` or `fraction` or `decimal` *Obligatorio* *Posicional*

El valor cuyo valor absoluto se calcula.

/2.2 **pow** Desde: siempre

Eleva un valor a un exponente.

```
#calc.pow(2, 3) \
#calc.pow(decimal("2.5"), 2)
```

8
6.25

```
calc.pow(
  int float decimal,
  int float,
) → int float decimal
```

/2.2.1 **base** `int` or `float` or `decimal` *Obligatorio* *Posicional*

La base de la potencia.

Si es un decimal[3.5.11], el exponente solo puede ser un entero[3.5.17].

/2.2.2 **exponent** `int` or `float` *Obligatorio* *Posicional*

El exponente de la potencia.

/2.3 **exp** Desde: 0.5.0

Eleva $e$ a un exponente.

```
#calc.exp(1)
```

2.7182818284590455

```
calc.exp(int float) → float
```

/2.3.1 **exponent** `int` or `float` *Obligatorio* *Posicional*

El exponente de la potencia.

/2.4 **sqrt** Desde: siempre

Calcula la raíz cuadrada de un número.

```
#calc.sqrt(16) \
#calc.sqrt(2.5)
```

4
1.5811388300841898

```
calc.sqrt(int float) → float
```

/2.4.1 **value** `int` or `float` *Obligatorio* *Posicional*

El número cuya raíz cuadrada se calcula. Debe ser no negativo.

/2.5 **root** Desde: 0.11.0

Calcula la raíz real $n^{\text{ésima}}$ de un número.

Si el número es negativo, $n$ debe ser impar.

```
#calc.root(16.0, 4) \
#calc.root(27.0, 3)
```

2
3

```
calc.root(
  float,
  int,
) → float
```

/2.5.1 **radicand** `float` *Obligatorio* *Posicional*

La expresión de la que se toma la raíz.

/2.5.2 **index** `int` *Obligatorio* *Posicional*

El valor de $n$.

/2.6 **sin** Desde: siempre

Calcula el seno de un ángulo.

Cuando se la llama con un entero o un flotante, se interpretan como radianes.

```
#calc.sin(1.5) \
#calc.sin(90deg)
```

0.9974949866040544
1

```
calc.sin( int float angle ) → float
```

/2.6.1 **angle** `int` or `float` or `angle` *Obligatorio* *Posicional*

El ángulo cuyo seno se calcula.

/2.7 **cos** Desde: siempre

Calcula el coseno de un ángulo.

Cuando se la llama con un entero o un flotante, se interpretan como radianes.

```
#calc.cos(1.5) \
#calc.cos(90deg)
```

0.0707372016677029
0.00000000000000006123233995736766

```
calc.cos( int float angle ) → float
```

/2.7.1 **angle** `int` or `float` or `angle` *Obligatorio* *Posicional*

El ángulo cuyo coseno se calcula.

/2.8 **tan** Desde: siempre

Calcula la tangente de un ángulo.

Cuando se la llama con un entero o un flotante, se interpretan como radianes.

```
#calc.tan(1.5) \
#calc.tan(90deg)
```

```
14.101419947171719
16331239353195370
```

```
calc.tan( int float angle ) → float
```

/2.8.1 **angle** `int` or `float` or `angle` *Obligatorio* *Posicional*

El ángulo cuya tangente se calcula.

/2.9 **asin** Desde: siempre

Calcula el arcoseno de un número.

```
#calc.asin(0) \
#calc.asin(1)
```

```
0deg
90deg
```

```
calc.asin( int float ) → angle
```

/2.9.1 **value** `int` or `float` *Obligatorio* *Posicional*

El número cuyo arcoseno se calcula. Debe estar entre $-1$ y $1$.

/2.10 **acos** Desde: siempre

Calcula el arcocoseno de un número.

```
#calc.acos(0) \
#calc.acos(1)
```

```
90deg
0deg
```

```
calc.acos( int float ) → angle
```

/2.10.1 **value** `int` or `float` *Obligatorio* *Posicional*

El número cuyo arcocoseno se calcula. Debe estar entre $-1$ y $1$.

/2.11 **atan** Desde: siempre

Calcula el arcotangente de un número.

```
#calc.atan(0) \
#calc.atan(1)
```

```
0deg
45deg
```

```
calc.atan( int float ) → angle
```

/2.11.1 **value** `int` or `float` *Obligatorio* *Posicional*

El número cuyo arcotangente se calcula.

/2.12 **atan2** Desde: 0.3.0

Calcula el arcotangente de cuatro cuadrantes de una coordenada.

El arcotangente de cuatro cuadrantes de $(x, y)$ se define como el argumento del número complejo $x + iy$.

Devuelve un angle[3.10.3] entre -180deg y 180deg.

Tené en cuenta que esta función acepta $(x, y)$, no $(y, x)$.

```
#calc.atan2(1, 1) \
#calc.atan2(-2, -3)
```

```
45deg
-123.69deg
```

```
calc.atan2(
  int float,
  int float,
) → angle
```

/2.12.1 **x** `int` or `float` *Obligatorio* *Posicional*

La coordenada $x$.

/2.12.2 **y** `int` or `float` *Obligatorio* *Posicional*

La coordenada $y$.

### /2.13 sinh Desde: siempre

Calcula el seno hiperbólico de un ángulo hiperbólico.

El seno hiperbólico de $x$ se define de la siguiente manera:

$$\frac{e^x - e^{-x}}{2}$$

```
#calc.sinh(0) \
#calc.sinh(1.5)
```

```
0
2.1292794550948173
```

```
calc.sinh( float ) → float
```

/2.13.1 **value** `float` *Obligatorio* *Posicional*

El ángulo hiperbólico cuyo seno hiperbólico se calcula.

### /2.14 cosh Desde: siempre

Calcula el coseno hiperbólico de un ángulo hiperbólico.

El coseno hiperbólico de $x$ se define de la siguiente manera:

$$\frac{e^x + e^{-x}}{2}$$

```
#calc.cosh(0) \
#calc.cosh(1.5)
```

```
1
2.352409615243247
```

```
calc.cosh( float ) → float
```

/2.14.1 **value** `float` *Obligatorio* *Posicional*

El ángulo hiperbólico cuyo coseno hiperbólico se calcula.

### /2.15 tanh Desde: siempre

Calcula la tangente hiperbólica de un ángulo hiperbólico.

La tangente hiperbólica de $x$ se define de la siguiente manera:

$$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$

```
#calc.tanh(0) \
#calc.tanh(1.5)
```

```
0
0.9051482536448664
```

```
calc.tanh( float ) → float
```

#### /2.15.1 value `float` *Obligatorio* *Posicional*

El ángulo hiperbólico cuya tangente hiperbólica se calcula.

### /2.16 asinh Desde: 0.15.0

Calcula el seno hiperbólico inverso de un número.

El seno hiperbólico inverso de $x$ se define de la siguiente manera:

$$\ln\left(x + \sqrt{x^2 + 1}\right)$$

```
#calc.asinh(0) \
#calc.asinh(1)
```

```
0
0.881373587019543
```

```
calc.asinh( float ) → float
```

#### /2.16.1 value `float` *Obligatorio* *Posicional*

El número cuyo seno hiperbólico inverso se calcula.

### /2.17 acosh Desde: 0.15.0

Calcula el coseno hiperbólico inverso de un número.

El coseno hiperbólico inverso de $x$ se define de la siguiente manera:

$$\ln\left(x + \sqrt{x^2 - 1}\right)$$

```
#calc.acosh(1) \
#calc.acosh(2.5)
```

```
0
1.566799236972411
```

```
calc.acosh( float ) → float
```

#### /2.17.1 value `float` *Obligatorio* *Posicional*

El número cuyo coseno hiperbólico inverso se calcula. Debe ser mayor o igual que 1.

### /2.18 atanh Desde: 0.15.0

Calcula la tangente hiperbólica inversa de un número.

La tangente hiperbólica inversa de $x$ se define de la siguiente manera:

$$\frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$$

```
#calc.atanh(0) \
#calc.atanh(0.5)
```

```
0
0.5493061443340548
```

```
calc.atanh( float ) → float
```

/2.18.1 **value** `float` *Obligatorio* *Posicional*

El número cuya tangente hiperbólica inversa se calcula. Debe estar entre $-1$ y $1$ (exclusivos).

/2.19 **log** Desde: siempre

Calcula el logaritmo de un número.

Si no se especifica la base, el logaritmo se calcula en base diez.

```
#calc.log(100)
```

```
2
```

```
calc.log(
  int float ,
  base: float ,
) → float
```

/2.19.1 **value** `int` or `float` *Obligatorio* *Posicional*

El número cuyo logaritmo se calcula. Debe ser estrictamente positivo.

/2.19.2 **base** `float` Por defecto: `10.0`

La base del logaritmo. No puede ser cero.

/2.20 **ln** Desde: 0.5.0

Calcula el logaritmo natural de un número.

```
#calc.ln(calc.e)
```

```
1
```

```
calc.ln( int float ) → float
```

/2.20.1 **value** `int` or `float` *Obligatorio* *Posicional*

El número cuyo logaritmo se calcula. Debe ser estrictamente positivo.

/2.21 **erf** Desde: 0.15.0

Aplica la función de error a un número.

El valor de la función de error en $x$ se define de la siguiente manera:

$$\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,\mathrm{d}t$$

```typ
#calc.erf(0.2)
```

0.22270258921047847

```
calc.erf(float) → float
```

/2.21.1 **value** `float` *Obligatorio* *Posicional*

El número en el que se calcula la función de error.

/2.22 **fact** Desde: 0.3.0

Calcula el factorial de un número.

```typ
#calc.fact(5)
```

120

```
calc.fact(int) → int
```

/2.22.1 **number** `int` *Obligatorio* *Posicional*

El número cuyo factorial se calcula. Debe ser no negativo.

/2.23 **perm** Desde: 0.3.0

Calcula una permutación.

Devuelve la $k$-permutación de $n$, o la cantidad de maneras de elegir $k$ elementos de un conjunto de $n$ teniendo en cuenta el orden, definida de la siguiente manera:

$$\begin{cases} 0 & \text{si } k > n \\ \frac{n!}{(n-k)!} & \text{si } k \leq n \end{cases}$$

```typ
#calc.perm(5, 3)
```

60

```
calc.perm(
  int,
  int,
) → int
```

/2.23.1 **base** `int` *Obligatorio* *Posicional*

El valor de $n$: la cantidad de elementos entre los que se elige. Debe ser no negativo.

/2.23.2 **numbers** `int` *Obligatorio* *Posicional*

El valor de $k$: la cantidad de elementos que se eligen. Debe ser no negativo.

/2.24 **binom** Desde: 0.3.0

Calcula un coeficiente binomial.

Devuelve la $k$-combinación de $n$, o la cantidad de maneras de elegir $k$ elementos de un conjunto de $n$ sin tener en cuenta el orden, definida de la siguiente manera:

$$\begin{cases} \frac{n!}{k!(n-k)!} & \text{si } 0 \leq k \leq n \\ 0 & \text{en otro caso} \end{cases}$$

```
#calc.binom(10, 5)
```

252

```
calc.binom(
  int,
  int,
) → int
```

/2.24.1 **n** `int` *Obligatorio* *Posicional*

El valor de $n$: la cantidad de elementos entre los que se elige. Debe ser no negativo.

/2.24.2 **k** `int` *Obligatorio* *Posicional*

El valor de $k$: la cantidad de elementos que se eligen. Debe ser no negativo.

/2.25 **gcd** Desde: 0.3.0

Calcula el máximo común divisor de dos enteros.

Producirá un error si el resultado de la división entera fuera mayor que el máximo entero con signo de 64 bits.

```
#calc.gcd(7, 42)
```

7

```
calc.gcd(
  int,
  int,
) → int
```

/2.25.1 **a** `int` *Obligatorio* *Posicional*

El primer entero.

/2.25.2 **b** `int` *Obligatorio* *Posicional*

El segundo entero.

/2.26 **lcm** Desde: 0.3.0

Calcula el mínimo común múltiplo de dos enteros.

```
#calc.lcm(96, 13)
```

1248

```
calc.lcm(
  int,
  int,
) → int
```

/2.26.1 **a** `int` *Obligatorio* *Posicional*

El primer entero.

/2.26.2 **b** `int` *Obligatorio* *Posicional*

El segundo entero.

/2.27 **floor** Desde: siempre

Redondea un número hacia abajo al entero más cercano.

Si el número ya es un entero, se devuelve sin cambios.

Tené en cuenta que esta función siempre devuelve un entero[3.5.17] y producirá un error si el float[3.5.15] o decimal[3.5.11] resultante es mayor que el máximo entero con signo de 64 bits o menor que el mínimo de ese tipo.

```
#calc.floor(500.1)
#assert(calc.floor(3) == 3)
#assert(calc.floor(3.14) == 3)
#assert(calc.floor(decimal("-3.14")) == -4)
```

500

```
calc.floor( int  float  decimal ) → int
```

/2.27.1 **value** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El número que se redondea hacia abajo.

/2.28 **ceil** Desde: siempre

Redondea un número hacia arriba al entero más cercano.

Si el número ya es un entero, se devuelve sin cambios.

Tené en cuenta que esta función siempre devuelve un entero[3.5.17] y producirá un error si el float[3.5.15] o decimal[3.5.11] resultante es mayor que el máximo entero con signo de 64 bits o menor que el mínimo de ese tipo.

```
#calc.ceil(500.1)
#assert(calc.ceil(3) == 3)
#assert(calc.ceil(3.14) == 4)
#assert(calc.ceil(decimal("-3.14")) == -3)
```

501

```
calc.ceil( int  float  decimal ) → int
```

/2.28.1 **value** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El número que se redondea hacia arriba.

/2.29 **trunc** Desde: 0.3.0

Devuelve la parte entera de un número.

Si el número ya es un entero, se devuelve sin cambios.

Tené en cuenta que esta función siempre devuelve un entero[3.5.17] y producirá un error si el float[3.5.15] o decimal[3.5.11] resultante es mayor que el máximo entero con signo de 64 bits o menor que el mínimo de ese tipo.

```
#calc.trunc(15.9)
#assert(calc.trunc(3) == 3)
```

```
#assert(calc.trunc(-3.7) == -3)
#assert(calc.trunc(decimal("8493.12949582390")) == 8493)
```

15

```
calc.trunc( int  float  decimal ) → int
```

/2.29.1 **value** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El número que se trunca.

/2.30 **fract** Desde: 0.3.0

Devuelve la parte fraccionaria de un número.

Si el número es un entero, devuelve `0`.

```
#calc.fract(-3.1)
#assert(calc.fract(3) == 0)
#assert(calc.fract(decimal("234.23949211")) == decimal("0.23949211"))
```

−0.10000000000000009

```
calc.fract( int  float  decimal ) → int  float  decimal
```

/2.30.1 **value** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El número que se trunca.

/2.31 **round** Desde: siempre

Redondea un número al entero más cercano.

Los valores intermedios (con parte fraccionaria de un medio) se redondean alejándose de cero.

Opcionalmente, se puede especificar una cantidad de decimales. Si es negativa, su valor absoluto indicará la cantidad de dígitos enteros significativos que se eliminan antes del punto decimal.

Tené en cuenta que esta función devuelve el mismo tipo que el operando. Es decir, aplicar `round` a un `float`[3.5.15] devuelve un `float` y aplicarlo a un `decimal`[3.5.11] devuelve otro `decimal`. Podés convertir explícitamente la salida de esta función en un entero con `int`[3.5.17], pero tené en cuenta que esa conversión producirá un error si el `float` o `decimal` es mayor que el máximo entero con signo de 64 bits o menor que el entero mínimo.

Además, esta función puede producir un error si se intenta redondear más allá del entero o `decimal` máximo o mínimo. Si el número es un `float`, ese intento hará que se devuelva `calc.inf`[3.5.9/1.1] o `-calc.inf` para el máximo y el mínimo, respectivamente.

```
#calc.round(3.1415, digits: 2)
#assert(calc.round(3) == 3)
#assert(calc.round(3.14) == 3)
#assert(calc.round(3.5) == 4.0)
#assert(calc.round(3333.45, digits: -2) == 3300.0)
#assert(calc.round(-48953.45, digits: -3) == -49000.0)
#assert(calc.round(3333, digits: -2) == 3300)
#assert(calc.round(-48953, digits: -3) == -49000)
#assert(calc.round(decimal("-6.5")) == decimal("-7"))
```

```
#assert(calc.round(decimal("7.123456789"), digits: 6) == decimal("7.123457"))
#assert(calc.round(decimal("3333.45"), digits: -2) == decimal("3300"))
#assert(calc.round(decimal("-48953.45"), digits: -3) == decimal("-49000"))
```

3.14

```
calc.round(
  int float decimal,
  digits: int,
) → int float decimal
```

/2.31.1 **value** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El número que se redondea.

/2.31.2 **digits** `int` Por defecto: `0`

Si es positivo, la cantidad de decimales.

Si es negativo, la cantidad de dígitos enteros significativos que se deben eliminar antes del punto decimal.

/2.32 **clamp** Desde: siempre

Limita un número entre un valor mínimo y uno máximo.

```
#calc.clamp(5, 0, 4)
#assert(calc.clamp(5, 0, 10) == 5)
#assert(calc.clamp(5, 6, 10) == 6)
#assert(calc.clamp(decimal("5.45"), 2, decimal("45.9")) == decimal("5.45"))
#assert(calc.clamp(decimal("5.45"), decimal("6.75"), 12) == decimal("6.75"))
```

4

```
calc.clamp(
  int float decimal,
  int float decimal,
  int float decimal,
) → int float decimal
```

/2.32.1 **value** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El número que se limita.

/2.32.2 **min** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El valor mínimo inclusivo.

/2.32.3 **max** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El valor máximo inclusivo.

/2.33 **min** Desde: siempre

Determina el mínimo de una secuencia de valores.

```
#calc.min(1, -3, -5, 20, 3, 6) \
#calc.min("typst", "is", "cool")
```

```
-5
cool
```

```
calc.min(.. any ) → any
```

/2.33.1 **values** `any` *Obligatorio* *Posicional* *Variádico*

La secuencia de valores de la que se extrae el mínimo. No debe estar vacía.

/2.34 **max** Desde: siempre

Determina el máximo de una secuencia de valores.

```
#calc.max(1, -3, -5, 20, 3, 6) \
#calc.max("typst", "is", "cool")
```

```
20
typst
```

```
calc.max(.. any ) → any
```

/2.34.1 **values** `any` *Obligatorio* *Posicional* *Variádico*

La secuencia de valores de la que se extrae el máximo. No debe estar vacía.

/2.35 **even** Desde: siempre

Determina si un entero es par.

```
#calc.even(4) \
#calc.even(5) \
#range(10).filter(calc.even)
```

```
true
false
(0, 2, 4, 6, 8)
```

```
calc.even( int ) → bool
```

/2.35.1 **value** `int` *Obligatorio* *Posicional*

El número cuya paridad se comprueba.

/2.36 **odd** Desde: siempre

Determina si un entero es impar.

```
#calc.odd(4) \
#calc.odd(5) \
#range(10).filter(calc.odd)
```

```
false
true
(1, 3, 5, 7, 9)
```

```
calc.odd( int ) → bool
```

/2.36.1 **value** `int` *Obligatorio* *Posicional*

El número cuya imparidad se comprueba.

/2.37 **rem** Desde: 0.3.0

Calcula el resto de dos números (módulo con signo).

El valor de `calc.rem(x, y)` siempre tiene el mismo signo que `x` y es menor en magnitud que `y`.

Puede producir un error si se le da una entrada `decimal`[3.5.11] y el dividendo es demasiado pequeño en magnitud en comparación con el divisor.

```
#calc.rem(7, 3) \
#calc.rem(7, -3) \
#calc.rem(-7, 3) \
#calc.rem(-7, -3) \
#calc.rem(1.75, 0.5)
```

1
1
−1
−1
0.25

```
calc.rem(
  int float decimal,
  int float decimal,
) → int float decimal
```

/2.37.1 **dividend** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El dividendo del resto.

/2.37.2 **divisor** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El divisor del resto.

## /2.38 div-euclid Desde: 0.10.0

Realiza la división euclidiana de dos números.

El resultado de este cálculo es el de una división redondeada al entero `n` tal que el dividendo es mayor o igual que `n` veces el divisor.

Puede producir un error si el número resultante es mayor que el valor máximo o menor que el valor mínimo de su tipo.

```
#calc.div-euclid(7, 3) \
#calc.div-euclid(7, -3) \
#calc.div-euclid(-7, 3) \
#calc.div-euclid(-7, -3) \
#calc.div-euclid(1.75, 0.5) \
#calc.div-euclid(decimal("1.75"), decimal("0.5"))
```

2
−2
−3
3
3
3

```
calc.div-euclid(
  int float decimal,
  int float decimal,
) → int float decimal
```

/2.38.1 **dividend** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El dividendo de la división.

/2.38.2 **divisor** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El divisor de la división.

/2.39 **rem-euclid** Desde: 0.10.0

Calcula el menor resto no negativo de una división (módulo no negativo).

Advertencia: debido a un error de redondeo de punto flotante, el resto puede ser igual al valor absoluto del divisor si el dividendo es mucho menor en magnitud que el divisor y el dividendo es negativo. Esto solo se aplica a entradas de punto flotante.

Además, puede producir un error si se le da una entrada `decimal`[3.5.11] y el dividendo es demasiado pequeño en magnitud en comparación con el divisor.

```
#calc.rem-euclid(7, 3) \
#calc.rem-euclid(7, -3) \
#calc.rem-euclid(-7, 3) \
#calc.rem-euclid(-7, -3) \
#calc.rem-euclid(1.75, 0.5) \
#calc.rem-euclid(decimal("1.75"), decimal("0.5"))
```

1
1
2
2
0.25
0.25

```
calc.rem-euclid(
  int float decimal,
  int float decimal,
) → int float decimal
```

/2.39.1 **dividend** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El dividendo del resto.

/2.39.2 **divisor** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El divisor del resto.

/2.40 **quo** Desde: 0.3.0

Calcula el cociente (división con redondeo hacia abajo) de dos números.

Tené en cuenta que esta función siempre devuelve un entero[3.5.17] y producirá un error si el número resultante es mayor que el máximo entero con signo de 64 bits o menor que el mínimo de ese tipo.

```
$ "quo"(a, b) &= floor(a/b) \
  "quo"(14, 5) &= #calc.quo(14, 5) \
  "quo"(3.46, 0.5) &= #calc.quo(3.46, 0.5) $
```

$$\operatorname{quo}(a, b) = \left\lfloor \frac{a}{b} \right\rfloor$$
$$\operatorname{quo}(14, 5) = 2$$
$$\operatorname{quo}(3.46, 0.5) = 6$$

```
calc.quo(
  int float decimal,
  int float decimal,
) → int
```

/2.40.1 **dividend** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El dividendo del cociente.

/2.40.2 **divisor** `int` or `float` or `decimal` *Obligatorio* *Posicional*

El divisor del cociente.

/2.41 **norm** Desde: 0.13.0

Calcula la norma $p$ de una secuencia de valores.

La norma $p$ de $x_1, ..., x_n$ se define de la siguiente manera:

$$\begin{cases} \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} & \text{si } 0 < p < +\infty \\ \max_{i=1}^{n} |x_i| & \text{si } p = +\infty \end{cases}$$

```
#calc.norm(1, 2, -3, 0.5) \
#calc.norm(p: 3, 1, 2)
```

```
3.774917217635375
2.080083823051904
```

```
calc.norm(
  p: float,
  ..float,
) → float
```

/2.41.1 **p** `float` Por defecto: `2.0`

El valor de $p$. Debe ser mayor que cero.

El valor por defecto `2.0` corresponde a la norma euclidiana:

$$\sqrt{\sum_{i=1}^{n} x_i^2}$$

/2.41.2 **values** `float` *Obligatorio* *Posicional* *Variádico*

La secuencia de valores cuya norma $p$ se calcula. Devuelve `0.0` si está vacía.

### 3.5.10 `datetime` Desde: 0.5.0

Representa una fecha, una hora o una combinación de ambas.

Se puede crear especificando una fecha y hora personalizadas con la función constructora de este tipo u obteniendo la fecha actual con `datetime.today`[3.5.10/5.1].

#### /1 Ejemplo

```
#let date = datetime(
  year: 2020,
  month: 10,
  day: 4,
)

#date.display() \
#date.display(
  "y:[year repr:last_two]"
)

#let time = datetime(
  hour: 18,
  minute: 2,
  second: 23,
)

#time.display() \
#time.display(
  "h:[hour repr:12][period]"
)
```

2020-10-04
y:20

18:02:23
h:06PM

#### /2 Datetime y Duration

Podés obtener una duración[3.5.13] restando dos datetime:

```
#let first-of-march = datetime(day: 1, month: 3, year: 2024)
#let first-of-jan = datetime(day: 1, month: 1, year: 2024)
#let distance = first-of-march - first-of-jan
#distance.hours()
```

1440

También podés sumar o restar una duración a un datetime para obtener un nuevo datetime desplazado:

```
#let date = datetime(day: 1, month: 3, year: 2024)
#let two-days = duration(days: 2)
#let two-days-earlier = date - two-days
#let two-days-later = date + two-days

#date.display() \
#two-days-earlier.display() \
#two-days-later.display()
```

```
2024-03-01
2024-02-28
2024-03-03
```

## /3 Formato

Podés especificar un formato personalizado con el método display[3.5.10/5.2]. El formato de un datetime se especifica proporcionando *componentes* con una cantidad determinada de *modificadores*. Un componente representa una cierta parte del datetime que querés mostrar y, con la ayuda de los modificadores, podés definir cómo querés mostrar ese componente. Para mostrar un componente, encerrás el nombre del componente entre corchetes (por ejemplo, `[year]` mostrará el año). Para agregar modificadores, agregás un espacio después del nombre del componente seguido del nombre del modificador, dos puntos y el valor del modificador (por ejemplo, `[month repr:short]` mostrará la representación corta del mes).

La combinación posible de componentes y sus respectivos modificadores es la siguiente:

- `year`: Muestra el año del datetime.
  - `base`: Puede ser `calendar` o `iso_week`. Especifica si el año se basa en el calendario gregoriano o en el número de semana ISO.
  - `padding`: Puede ser `zero`, `space` o `none`. Especifica cómo se rellena el año.
  - `repr` Puede ser `full`, en cuyo caso se muestra el año completo, o `last_two`, en cuyo caso solo se muestran los dos últimos dígitos.
  - `sign`: Puede ser `automatic` o `mandatory`. Especifica cuándo se debe mostrar el signo.
- `month`: Muestra el mes del datetime.
  - `padding`: Puede ser `zero`, `space` o `none`. Especifica cómo se rellena el mes.
  - `repr`: Puede ser `numerical`, `long` o `short`. Especifica si el mes se debe mostrar como número o como palabra. Lamentablemente, al elegir la representación con palabras, actualmente solo puede mostrar la versión en inglés. Está previsto admitir la localización en el futuro.
- `day`: Muestra el día del datetime.
  - `padding`: Puede ser `zero`, `space` o `none`. Especifica cómo se rellena el día.
- `week_number`: Muestra el número de semana del datetime.
  - `padding`: Puede ser `zero`, `space` o `none`. Especifica cómo se rellena el número de semana.
  - `repr`: Puede ser `ISO`, `sunday` o `monday`. En el caso de `ISO`, los números de semana van de 1 a 53, mientras que los otros van de 0 a 53.
- `weekday`: Muestra el día de la semana de la fecha.
  - `repr` Puede ser `long`, `short`, `sunday` o `monday`. En el caso de `long` y `short`, se mostrará el nombre correspondiente en inglés (igual que para el mes; por ahora no se admiten otros idiomas). En el caso de `sunday` y `monday`, se mostrará el valor numérico (suponiendo el domingo y el lunes, respectivamente, como primer día de la semana).
  - `one_indexed`: Puede ser `true` o `false`. Define si la representación numérica de la semana empieza en 0 o en 1.
- `hour`: Muestra la hora de la fecha.
  - `padding`: Puede ser `zero`, `space` o `none`. Especifica cómo se rellena la hora.
  - `repr`: Puede ser `24` o `12`. Cambia si la hora se muestra en formato de 24 o de 12 horas.
- `period`: La parte AM/PM de la hora
  - `case`: Puede ser `lower` para mostrarla en minúsculas y `upper` para mostrarla en mayúsculas.
- `minute`: Muestra el minuto de la fecha.
  - `padding`: Puede ser `zero`, `space` o `none`. Especifica cómo se rellena el minuto.
- `second`: Muestra el segundo de la fecha.
  - `padding`: Puede ser `zero`, `space` o `none`. Especifica cómo se rellena el segundo.

Mirá acá[A] para más detalles sobre la sintaxis admitida.

[A] https://time-rs.github.io/book/api/format-description.html#components

Tené en cuenta que no siempre se pueden usar todos los componentes. Por ejemplo, si creás un nuevo datetime con `datetime(year: 2023, month: 10, day: 13)`, se almacenará internamente como una fecha simple, lo que significa que no podés usar componentes como `hour` o `minute`, que solo funcionarían en datetimes que tienen una hora especificada.

## /4 Constructor

Crea un datetime nuevo.

Podés especificar el datetime[3.5.10] con año, mes, día, hora, minuto y segundo.

*Nota*: Según qué componentes del datetime especifiques, Typst lo almacenará de una de las tres maneras siguientes:
- Si especificás año, mes y día, Typst almacenará solo una fecha.
- Si especificás hora, minuto y segundo, Typst almacenará solo una hora.
- Si especificás todos los componentes (año, mes, día, hora, minuto y segundo), Typst almacenará un datetime completo.

Según cómo se almacene, el método `display`[3.5.10/5.2] elegirá por defecto un formato distinto.

```
#datetime(
  year: 2012,
  month: 8,
  day: 3,
).display()
```

2012-08-03

```
datetime(
  year: int,
  month: int,
  day: int,
  hour: int,
  minute: int,
  second: int,
) → datetime
```

### /4.1 year int

El año del datetime.

### /4.2 month int

El mes del datetime.

### /4.3 day int

El día del datetime.

### /4.4 hour int

La hora del datetime.

### /4.5 minute int

El minuto del datetime.

### /4.6 second int

El segundo del datetime.

## /5 Funciones

### /5.1 today Desde: 0.5.0

Devuelve la fecha actual.

En la CLI, esto se puede reemplazar con el argumento `--creation-timestamp` o estableciendo la variable de entorno SOURCE_DATE_EPOCH[A]. En ambos casos, el valor debe darse como una marca de tiempo UNIX.

```
Today's date is
#datetime.today().display().
```

Today's date is 1970-01-01.

```
datetime.today(offset: auto int duration) → datetime
```

/5.1.1 **offset** `auto` or `int` or `duration` Por defecto: `auto`

Un desfase que se aplica a la fecha UTC actual. Si se establece en `auto`, el desfase será el local.

Cuando se da un desfase entero, se trata como una duración en horas.

/5.2 **display** Desde: 0.5.0

Muestra el datetime en un formato especificado.

Según hayas definido solo una fecha, solo una hora o ambas, el formato por defecto será distinto. Si especificaste una fecha, será `[year]-[month]-[day]`. Si especificaste una hora, será `[hour]:[minute]:[second]`. En el caso de un datetime, será `[year]-[month]-[day] [hour]:[minute]:[second]`.

Mirá la sintaxis de formato[3.5.10/3] para más información.

```
self.display(auto str) → str
```

/5.2.1 **pattern** `auto` or `str` *Posicional* Por defecto: `auto`

El formato que se usa para mostrar el datetime.

/5.3 **year** Desde: 0.5.0

El año si se especificó, o none para horas sin fecha.

```
self.year() → none int
```

/5.4 **month** Desde: 0.5.0

El mes si se especificó, o none para horas sin fecha.

```
self.month() → none int
```

/5.5 **weekday** Desde: 0.5.0

El día de la semana (contando el lunes como 1) o none para horas sin fecha.

```
self.weekday() → none int
```

/5.6 **day** Desde: 0.5.0

El día si se especificó, o none para horas sin fecha.

```
self.day() → none int
```

[A] https://reproducible-builds.org/specs/source-date-epoch/

/5.7 **hour** Desde: 0.5.0

La hora si se especificó, o none para fechas sin hora.

```
self.hour() → none int
```

/5.8 **minute** Desde: 0.5.0

El minuto si se especificó, o none para fechas sin hora.

```
self.minute() → none int
```

/5.9 **second** Desde: 0.5.0

El segundo si se especificó, o none para fechas sin hora.

```
self.second() → none int
```

/5.10 **ordinal** Desde: 0.5.0

El ordinal (día del año), o none para horas sin fecha.

```
self.ordinal() → none int
```

### 3.5.11 `decimal` Desde: 0.12.0

Un tipo de número decimal de punto fijo.

Este tipo debe usarse para operaciones aritméticas precisas con números representados en base 10. Un caso de uso típico es representar moneda.

#### /1 Ejemplo

```
Decimal: #(decimal("0.1") + decimal("0.2")) \
Float: #(0.1 + 0.2)
```

Decimal: 0.3
Float: 0.30000000000000004

#### /2 Construcción y conversiones

Para crear un número decimal, usá el constructor `decimal(string)`, como en `decimal("3.141592653")` *(fijate en las comillas dobles)*. Este constructor conserva todos los dígitos fraccionarios dados, siempre que sean representables según los límites que se especifican más abajo (de lo contrario, se produce un error).

También podés convertir cualquier entero[3.5.17] en decimal con el constructor `decimal(int)`, por ejemplo `decimal(59)`. Sin embargo, tené en cuenta que construir un decimal a partir de un número de punto flotante[3.5.15], si bien se admite, **es una conversión imprecisa y, por lo tanto, no se recomienda.** Se producirá una advertencia si Typst detecta que hubo una conversión accidental de `float` a `decimal` mediante su constructor, por ejemplo al escribir `decimal(3.14)` (fijate en la ausencia de comillas dobles, que indica que es una conversión accidental desde `float` y, por lo tanto, imprecisa). Se recomienda usar cadenas de texto para los valores decimales constantes (por ejemplo, `decimal("3.14")`).

La precisión de una conversión de `float` a `decimal` se puede mejorar un poco redondeando el resultado a 15 dígitos con `calc.round`[3.5.9/2.31], pero aun así no hay garantías de precisión para ese tipo de conversión.

#### /3 Operaciones

Se admiten las operaciones aritméticas básicas entre dos decimales y entre pares de decimales y enteros.

Las operaciones incorporadas entre `float` y `decimal` no se admiten, para evitar pérdidas accidentales de precisión. En su lugar, producirán un error.

Ciertas funciones de `calc`, como las trigonométricas y la potencia entre dos números reales, también se admiten solo para `float` (aunque se admite elevar un `decimal` a exponentes enteros). Podés optar por operaciones potencialmente imprecisas con el constructor `float(decimal)`, que convierte el número `decimal` en un `float` y permite realizar operaciones sin garantías de precisión.

#### /4 Mostrar decimales

Para mostrar un decimal, simplemente insertá el valor en el documento. Para mostrar solo una cierta cantidad de dígitos, primero redondeá[3.5.9/2.31] el decimal. El formato localizado de decimales y otros números todavía no se admite, pero está previsto para el futuro.

Podés convertir decimales en cadenas de texto con el constructor de `str`[3.5.28]. De esta manera, podés posprocesar la representación mostrada, por ejemplo, para reemplazar el punto por una coma (como sustituto de una localización incorporada adecuada para los idiomas que usan la coma).

#### /5 Precisión y límites

Un número `decimal` tiene un límite de 28 a 29 dígitos significativos en base 10. Esto incluye la suma de los dígitos antes y después del punto decimal. Por lo tanto, los números con más dígitos fraccionarios tienen un rango menor. Los números `decimal` máximo y mínimo tienen un valor de 79228162514264337593543950335

y -79228162514264337593543950335, respectivamente. A diferencia de float[3.5.15], este tipo no admite infinito ni NaN, así que las operaciones que desbordan por exceso o por defecto producirán un error.

Las operaciones típicas entre números decimal, como la suma, la multiplicación y la potencia[3.5.9/2.2] con exponente entero, serán muy precisas gracias a su representación de punto fijo. Sin embargo, tené en cuenta que la multiplicación y la división podrían no conservar todos los dígitos en algunos casos extremos: si bien se consideran precisas, los dígitos que superan los límites especificados arriba se redondean y se pierden, por lo que es posible cierta pérdida de precisión más allá de la cantidad máxima de dígitos representables. Tené en cuenta que este comportamiento se puede observar no solo al dividir, sino también al multiplicar por números entre 0 y 1, ya que ambas operaciones pueden llevar los dígitos fraccionarios de un número más allá de los límites descritos arriba, lo que produce redondeo. Cuando esas dos operaciones no superan los límites de dígitos, son totalmente precisas.

## /6 Constructor

Convierte un valor en un decimal.

Se recomienda usar una cadena de texto para construir el número decimal, o un entero[3.5.17] (si se desea). La cadena debe contener un número con el formato "3.14159" (o "-3.141519" para números negativos). Los dígitos fraccionarios se conservan por completo; si eso no es posible porque se alcanzó el límite de dígitos significativos (alrededor de 28 a 29), se produce un error, ya que el número decimal dado no sería representable.

Si bien este constructor se puede usar con números de punto flotante[3.5.15] para convertirlos en decimal, hacerlo **no se recomienda**, porque **esta conversión es intrínsecamente imprecisa.** Es fácil realizar esta conversión por accidente al escribir decimal(1.234) (fijate en la ausencia de comillas dobles), por lo que Typst emitirá una advertencia en ese caso. Escribí en su lugar decimal("1.234") para ese caso particular (inicialización de un decimal constante). Tené en cuenta también que los flotantes que son NaN o infinitos no se pueden convertir en decimales y producirán un error.

```
#decimal("1.222222222222222")
```

1.222222222222222

```
decimal(bool int float decimal str) → decimal
```

### /6.1 value bool or int or float or decimal or str *Obligatorio* *Posicional*

El valor que se debe convertir en decimal.

### 3.5.12 `dictionary` Desde: siempre

Un mapa de claves de tipo cadena de texto a valores.

Podés construir un diccionario encerrando entre paréntesis pares `key: value` separados por comas. Los valores no tienen por qué ser del mismo tipo. Como los paréntesis vacíos ya producen un array vacío, tenés que usar la sintaxis especial `(:)` para crear un diccionario vacío.

Un diccionario es conceptualmente similar a un array[3.5.2], pero se indexa por cadenas de texto en lugar de enteros. Podés acceder a las entradas de un diccionario y crearlas con el método `.at()`. Si conocés la clave de forma estática, alternativamente podés usar la notación de acceso a campos[3.3.6] (`.key`) para acceder al valor. Para comprobar si una clave está presente en el diccionario, usá la palabra clave `in`.

Podés iterar sobre los pares de un diccionario con un bucle for[3.3.5]. Esto itera en el orden en que se insertaron o declararon inicialmente los pares.

Los diccionarios se pueden sumar con el operador + y unir[3.3.2]. También se pueden expandir[3.5.1/2] en una llamada a función o en otro diccionario con el operador `..spread`[A]. En cada caso, si una clave aparece varias veces, el último valor sobrescribe a los demás.

#### /1 Ejemplo

```
#let dict = (
  name: "Typst",
  born: 2019,
)

#dict.name \
#(dict.launch = 20)
#dict.len() \
#dict.keys() \
#dict.values() \
#dict.at("born") \
#dict.insert("city", "Berlin")
#("name" in dict)
```

```
Typst
3
("name", "born", "launch")
("Typst", 2019, 20)
2019
true
```

#### /2 Constructor

Convierte un valor en un diccionario.

Tené en cuenta que esta función solo está pensada para convertir en diccionario un valor similar a un diccionario, no para crear un diccionario a partir de pares individuales. Para eso, usá la sintaxis de diccionario `(key: value)`. Mirá también `array.to-dict`[3.5.2/3.32] para convertir arrays en diccionarios.

```
#dictionary(sys).at("version")
```

```
0.15.1
```

```
dictionary(module) → dictionary
```

##### /2.1 `value` `module` *Obligatorio* *Posicional*

El valor que se debe convertir en diccionario.

#### /3 Funciones

##### /3.1 `len` Desde: siempre

La cantidad de pares del diccionario.

[A] Al expandir en un diccionario, si todos los elementos entre paréntesis se expanden, tenés que empezar el contenedor con `(:`, como en `(: ..dict, ..other_dict)`. De lo contrario, se infiere que el contenedor es un array y se produce un error.

```
self.len() → int
```

### /3.2 at Desde: siempre

Devuelve el valor asociado a la clave especificada en el diccionario.

Se puede usar en el lado izquierdo de una asignación si la clave ya está presente en el diccionario. Devuelve el valor por defecto si la clave no forma parte del diccionario, o falla con un error si no se especificó ningún valor por defecto.

También se puede acceder a los valores con la sintaxis de campos (por ejemplo, `(key: 42).key`) si no hace falta un valor por defecto.

```
self.at(
  str,
  default: any,
) → any
```

#### /3.2.1 key str *Obligatorio* *Posicional*

La clave del elemento que se quiere obtener.

#### /3.2.2 default any

Un valor por defecto para devolver si la clave no forma parte del diccionario.

### /3.3 insert Desde: siempre

Inserta un par nuevo en el diccionario. Si el diccionario ya contiene esta clave, se actualiza el valor.

Para insertar varios pares a la vez, alternativamente podés sumar otro diccionario con el operador +=.

```
self.insert(
  str,
  any,
) → none
```

#### /3.3.1 key str *Obligatorio* *Posicional*

La clave del par que se debe insertar.

#### /3.3.2 value any *Obligatorio* *Posicional*

El valor del par que se debe insertar.

### /3.4 remove Desde: siempre

Elimina un par del diccionario según su clave y devuelve el valor.

```
self.remove(
  str,
  default: any,
) → any
```

#### /3.4.1 key str *Obligatorio* *Posicional*

La clave del par que se quiere eliminar.

#### /3.4.2 default any

Un valor por defecto para devolver si la clave no existe.

### /3.5 keys Desde: siempre

Devuelve las claves del diccionario como un array, en orden de inserción.

```
self.keys() → array
```

### /3.6 values Desde: siempre

Devuelve los valores del diccionario como un array, en orden de inserción.

```
self.values() → array
```

### /3.7 pairs Desde: siempre

Devuelve las claves y los valores del diccionario como un array de pares. Cada par se representa como un array de longitud dos.

```
self.pairs() → array
```

### /3.8 filter Desde: 0.15.0

Produce un diccionario nuevo solo con los pares del original para los cuales la función indicada devuelve `true`.

Example · Uso básico

```
#{
  (a: 0, b: 1, c: 2)
    .filter(v => v > 0)
}
```

```
(b: 1, c: 2)
```

Example · Filtrado según la clave en lugar del valor

```
#{
  (a: 0, b: 1, c: 2)
    .pairs()
    .filter(((k, v)) => k != "a")
    .to-dict()
}
```

```
(b: 1, c: 2)
```

```
self.filter(function) → dictionary
```

#### /3.8.1 test function *Obligatorio* *Posicional*

La función que se aplica a cada valor. Debe devolver un booleano.

### /3.9 map Desde: 0.15.0

Produce un diccionario nuevo en el que las claves son las mismas, pero los valores se transforman con la función indicada.

```
#(a: 0, b: 1, c: 2).map(v => v + 1)
```

```
(a: 1, b: 2, c: 3)
```

```
self.map(function) → dictionary
```

#### /3.9.1 mapper function *Obligatorio* *Posicional*

La función que se aplica a cada valor.

### 3.5.13 `duration` Desde: 0.8.0

Representa un lapso de tiempo positivo o negativo.

#### /1 Constructor

Crea una duración nueva.

Podés especificar la duración[3.5.13] con semanas, días, horas, minutos y segundos. También podés obtener una duración restando dos fechas y horas[3.5.10].

```
#duration(
  days: 3,
  hours: 12,
).hours()
```

```
84
```

```
duration(
  seconds: int,
  minutes: int,
  hours: int,
  days: int,
  weeks: int,
) → duration
```

##### /1.1 **seconds** `int` Por defecto: `0`

La cantidad de segundos.

##### /1.2 **minutes** `int` Por defecto: `0`

La cantidad de minutos.

##### /1.3 **hours** `int` Por defecto: `0`

La cantidad de horas.

##### /1.4 **days** `int` Por defecto: `0`

La cantidad de días.

##### /1.5 **weeks** `int` Por defecto: `0`

La cantidad de semanas.

#### /2 Funciones

##### /2.1 **seconds** Desde: 0.8.0

La duración expresada en segundos.

Esta función devuelve la duración total representada en segundos como un número de punto flotante, en lugar del componente de segundos de la duración.

```
self.seconds() → float
```

##### /2.2 **minutes** Desde: 0.8.0

La duración expresada en minutos.

Esta función devuelve la duración total representada en minutos como un número de punto flotante, en lugar del componente de minutos de la duración.

```
self.minutes() → float
```

### /2.3 hours Desde: 0.8.0

La duración expresada en horas.

Esta función devuelve la duración total representada en horas como un número de punto flotante, en lugar del componente de horas de la duración.

```
self.hours() → float
```

### /2.4 days Desde: 0.8.0

La duración expresada en días.

Esta función devuelve la duración total representada en días como un número de punto flotante, en lugar del componente de días de la duración.

```
self.days() → float
```

### /2.5 weeks Desde: 0.8.0

La duración expresada en semanas.

Esta función devuelve la duración total representada en semanas como un número de punto flotante, en lugar del componente de semanas de la duración.

```
self.weeks() → float
```

### 3.5.14 `eval` Desde: siempre

Evalúa una cadena de texto como código Typst.

Esta función solo debería usarse como último recurso.

#### /1 Ejemplo

```
#eval("1 + 1") \
#eval("(1, 2, 3, 4)").len() \
#eval("*Markup!*", mode: "markup") \
```

2
4
**Markup!**

#### /2 Parámetros

```
eval(
  str,
  mode: str,
  scope: dictionary,
) → any
```

/2.1 **source** `str` *Obligatorio* *Posicional*

Una cadena de código Typst para evaluar.

/2.2 **mode** `str` Por defecto: `"code"`

El modo sintáctico[3.1.1] en el que se analiza la cadena de texto.

```
#eval("= Heading", mode: "markup")
#eval("1_2^3", mode: "math")
```

**Heading**
$1_2^3$

| Variante | Detalles |
|---|---|
| `"markup"` | Se evalúa como marcado, como en un archivo de Typst. |
| `"math"` | Se evalúa como matemática, como en una ecuación. |
| `"code"` | Se evalúa como código, como después de un numeral. |

/2.3 **scope** `dictionary` Por defecto: `(:)`

Un ámbito de definiciones que se ponen a disposición.

```
#eval("x + 1", scope: (x: 2)) \
#eval(
  "abc/xyz",
  mode: "math",
  scope: (
    abc: $a + b + c$,
    xyz: $x + y + z$,
  ),
)
```

3
$\frac{a+b+c}{x+y+z}$

### 3.5.15 `float` Desde: siempre

Un número de punto flotante.

Una representación de precisión limitada de un número real. Typst usa 64 bits para almacenar los flotantes. Dondequiera que se espere un flotante, también podés pasar un entero[3.5.17].

Podés convertir un valor en flotante con el constructor de este tipo.

#### /1 Ejemplo

```
#3.14 \
#1e4 \
#(10 / 4)
```

```
3.14
10000
2.5
```

#### /2 Constructor

Convierte un valor en un flotante.

- Los booleanos se convierten en `0.0` o `1.0`.
- Los enteros se convierten en el flotante de 64 bits más cercano. Para enteros con valor absoluto menor que `calc.pow(2, 53)`, esta conversión es exacta.
- Las proporciones se dividen por 100%.
- Las cadenas de texto se analizan en base 10 al flotante de 64 bits más cercano. Se admite la notación exponencial.

```
#float(false) \
#float(true) \
#float(4) \
#float(40%) \
#float("2.7") \
#float("1e5")
```

```
0
1
4
0.4
2.7
100000
```

`float(` `bool` `int` `float` `ratio` `decimal` `str` `) →` `float`

##### /2.1 **`value`** `bool` or `int` or `float` or `ratio` or `decimal` or `str` *Obligatorio* *Posicional*

El valor que se debe convertir en flotante.

#### /3 Constantes

##### /3.1 **`inf`** `float` Desde: 0.12.0

Infinito ($+\infty$) como valor.

Este valor es mayor que todo número real.

El infinito negativo ($-\infty$) se puede obtener negando este valor: `-float.inf`.

```
#(9999999 < float.inf) \
#(-float.inf < -9999999)
```

```
true
true
```

### /3.2 nan `float` Desde: 0.12.0

Un valor NaN, tal como lo define el estándar IEEE 754[A].

## /4 Funciones

### /4.1 is-nan Desde: 0.11.0

Comprueba si un flotante no es un número.

En IEEE 754, más de un patrón de bits representa un NaN. Esta función devuelve `true` si el flotante es cualquiera de esos patrones de bits.

```
#float.is-nan(0) \
#float.is-nan(1) \
#float.is-nan(float.nan)
```

```
false
false
true
```

```
self.is-nan() → bool
```

### /4.2 is-infinite Desde: 0.11.0

Comprueba si un flotante es infinito.

Los flotantes pueden representar el infinito positivo y el infinito negativo. Esta función devuelve `true` si el flotante es un infinito.

```
#float.is-infinite(0) \
#float.is-infinite(1) \
#float.is-infinite(float.inf)
```

```
false
false
true
```

```
self.is-infinite() → bool
```

### /4.3 signum Desde: 0.11.0

Calcula el signo de un número de punto flotante.

- Si el número es positivo (incluido `+0.0`), devuelve `1.0`.
- Si el número es negativo (incluido `-0.0`), devuelve `-1.0`.
- Si el número es NaN, devuelve `float.nan`[3.5.15/3.2].

```
#(5.0).signum() \
#(-5.0).signum() \
#(0.0).signum() \
#float.nan.signum()
```

```
1
−1
1
NaN
```

```
self.signum() → float
```

[A] https://en.wikipedia.org/wiki/IEEE_754

### /4.4 from-bytes Desde: 0.12.0

Interpreta bytes como un flotante.

```
#float.from-bytes(bytes((0, 0, 0, 0, 0, 0, 240, 63))) \
#float.from-bytes(bytes((63, 240, 0, 0, 0, 0, 0, 0)), endian: "big")
```

```
1
1
```

```
float.from-bytes(
  bytes,
  endian: str,
) → float
```

#### /4.4.1 bytes bytes *Obligatorio* *Posicional*

Los bytes que se deben convertir en flotante.

Deben tener una longitud de 4 u 8. Los bytes se interpretan entonces en el formato binary32 (precisión simple) o binary64 (precisión doble) de IEEE 754[A], según la longitud de los bytes.

#### /4.4.2 endian str

Por defecto: "little"

El orden de bytes (endianness) de la conversión.

| Variante | Detalles |
|---|---|
| "big" | Orden de bytes big-endian: el byte de mayor valor está al comienzo de los bytes. |
| "little" | Orden de bytes little-endian: el byte de menor valor está al comienzo de los bytes. |

### /4.5 to-bytes Desde: 0.12.0

Convierte un flotante en bytes.

```
#array(1.0.to-bytes(endian: "big")) \
#array(1.0.to-bytes())
```

```
(63, 240, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 240, 63)
```

```
self.to-bytes(
  endian: str,
  size: int,
) → bytes
```

#### /4.5.1 endian str

Por defecto: "little"

El orden de bytes (endianness) de la conversión.

| Variante | Detalles |
|---|---|
| "big" | Orden de bytes big-endian: el byte de mayor valor está al comienzo de los bytes. |
| "little" | Orden de bytes little-endian: el byte de menor valor está al comienzo de los bytes. |

#### /4.5.2 size int

Por defecto: 8

El tamaño de los bytes resultantes.

[A] https://en.wikipedia.org/wiki/IEEE_754

Debe ser 4 u 8. La llamada devuelve la representación de este flotante en el formato binary32 (precisión simple) o binary64 (precisión doble) de IEEE 754[A], según el tamaño indicado.

[A] https://en.wikipedia.org/wiki/IEEE_754

### 3.5.16 `function` Desde: siempre

Una correspondencia de valores de argumentos a un valor de retorno.

Podés llamar a una función escribiendo, directamente después del nombre de la función, una lista de *argumentos* separados por comas entre paréntesis. Además, podés pasarle a una función cualquier cantidad de argumentos finales de bloque de contenido *después* de la lista normal de argumentos. Si la lista normal de argumentos quedara vacía, se puede omitir. Typst admite argumentos posicionales y con nombre. Los primeros se identifican por posición y tipo, mientras que los segundos se escriben como `name: value`.

En modo matemático, las llamadas a funciones tienen un comportamiento especial. Mirá la documentación de matemática[3.8] para más detalles.

#### /1 Ejemplo

```
// Call a function.
#list([A], [B])

// Named arguments and trailing
// content blocks.
#enum(start: 2)[A][B]

// Version without parentheses.
#list[A][B]
```

- A
- B

2. A
3. B

- A
- B

Las funciones son un componente fundamental de Typst. Typst ofrece funciones para una variedad de tareas de composición. Además, el marcado que escribís está respaldado por funciones y todo el estilo se aplica a través de funciones. Esta referencia enumera todas las funciones disponibles y cómo podés usarlas. Mirá también la documentación sobre las reglas set[3.2.1] y show[3.2.2] para conocer otras formas de trabajar con funciones en Typst.

#### /2 Funciones de elemento

Algunas funciones están asociadas a *elementos* como los títulos[3.6.9] o las tablas[3.6.19]. Cuando se las llama, crean un elemento de su tipo respectivo. A diferencia de las funciones normales, además se pueden usar en reglas set[3.2.1], reglas show[3.2.2] y selectores[3.5.26].

#### /3 Ámbitos de funciones

Las funciones pueden contener definiciones relacionadas en su propio ámbito, de forma similar a un módulo[3.3.8]. Ejemplos de esto son `assert.eq`[3.5.3/3.1] o `list.item`[3.6.2/4.1]. Sin embargo, esta característica actualmente solo está disponible para funciones incorporadas.

#### /4 Definir funciones

Podés definir tu propia función con un enlace let[3.3.3] que tenga una lista de parámetros después del nombre del enlace. La lista de parámetros puede contener parámetros posicionales obligatorios, parámetros con nombre con valores por defecto y sumideros de argumentos[3.5.1].

El lado derecho de un enlace de función es el cuerpo de la función, que puede ser un bloque o cualquier otra expresión. Define el valor de retorno de la función y puede depender de los parámetros. Si el cuerpo de la función es un bloque de código[3.3.2], el valor de retorno es el resultado de unir los valores de cada expresión del bloque.

Dentro del cuerpo de una función, se puede usar la palabra clave `return` para salir anticipadamente y, opcionalmente, especificar un valor de retorno. Si no se da un valor de retorno explícito, el cuerpo se evalúa al resultado de unir todas las expresiones que preceden a `return`.

Las funciones que no devuelven ningún valor significativo devuelven none[3.5.20]. El tipo de retorno de esas funciones no se especifica explícitamente en la documentación. (Un ejemplo de esto es `array.push`[3.5.2/3.5]).

```
#let alert(body, fill: red) = {
  set text(white)
  set align(center)
  rect(
    fill: fill,
    inset: 8pt,
    radius: 4pt,
    [*Warning:\ #body*],
  )
}

#alert[
  Danger is imminent!
]

#alert(fill: blue)[
  KEEP OFF TRACKS
]
```




## /5 Importar funciones

Las funciones se pueden importar de un archivo (`module`[3.3.8]) a otro con `import`. Por ejemplo, supongamos que definimos la función `alert` del ejemplo anterior en un archivo llamado `foo.typ`. Podemos importarla en otro archivo escribiendo `import "foo.typ": alert`.

## /6 Funciones sin nombre

También podés crear una función sin nombre, sin crear un enlace, especificando una lista de parámetros seguida de => y el cuerpo de la función. Si tu función tiene un solo parámetro, los paréntesis alrededor de la lista de parámetros son opcionales. Las funciones sin nombre son útiles principalmente para las reglas show, pero también como argumentos de otras funciones, como `array.position`[3.5.2/3.12].

```
#show "once?": it => [#it #it]
once?
```

once? once?

## /7 Nota sobre la pureza de las funciones

En Typst, todas las funciones son *puras*. Esto significa que, para los mismos argumentos, siempre devuelven el mismo resultado. No pueden "recordar" cosas para producir otro valor cuando se las llama por segunda vez.

La única excepción son los métodos incorporados como `array.push(value)`[3.5.2/3.5]. Estos pueden modificar los valores sobre los que se los llama.

## /8 Funciones

### /8.1 `with` Desde: siempre

Devuelve una función nueva a la que se le aplicaron previamente los argumentos dados.

```
self.with(.. any ) → function
```

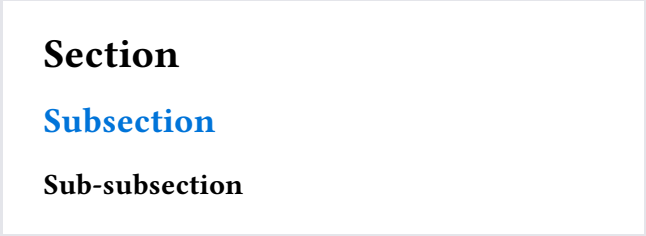
#### /8.1.1 `arguments` any *Obligatorio* *Posicional* *Variádico*

Los argumentos que se aplican a la función.

### /8.2 `where` Desde: siempre

Devuelve un selector que filtra los elementos que pertenecen a esta función y cuyos campos tienen los valores de los argumentos dados.

```
#show heading.where(level: 2): set text(blue)
= Section
== Subsection
=== Sub-subsection
```




```
self.where(.. any ) → selector
```

/8.2.1 **fields** any *Obligatorio* *Posicional* *Variádico*

Los campos por los que se filtra.

### 3.5.17 `int` Desde: siempre

Un entero: un número entero positivo, un número entero negativo o cero.

Typst almacena los enteros con signo con la representación de complemento a dos[A] en 64 bits. Esto permite almacenar números hasta $2^{63} - 1$ (`int.max`[3.5.17/4.1]) y hasta $-2^{63}$ (`int.min`[3.5.17/4.2]).

Los enteros también se pueden especificar en hexadecimal, octal o binario empezando con los prefijos: `0x`, `0o` o `0b`.

Podés convertir un valor en entero con el constructor de este tipo.

## /1 Ejemplo

```
#(1 + 2) \
#(2 - 5) \
#(3 + 4 < 8)

#0xff \
#0o10 \
#0b1001

#(int(3.8) + int("26"))

/ Max: #int.max
/ Min: #int.min
```

3
−3
true

255
8
9

29

**Max** 9223372036854775807
**Min** −9223372036854775808

## /2 Sintaxis

Los enteros de Typst se pueden ingresar en modo código con los dígitos decimales 0–9. Además, si un 0 solitario va seguido de x, o o b (`0x`, `0o`, `0b`), Typst tratará los dígitos siguientes como un número hexadecimal[B] (base 16), octal[C] (base 8) o binario[D] (base 2).

Los números hexadecimales usan las letras a–f o A–F para los valores 10–15.

Typst dará un error si se escribe un entero mayor que `int.max`[3.5.17/4.1]. Si esto pasa, quizás quieras usar en su lugar un número de punto flotante[3.5.15], agregando un punto al final del número.

Typst se diferencia de otros lenguajes de programación en que no trata los enteros negativos como tokens individuales en su sintaxis. En cambio, una entrada como `-6` se trata como el operador de negación aplicado al entero positivo 6. Esto puede causar un problema al intentar escribir el entero negativo mínimo `-9223372036854775808`, ya que `9223372036854775808` es mayor que `int.max`[3.5.17/4.1]. Para escribir el entero negativo mínimo, usá en su lugar `int.min`[3.5.17/4.2].

Esto también significa que, si querés incrustar un entero negativo en el marcado, vas a tener que usar paréntesis para agrupar el operador de negación: `#(-6)`.

## /3 Constructor

Convierte un valor en un entero. Produce un error si se intenta analizar una cadena de texto no válida o producir un entero que no entra en un entero con signo de 64 bits.

- Los booleanos se convierten en 0 o 1.
- Los flotantes y decimales se redondean hacia cero al siguiente entero de 64 bits.
- Las cadenas de texto se analizan por defecto en base 10.

[A] https://en.wikipedia.org/wiki/Two%27s_complement
[B] https://en.wikipedia.org/wiki/Hexadecimal
[C] https://en.wikipedia.org/wiki/Octal
[D] https://en.wikipedia.org/wiki/Binary_number

```
#int(false) \
#int(true) \
#int(2.7) \
#int(decimal("3.8")) \
#(int("27") + int("4")) \
#int("beef", base: 16)
```

0
1
2
3
31
48879

```
int(
  bool int float decimal str,
  base: int,
) → int
```

/3.1 **value** bool or int or float or decimal or str *Obligatorio* *Posicional*

El valor que se debe convertir en entero.

/3.2 **base** int Por defecto: 10

La base (raíz) para analizar cadenas de texto, entre 2 y 36.

Por encima de la base 10, Typst acepta las letras a–z o A–Z para los valores 10–35.

## /4 Constantes

### /4.1 max int Desde: siempre

El valor máximo que se puede representar como entero de Typst: $2^{63} - 1$.

```
#int.max \
// In hexadecimal:
#0x7FFFFFFFFFFFFFFF
```

9223372036854775807
9223372036854775807

### /4.2 min int Desde: siempre

El valor mínimo que se puede representar como entero de Typst: $-2^{63}$.

Intentar escribir este entero explícitamente como `-9223372036854775808` en lugar de usar `int.min` producirá un error. Para más información, leé la sección Sintaxis[3.5.17/2].

```
#int.min
```

−9223372036854775808

## /5 Funciones

### /5.1 signum Desde: 0.11.0

Calcula el signo de un entero.

- Si el número es positivo, devuelve `1`.
- Si el número es negativo, devuelve `-1`.
- Si el número es cero, devuelve `0`.

```
#(5).signum() \
#(-5).signum() \
#(0).signum()
```

```
1
−1
0
```

```
self.signum() → int
```

### /5.2 bit-not Desde: 0.11.0

Calcula el NOT bit a bit de un entero.

A los fines de esta función, el operando se trata como un entero con signo de 64 bits.

```
#4.bit-not() \
#(-1).bit-not()
```

```
−5
0
```

```
self.bit-not() → int
```

### /5.3 bit-and Desde: 0.11.0

Calcula el AND bit a bit entre dos enteros.

A los fines de esta función, los operandos se tratan como enteros con signo de 64 bits.

```
#128.bit-and(192)
```

```
128
```

```
self.bit-and(int) → int
```

#### /5.3.1 rhs int *Obligatorio* *Posicional*

El operando derecho del AND bit a bit.

### /5.4 bit-or Desde: 0.11.0

Calcula el OR bit a bit entre dos enteros.

A los fines de esta función, los operandos se tratan como enteros con signo de 64 bits.

```
#64.bit-or(32)
```

```
96
```

```
self.bit-or(int) → int
```

#### /5.4.1 rhs int *Obligatorio* *Posicional*

El operando derecho del OR bit a bit.

### /5.5 bit-xor Desde: 0.11.0

Calcula el XOR bit a bit entre dos enteros.

A los fines de esta función, los operandos se tratan como enteros con signo de 64 bits.

```
#64.bit-xor(96)
```

```
32
```

```
self.bit-xor( int ) → int
```

/5.5.1 **rhs** `int` *Obligatorio* *Posicional*

El operando derecho del XOR bit a bit.

## /5.6 bit-lshift Desde: 0.11.0

Desplaza los bits del operando a la izquierda la cantidad especificada.

A los fines de esta función, el operando se trata como un entero con signo de 64 bits. Se producirá un error si el resultado es demasiado grande para entrar en un entero de 64 bits.

```
#33.bit-lshift(2) \
#(-1).bit-lshift(3)
```

```
132
−8
```

```
self.bit-lshift( int ) → int
```

/5.6.1 **shift** `int` *Obligatorio* *Posicional*

La cantidad de bits que se desplazan. No debe ser negativa.

## /5.7 bit-rshift Desde: 0.11.0

Desplaza los bits del operando a la derecha la cantidad especificada. Por defecto realiza un desplazamiento aritmético (extiende el bit de signo hacia la izquierda, de modo que los números negativos siguen siendo negativos), pero esto se puede cambiar con el parámetro `logical`.

A los fines de esta función, el operando se trata como un entero con signo de 64 bits.

```
#64.bit-rshift(2) \
#(-8).bit-rshift(2) \
#(-8).bit-rshift(2, logical: true)
```

```
16
−2
4611686018427387902
```

```
self.bit-rshift(
  int ,
  logical: bool ,
) → int
```

/5.7.1 **shift** `int` *Obligatorio* *Posicional*

La cantidad de bits que se desplazan. No debe ser negativa.

Se permiten desplazamientos mayores que 63 y harán que el valor de retorno se sature. Para números no negativos, el valor de retorno se satura en `0`, mientras que, para números negativos, se satura en `-1` si `logical` es `false`, o en `0` si es `true`. Este comportamiento es coherente con aplicar esta operación varias veces. Por lo tanto, el desplazamiento siempre tendrá éxito.

/5.7.2 **logical** `bool` Por defecto: `false`

Activa o desactiva que se realice un desplazamiento lógico (sin signo) a la derecha en lugar de uno aritmético. Si es `true`, los operandos negativos no conservarán su bit de signo y los bits que aparezcan

a la izquierda después del desplazamiento serán 0. Este parámetro no tiene efecto en operandos no negativos.

### /5.8 from-bytes Desde: 0.12.0

Convierte bytes en un entero.

```
#int.from-bytes(bytes((0, 0, 0, 0, 0, 0, 0, 1))) \
#int.from-bytes(bytes((1, 0, 0, 0, 0, 0, 0, 0)), endian: "big")
```

```
72057594037927936
72057594037927936
```

```
int.from-bytes(
  bytes,
  endian: str,
  signed: bool,
) → int
```

#### /5.8.1 bytes bytes *Obligatorio* *Posicional*

Los bytes que se deben convertir en entero.

Deben tener una longitud máxima de 8 para que el resultado entre en un entero con signo de 64 bits.

#### /5.8.2 endian str Por defecto: "little"

El orden de bytes (endianness) de la conversión.

| Variante | Detalles |
|---|---|
| "big" | Orden de bytes big-endian: el byte de mayor valor está al comienzo de los bytes. |
| "little" | Orden de bytes little-endian: el byte de menor valor está al comienzo de los bytes. |

#### /5.8.3 signed bool Por defecto: true

Si los bytes se deben tratar como un entero con signo. Si es true y el bit más significativo está activado, el número resultante será negativo.

### /5.9 to-bytes Desde: 0.12.0

Convierte un entero en bytes.

```
#array(10000.to-bytes(endian: "big")) \
#array(10000.to-bytes(size: 4))
```

```
(0, 0, 0, 0, 0, 0, 39, 16)
(16, 39, 0, 0)
```

```
self.to-bytes(
  endian: str,
  size: int,
) → bytes
```

#### /5.9.1 endian str Por defecto: "little"

El orden de bytes (endianness) de la conversión.

| Variante | Detalles |
|---|---|
| `"big"` | Orden de bytes big-endian: el byte de mayor valor está al comienzo de los bytes. |
| `"little"` | Orden de bytes little-endian: el byte de menor valor está al comienzo de los bytes. |

/5.9.2 **`size`** `int` Por defecto: 8

El tamaño en bytes de los bytes resultantes (debe ser al menos cero). Si el entero es demasiado grande para entrar en el tamaño especificado, la conversión truncará los bytes restantes según el orden de bytes. Para mantener el mismo valor resultante, si el orden es big-endian, el truncamiento se hará en los bytes de más a la derecha. En cambio, si el orden es little-endian, el truncamiento se hará en los bytes de más a la izquierda.

Tené en cuenta que, si el entero es negativo y el tamaño no alcanza para que el número entre, al pasar los bytes resultantes a `int.from_bytes`, el número resultante podría ser positivo, ya que el bit más significativo podría no estar en

1.

### 3.5.18 `label` Desde: siempre

Una etiqueta para un elemento.

Al insertar una etiqueta en el contenido, esta se adjunta al elemento anterior más cercano que no sea un espacio. El elemento anterior debe estar en el mismo ámbito que la etiqueta, lo que significa que, por ejemplo, `Hello #[<label>]` no funcionaría.

Un elemento etiquetado se puede referenciar[3.6.17], consultar[3.12.6] y estilizar[3.2] a través de su etiqueta.

Una vez construida, podés obtener el nombre de una etiqueta usando `str`[3.5.28/3].

#### /1 Ejemplo

```
#show <a>: set text(blue)
#show label("b"): set text(red)

= Heading <a>
*Strong* #label("b")
```

Heading
Strong

#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: podés crear una etiqueta encerrando su nombre entre paréntesis angulares. Esto funciona tanto en marcado como en código. El nombre de una etiqueta puede contener letras, números, _, -, : y .. Una etiqueta no puede estar vacía.

Tené en cuenta que hay una diferencia sintáctica al usar la sintaxis dedicada de esta función. En el siguiente código, el `<a>` termina el título y, por lo tanto, se adjunta al título en sí, mientras que el `#label("b")` es parte del título y, por lo tanto, se adjunta al texto del título.

```
// Equivalent to `#heading[Introduction] <a>`.
= Introduction <a>

// Equivalent to `#heading[Conclusion #label("b")]`.
= Conclusion #label("b")
```

Actualmente, las etiquetas solo se pueden adjuntar a elementos en modo marcado, no en modo código. Esto podría cambiar en el futuro.

#### /3 Constructor

Crea una etiqueta a partir de una cadena de texto.

```
label(str) → label
```

##### /3.1 **name** `str` *Obligatorio* *Posicional*

El nombre de la etiqueta.

A diferencia de la sintaxis dedicada[3.5.18/2], este constructor acepta cualquier cadena de texto no vacía, incluidos los nombres con caracteres especiales.

### 3.5.19 `module` Desde: siempre

Una colección de variables y funciones que suelen estar relacionadas con un único tema.

Un módulo puede
- ser incorporado
- provenir de una importación de archivo[3.3.8]
- provenir de una importación de paquete[3.3.9] (y, por lo tanto, indirectamente de su archivo de punto de entrada)
- ser el resultado de una llamada a la función plugin[3.5.23]

Podés acceder a las definiciones del módulo con la notación de acceso a campos[3.3.6] e interactuar con él usando las sintaxis de importación e inclusión[3.3.8].

```
#import "utils.typ"
#utils.add(2, 5)

#import utils: sub
#sub(1, 4)
```

```
7
−3
```

Podés comprobar si una definición está presente en un módulo usando el operador `in`, con una cadena de texto a la izquierda. Esto puede ser útil para acceder condicionalmente[3.5.5/2] a definiciones de un módulo.

```
#("table" in std) \
#("nope" in std)
```

```
true
false
```

Como alternativa, se puede convertir un módulo en un diccionario y, de esa manera, acceder a su contenido de forma dinámica, usando el constructor de diccionarios[3.5.12/2].

### 3.5.20 none

Un valor que indica la ausencia de cualquier otro valor.

El tipo none tiene exactamente un valor: none.

Cuando se lo inserta en el documento, no es visible. Este es también el valor que producen los bloques de código vacíos. Se puede unir[3.3.2] con cualquier valor, lo que da como resultado el otro valor.

#### /1 Ejemplo

```
Not visible: #none
```

Not visible:

### 3.5.21 `panic` Desde: siempre

Falla con un error.

Los argumentos se muestran al usuario (no se renderizan en el documento) como cadenas de texto, convirtiéndolos con `repr` si es necesario.

#### /1 Ejemplo

El siguiente código produce el error `panicked with: this is wrong`.

```
#panic("this is wrong")
```

#### /2 Parámetros

```
panic(..any)
```

/2.1 **`values`** `any` *Obligatorio* *Posicional* *Variádico*

Los valores con los que se entra en pánico y que se muestran al usuario.

### 3.5.22 `path` Desde: 0.15.0

Una ruta del sistema de archivos.

Cuando dividís tu proyecto o paquete en varios archivos, o hacés referencia a recursos como imágenes o bibliografías, necesitás interactuar con *rutas*.

## /1 Cadenas de ruta

Comúnmente, las rutas se expresan simplemente como cadenas de texto[3.5.28]. Las funciones incorporadas que esperan rutas suelen aceptar también cadenas de texto. Por ejemplo, podés escribir:

```
#figure(
  // Path to an image
  image("tiger.jpg"),
  caption: [A tiger],
)

// Path to a Typst file
#include "chapter.typ"
```

Hay dos tipos de estas cadenas de ruta: relativas y absolutas.

- Una **ruta relativa** se resuelve en relación con el directorio padre del archivo Typst donde se llama a la función. Si bien este es el comportamiento por defecto, también se puede especificar explícitamente que una ruta es relativa empezándola con `./`.

  ```
  #image("images/logo.png")
  #image("./images/logo.png") // This is equivalent
  ```

- Una **ruta absoluta** siempre se resuelve en relación con la *raíz*[3.5.22/3.1] del proyecto. Una ruta así se indica con una / inicial:

  ```
  #image("/assets/logo.png")
  ```

Las rutas se componen de segmentos separados por barras, donde los segmentos interiores indican directorios y el último, un archivo o un directorio. Hay dos componentes de ruta que reciben un tratamiento especial:

- El segmento `.` hace referencia al directorio *actual*. Por eso `"./image.png"` y `"image.png"` son equivalentes.
- El segmento `..` hace referencia al directorio *padre*. Si tenés tres archivos `main.typ`, `utils.typ` y `text/chapter1.typ`, podés hacer referencia a tu archivo de utilidades desde el capítulo 1 con la ruta `"../utils.typ"`.

## /2 El tipo path

Para la mayoría de los usos típicos de las rutas, alcanza con las cadenas de texto. Sin embargo, a veces necesitás un poco más de control. Por ejemplo, quizás quieras resolver una ruta en relación con el archivo en el que estás escribiendo, pero luego pasársela a un paquete y dejar que el paquete lea desde esa ruta. Ahí es donde entra el tipo path.

Con él, podés resolver por completo una cadena de ruta en relación con el archivo donde la construís. Todas las operaciones posteriores que se realicen con la ruta (como leer un archivo o cargar una imagen) se comportarán igual sin importar en qué parte del código ocurran.

Este es un ejemplo de cómo podríamos tener un `main.typ` con un archivo `data.json` justo al lado y aun así dejar que un paquete que construimos lea ese archivo.

```
// This is main.typ, with data.json next to it.
#import "@local/my-pkg:0.1.0": process
```

```
#let data-path = path("data.json")
#process(data-path)
```

## /3 Raíces

### /3.1 La raíz del proyecto

Por razones de seguridad y reproducibilidad, Typst encapsula el acceso a archivos. Un proyecto de Typst solo puede acceder a rutas dentro de su *raíz del proyecto*. Si intentás crear o acceder a una ruta fuera de esta raíz, vas a obtener un error:

```
// ❌ Error: path `"../secret.txt"` would escape the project root
#path("../secret.txt")
```

Por defecto, la raíz del proyecto es el directorio padre del archivo principal de Typst. Si querés usar otra carpeta como raíz de tu proyecto, podés usar el indicador `--root` de la CLI:

```
typst compile --root .. file.typ
```

Asegurate de que el archivo principal esté contenido en el subárbol de la carpeta, para que Typst pueda acceder a él.

En la app web, el proyecto en sí es el directorio raíz. Siempre podés leer todos los archivos que contiene, sin importar cuál se esté previsualizando (mediante el interruptor con forma de ojo junto a cada archivo Typst en el panel de archivos).

### /3.2 Raíces de paquetes

Igual que el proyecto, cada paquete que importás tiene su propia raíz. Dentro de un paquete, las rutas absolutas apuntan a la raíz del paquete y no a la raíz del proyecto. Por sí solo, el código de un paquete no puede construir una ruta que esté en el proyecto o en otro paquete.

Si necesitás proporcionarle a un paquete recursos del proyecto (como una imagen de logo), podés hacerlo creando explícitamente en tu código una ruta al recurso con el constructor de path[3.5.22/5]. Después podés pasarle al paquete la ruta resultante. En la sección "El tipo path"[3.5.22/2] de arriba se muestra un ejemplo de esto.

Como alternativa, podés realizar la operación de ruta en tu código y pasarle el resultado al paquete. Esto podría ser, por ejemplo, el resultado de una llamada a `read`[3.13.5] o una imagen completa (por ejemplo, como parámetro con nombre `logo: image("mylogo.svg")`). Tené en cuenta que, si le pasás una imagen a un paquete de esta manera, igualmente podés personalizar el aspecto de la imagen con una regla set dentro del paquete.

## /4 Otras operaciones

Por ahora, el propósito del tipo path se limita a manejar y transferir correctamente rutas entre los archivos de tu proyecto y los paquetes. En el futuro, podría habilitar capacidades adicionales, como comprobar la existencia de un archivo o enumerar los archivos de un directorio.

## /5 Constructor

Crea una ruta a partir de una cadena de texto.

```
// A relative path without a leading slash.
// May optionally start with `./`.
#path("relative/path/to/file.typ")
#path("./relative/path/to/file.typ")

// An absolute path with a leading slash.
#path("/absolute/path/to/file.typ")
```

```
path( str path ) → path
```

/5.1 **path** `str` or `path` *Obligatorio* *Posicional*

Convierte una cadena de texto o una ruta en una ruta.

Si es una cadena de ruta[3.5.22/1]:

- Si la ruta es absoluta, se resuelve en relación con la raíz del proyecto o paquete en el que se llama a esta función.
- Si la ruta es relativa, se resuelve en relación con el archivo donde se llama a esta función.

Si ya es un `path`, se devuelve sin cambios.

## 3.5.23 `plugin` Desde: 0.8.0

Carga un módulo de WebAssembly.

El módulo[3.5.19] resultante contendrá una función[3.5.16] de Typst por cada función exportada del módulo de WebAssembly cargado.

Los plugins de WebAssembly de Typst tienen que seguir un protocolo[3.5.23/5] específico. Para ejecutarse como plugin, un programa debe compilarse como biblioteca compartida de WebAssembly de 32 bits. Las funciones de un plugin pueden aceptar varios buffers de bytes[3.5.7] como argumentos y devolver un único buffer de bytes. Normalmente deberían envolverse en funciones idiomáticas de Typst que realicen las conversiones necesarias entre los tipos nativos de Typst y los bytes, aprovechando `str`[3.5.28/3], `bytes`[3.5.7/1] y las funciones de carga de datos[3.13].

Por razones de seguridad, los plugins se ejecutan aislados de tu sistema. Esto significa que no se admite imprimir, leer archivos ni cosas similares.

### /1 Ejemplo

```
#let myplugin = plugin("hello.wasm")
#let concat(a, b) = str(
  myplugin.concatenate(
    bytes(a),
    bytes(b),
  )
)

#concat("hello", "world")
```

hello*world

Como la función plugin devuelve un módulo, se puede usar con la sintaxis de importación:

```
#import plugin("hello.wasm"): concatenate
```

### /2 Pureza

Las funciones de un plugin **deben ser puras:** una llamada a una función de un plugin no debe tener ningún efecto secundario observable en futuras llamadas al plugin y, dados los mismos argumentos, siempre debe devolver el mismo valor.

La razón es que las funciones de Typst deben ser puras (algo bastante fundamental en el diseño del lenguaje) y, como las funciones de Typst pueden llamar a funciones de plugins, este requisito se hereda. En particular, si una función de un plugin se llama dos veces con los mismos argumentos, Typst podría almacenar en caché los resultados y llamar a tu función una sola vez. Además, Typst puede ejecutar varias instancias de tu plugin en varios hilos, sin estado compartido entre ellas.

Typst no obliga a que las funciones de los plugins sean puras (por razones de eficiencia), pero llamar a una función impura llevará a resultados impredecibles e irreproducibles y debe evitarse.

Dicho esto, las operaciones mutables *pueden ser* útiles para plugins que requieren una inicialización costosa en tiempo de ejecución. Debido al requisito de pureza, esa inicialización no se puede realizar mediante una llamada normal a una función. En cambio, Typst expone una API de transición de plugins[3.5.23/8.1], que ejecuta una llamada a una función y luego crea un módulo derivado con funciones nuevas que observarán los efectos secundarios producidos por la llamada de transición. El plugin original no se ve afectado.

### /3 Plugins y paquetes

Cualquier código de Typst puede usar un plugin simplemente incluyendo un archivo de WebAssembly y cargándolo. Sin embargo, como la interfaz de plugins basada en bytes es de muy bajo nivel, los plugins suelen exponerse a través de un paquete que contiene el plugin y funciones envoltorio idiomáticas.

## /4 WASI

Muchos compiladores usan por defecto, o como única opción, la ABI de WASI[A] (por ejemplo, emscripten), que permite imprimir, leer archivos, etcétera. Esta ABI no funciona directamente con Typst. Vas a tener que compilar a otro destino o reemplazar todas las funciones por stubs[B].

## /5 Protocolo

Para usarse como plugin, un módulo de WebAssembly debe cumplir el siguiente protocolo:

### /5.1 Exportaciones

Un módulo de plugin puede exportar funciones para que se puedan llamar desde Typst. Para cumplir el protocolo, una función exportada debe:

- Tomar `n` argumentos enteros de 32 bits `a_1`, `a_2`, ..., `a_n` (interpretados como longitudes, por lo que `usize`/`size_t` puede ser preferible) y devolver un entero de 32 bits.
- La función debe primero asignar un buffer `buf` de longitud `a_1 + a_2 + ... + a_n` y luego llamar a `wasm_minimal_protocol_write_args_to_buffer(buf.ptr)`.
- Los primeros `a_1` bytes del buffer constituyen ahora el primer argumento, los siguientes `a_2` bytes el segundo argumento, y así sucesivamente.
- La función ya puede hacer su trabajo con los argumentos y producir un buffer de salida. Antes de retornar, debe llamar a `wasm_minimal_protocol_send_result_to_host` para enviar su resultado de vuelta al host.
- Para señalar éxito, la función debe devolver `0`.
- Para señalar un error, la función debe devolver `1`. El buffer escrito se interpreta entonces como un mensaje de error codificado en UTF-8.

### /5.2 Importaciones

Los módulos de plugin necesitan importar dos funciones que provee el entorno de ejecución. (Los tipos y las funciones se describen con la sintaxis de WAT.)

- `(import "typst_env" "wasm_minimal_protocol_write_args_to_buffer" (func (param i32)))`

  Escribe los argumentos de la función actual en un buffer asignado por el plugin. Cuando se llama a una función de un plugin, esta recibe las longitudes[3.5.23/5.1] de sus buffers de entrada como argumentos. Debe entonces asignar un buffer cuya capacidad sea al menos la suma de esas longitudes. Luego debe llamar a esta función con un `ptr` al buffer para llenarlo con los argumentos, uno después del otro.

- `(import "typst_env" "wasm_minimal_protocol_send_result_to_host" (func (param i32 i32)))`

  Envía la salida de la función actual al host (Typst). El primer parámetro debe ser un puntero a un buffer (`ptr`), mientras que el segundo es la longitud de ese buffer (`len`). La memoria a la que apunta `ptr` se puede liberar inmediatamente después de que esta función retorne. Si el mensaje debe interpretarse como un mensaje de error, debe codificarse en UTF-8.

## /6 Recursos

Para más recursos, mirá el repositorio wasm-minimal-protocol[C]. Contiene:

- Una lista de implementaciones de plugins de ejemplo y un ejecutor de pruebas para esos ejemplos
- Envoltorios que te ayudan a escribir tu plugin en Rust
- Un generador de stubs para WASI

[A] https://wasi.dev/
[B] https://github.com/typst-community/wasm-minimal-protocol/tree/main/crates/wasi-stub
[C] https://github.com/typst-community/wasm-minimal-protocol

## /7 Parámetros

```
plugin( str  path  bytes ) → module
```

### /7.1 **source** `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo de WebAssembly o bytes de WebAssembly sin procesar.

## /8 Funciones

### /8.1 **transition** Desde: 0.13.0

Llama a una función de un plugin que tiene efectos secundarios y devuelve un módulo nuevo con funciones de plugin que garantizadamente observaron los resultados de la llamada mutable.

Tené en cuenta que llamar a una función impura mediante una llamada normal a una función (sin usar la API de transición) está prohibido y lleva a un comportamiento impredecible. Leé la sección sobre pureza[3.5.23/2] para más detalles.

En el siguiente ejemplo, cargamos el plugin `hello-mut.wasm`, que exporta dos funciones: la función `get()` obtiene un array global como cadena de texto. La función `add(value)` agrega un valor al array global.

Llamamos a `add` mediante la API de transición. La llamada `mutated.get()` en el módulo derivado observará la adición. Mientras tanto, el módulo original permanece intacto, como demuestra la llamada `base.get()`.

*Nota:* Debido a limitaciones de la implementación interna de WebAssembly, la API de transición solo puede garantizar que se reflejen los cambios en la memoria del plugin, no en los globales de WebAssembly. Si tu plugin depende de que los cambios en los globales sean visibles después de la transición, quizás quieras evitar por ahora el uso de la API de transición. Esperamos levantar esta limitación en el futuro.

```
#let base = plugin("hello-mut.wasm")
#assert.eq(base.get(), "[]")

#let mutated = plugin.transition(base.add, "hello")
#assert.eq(base.get(), "[]")
#assert.eq(mutated.get(), "[hello]")
```

```
plugin.transition(
  function,
  ..bytes,
) → module
```

#### /8.1.1 **func** `function` *Obligatorio* *Posicional*

La función del plugin a la que se llama.

#### /8.1.2 **arguments** `bytes` *Obligatorio* *Posicional* *Variádico*

Los buffers de bytes con los que se llama a la función.

### 3.5.24 `regex` Desde: siempre

Una expresión regular.

Se puede usar como selector de reglas show[3.2.2] o con métodos de cadenas de texto[3.5.28].

Visitá este sitio web[A] para ver la especificación completa de la sintaxis admitida.

#### /1 Con métodos de cadenas de texto

Las expresiones regulares se pueden usar con métodos de cadenas de texto como `find`[3.5.28/4.14], `split`[3.5.28/4.20], `replace`[3.5.28/4.18], `match`[3.5.28/4.16] o `matches`[3.5.28/4.17]. Mirá la documentación de estos métodos para más detalles.

```
#"a,b;c".split(regex("[,;]"))
```

```
("a", "b", "c")
```

#### /2 Con reglas show

Las expresiones regulares también se pueden usar con reglas show[3.2.2] para encontrar y transformar texto del documento. Por ejemplo, a continuación pintamos de rojo todos los números.

```
#show regex("\\d+"): set text(red)

The numbers 1 to 10.
```

```
The numbers 1 to 10.
```

A veces también puede convenir combinar ambos usos: primero buscar texto con una regla show y luego volver a aplicar la expresión sobre ese texto para extraer un grupo de captura[B] específico. En ese caso, puede ser cómodo guardar la expresión regular en una variable en lugar de repetirla dos veces.

```
#let pattern = regex("\|([^|]*)\|")
#show pattern: it => {
  let m = it.text.match(pattern)
  let inner = m.captures.first()
  highlight(inner)
}

A |handy-dandy| highlighter!
```

```
A handy-dandy highlighter!
```

#### /3 Constructor

Crea una expresión regular a partir de una cadena de texto.

```
regex( str ) → regex
```

##### /3.1 **regex** `str` *Obligatorio* *Posicional*

La expresión regular como cadena de texto.

Tanto las cadenas de texto de Typst como las expresiones regulares usan barras invertidas para el escape. Para producir una secuencia de escape de regex que también sea válida en Typst, tenés que escapar la propia barra invertida (por ejemplo, escribir `regex("\\\\")` para la regex `\\`). Las secuencias de escape de regex que no son secuencias de escape válidas de Typst (por ejemplo, `\d` y `\b`) se pueden ingresar directamente en las cadenas, pero es buena práctica escaparlas igualmente para evitar ambigüedades (es decir, `regex("\\b\\d")`). Mirá la lista de secuencias de escape válidas en cadenas de texto[3.5.28/2].

[A] https://docs.rs/regex/latest/regex/#syntax
[B] https://docs.rs/regex/latest/regex/#grouping-and-flags

Si necesitás muchas secuencias de escape, también podés crear un elemento raw y extraer su texto para usarlo en tus expresiones regulares: `regex(`\d+\.\d+\.\d+`.text)`.

### 3.5.25 `repr` Desde: siempre

Devuelve la representación en cadena de texto de un valor.

Cuando se inserta en contenido, la mayoría de los valores se muestran como esta representación en monoespaciado y con resaltado de sintaxis. Las excepciones son none, los enteros, los flotantes, las cadenas de texto, el contenido y las funciones.

#### /1 Ejemplo

```
#none vs #repr(none) \
#"hello" vs #repr("hello") \
#(1, 2) vs #repr((1, 2)) \
#[*Hi*] vs #repr([*Hi*])
```

vs none
hello vs "hello"
(1, 2) vs (1, 2)
**Hi** vs strong(body: [Hi])

#### /2 Solo para depuración

Esta función es para depuración. Su salida no debe considerarse estable y puede cambiar en cualquier momento.

Para ser específicos, tener el mismo `repr` no garantiza que los valores sean equivalentes, y `repr` no es una inversa estricta de `eval`[3.5.14]. En el siguiente ejemplo, para mayor legibilidad, la `length`[3.10.13] se redondea a dos cifras significativas y se omiten la lista de parámetros y el cuerpo de la `function` sin nombre[3.5.16/6].

```
#assert(2pt / 3 < 0.67pt)
#repr(2pt / 3)

#repr(x => x + 1)
```

0.67pt

(..) => ..

#### /3 Parámetros

`repr( any ) → str`

##### /3.1 `value` `any` *Obligatorio* *Posicional*

El valor del que se produce la representación en cadena de texto.

### 3.5.26 `selector` Desde: 0.3.0

Un filtro para seleccionar elementos dentro del documento.

Para construir un selector podés:
- usar una función de elemento[3.5.16/2]
- filtrar por una función de elemento con campos específicos[3.5.16/8.2]
- usar una cadena de texto[3.5.28] o una expresión regular[3.5.24]
- usar una `<label>`[3.5.18]
- usar una `location`[3.12.4]
- llamar al constructor de `selector`[3.5.26] para convertir cualquiera de los tipos anteriores en un valor selector y usar los métodos de abajo para refinarlo

Los selectores se usan para aplicar reglas de estilo[3.2.2] a los elementos. También podés usar selectores para consultar[3.12.6] el documento en busca de ciertos tipos de elementos.

Además, podés pasar un selector a varias de las funciones incorporadas de Typst para configurar su comportamiento. Un ejemplo es outline[3.6.13], donde se puede usar para cambiar qué elementos se listan en el índice.

Se pueden combinar varios selectores con los métodos que se muestran abajo. Sin embargo, por el momento no todos los tipos de selectores se admiten en todos los lugares.

#### /1 Ejemplo

```
#context query(
  heading.where(level: 1)
    .or(heading.where(level: 2))
)

= This will be found
== So will this
=== But this will not.
```

```
(
  heading(
    level: 1,
    depth: 1,
    offset: 0,
    numbering: none,
    supplement: [Sección],
    outlined: true,
    bookmarked: auto,
    hanging-indent: auto,
    body: [This will be found],
  ),
  heading(
    level: 2,
    depth: 2,
    offset: 0,
    numbering: none,
    supplement: [Sección],
    outlined: true,
    bookmarked: auto,
    hanging-indent: auto,
    body: [So will this],
  ),
)
```

**This will be found**

**So will this**

**But this will not.**

#### /2 Constructor

Convierte un valor en un selector. Se aceptan los siguientes valores:
- Una función de elemento como `heading` o `figure`.
- Una cadena de texto[3.5.28] o una expresión regular[3.5.24].
- Una `<label>`.
- Una `location`[3.12.4].

– Un selector más complejo como `heading.where(level: 1)`.

```
selector( str  regex  label  selector  location  function ) → selector
```

/2.1 **target** `str` or `regex` or `label` or `selector` or `location` or `function` *Obligatorio* *Posicional*

Puede ser una función de elemento como `heading` o `figure`, una `<label>` o un selector más complejo como `heading.where(level: 1)`.

## /3 Funciones

### /3.1 or Desde: 0.3.0

Selecciona todos los elementos que coinciden con este selector o con cualquiera de los demás.

```
self.or(.. str  regex  label  selector  location  function ) → selector
```

/3.1.1 **others** `str` or `regex` or `label` or `selector` or `location` or `function` *Obligatorio* *Posicional* *Variádico*

Los demás selectores con los que se debe coincidir.

### /3.2 and Desde: 0.3.0

Selecciona todos los elementos que coinciden con este selector y con todos los demás.

```
self.and(.. str  regex  label  selector  location  function ) → selector
```

/3.2.1 **others** `str` or `regex` or `label` or `selector` or `location` or `function` *Obligatorio* *Posicional* *Variádico*

Los demás selectores con los que se debe coincidir.

### /3.3 before Desde: 0.3.0

Devuelve un selector modificado que solo coincide con los elementos que aparecen antes de la primera coincidencia de end.

*Nota:* Actualmente, este selector solo se admite con funciones de introspección, no en reglas show.

```
self.before(
  label  selector  location  function ,
  inclusive:  bool ,
) → selector
```

/3.3.1 **end** `label` or `selector` or `location` or `function` *Obligatorio* *Posicional*

La selección original terminará en la primera coincidencia de end.

/3.3.2 **inclusive** `bool` Por defecto: `true`

Si la propia coincidencia de end debe incluirse o no. Solo es relevante si ambos selectores coinciden con el mismo tipo de elemento. Por defecto es `true`.

### /3.4 after Desde: 0.3.0

Devuelve un selector modificado que solo coincide con los elementos que aparecen después de la primera coincidencia de `start`.

*Nota:* Actualmente, este selector solo se admite con funciones de introspección, no en reglas show.

```
self.after(
  label  selector  location  function ,
```

```
  inclusive: bool,
) → selector
```

/3.4.1 **start** label or selector or location or function *Obligatorio* *Posicional*

La selección original empezará en la primera coincidencia de `start`.

/3.4.2 **inclusive** bool Por defecto: true

Si la propia coincidencia de `start` debe incluirse o no. Solo es relevante si ambos selectores coinciden con el mismo tipo de elemento. Por defecto es `true`.

## /3.5 within Desde: 0.15.0

Devuelve un selector modificado que solo coincide con los elementos contenidos dentro de cualquier elemento que coincida con el selector `ancestor`.

Example · Búsqueda de elementos strong en listas

```
*Strong emphasis* that does not count.

- An *important* word
- Another *key* word

Strong elements in lists:
#context {
  query(selector(strong).within(list))
    .map(it => it.body)
    .join[, ]
}
```

**Strong emphasis** that does not count.

- An **important** word
- Another **key** word

Strong elements in lists: important, key

También se puede usar en combinación con here[3.12.2] para encontrar todas las coincidencias de un selector dentro de una expresión de contexto[3.4]. Esto puede ser muy útil para que una introspección devuelva resultados locales a algún componente que estés construyendo.

Example · Conteo local de elementos en un bloque de contexto

```
#let count(sel, body) = context {
  let n = query(selector(sel).within(here())).len()
  [#body (#n matches)]
}

- #count(emph)[Has _two_ matching _elements_]
- #count(strong)[Has *one* matching element]
```

- Has *two* matching *elements* (2 matches)
- Has **one** matching element (1 matches)

*Nota:* Actualmente, este selector solo se admite con funciones de introspección, no en reglas show.

```
self.within(label selector location function) → selector
```

/3.5.1 **ancestor** label or selector or location or function *Obligatorio* *Posicional*

Solo se incluirán en la salida las coincidencias de `self` que sean descendientes de algún elemento que coincida con este selector.

Un elemento no se considera su propio ancestro.

## 3.5.27 Sistema

Módulo para las interacciones con el sistema.

### /1 Constantes

#### /1.1 `version` `version` Desde: 0.9.0

La versión del compilador de Typst actualmente activa.

```
#sys.version
```

```
0.15.1
```

#### /1.2 `inputs` `dictionary` Desde: 0.11.0

Hace que entradas externas estén disponibles para el proyecto.

Una entrada especificada en la línea de comandos como `--input key=value` queda disponible en `sys.inputs.key` como `"value"`. Para incluir espacios en el valor, se lo puede encerrar entre comillas simples o dobles.

El valor siempre es de tipo cadena de texto[3.5.28]. Los datos más complejos se pueden analizar manualmente con funciones como `json`[3.13.4].

### 3.5.28 `str` Desde: siempre

Una secuencia de puntos de código Unicode.

Podés iterar sobre los grupos de grafemas de la cadena con un bucle for[3.3.5]. Los grupos de grafemas son básicamente caracteres, pero mantienen juntas las cosas que van juntas, por ejemplo, varios puntos de código que en conjunto forman un emoji de bandera. Las cadenas de texto se pueden sumar con el operador +, unir[3.3.2] y multiplicar por enteros.

Typst ofrece métodos de utilidad para manipular cadenas de texto. Muchos de estos métodos (por ejemplo, `split`[3.5.28/4.20], `trim`[3.5.28/4.19] y `replace`[3.5.28/4.18]) operan sobre *patrones*: un patrón puede ser una cadena de texto o una expresión regular[3.5.24]. Esto hace que los métodos sean muy versátiles.

Todas las longitudes e índices se expresan en bytes UTF-8. Los índices empiezan en cero y los índices negativos dan la vuelta hasta el final de la cadena.

Podés convertir un valor en cadena de texto con el constructor `str`.

#### /1 Ejemplo

```
#"hello world!" \
#"\"hello\n  world\"!" \
#"1 2 3".split() \
#"1,2;3".split(regex("[,;]")) \
#(regex("\\d+") in "ten euros") \
#(regex("\\d+") in "10 euros")
```

```
hello world!
"hello
  world"!
("1", "2", "3")
("1", "2", "3")
false
true
```

#### /2 Secuencias de escape

Igual que en el marcado, en las cadenas de texto podés escapar algunos símbolos:
- `\\` para una barra invertida
- `\"` para una comilla
- `\n` para un salto de línea
- `\r` para un retorno de carro
- `\t` para una tabulación
- `\u{1f600}` para una secuencia de escape Unicode hexadecimal

#### /3 Constructor

Convierte un valor en una cadena de texto.

- Los enteros se formatean en base 10. Esto se puede cambiar con el parámetro opcional `base`.
- Los flotantes se formatean en base 10 y nunca en notación exponencial.
- Los enteros y flotantes negativos se formatean con el signo menos de Unicode ("−" U+2212) en lugar del signo menos ASCII ("-" U+002D).
- De las etiquetas se extrae el nombre.
- Los bytes se decodifican como UTF-8.

Si querés convertir desde y hacia puntos de código Unicode, mirá las funciones `to-unicode`[3.5.28/4.8] y `from-unicode`[3.5.28/4.9].

```
#str(10) \
#str(4000, base: 16) \
#str(2.7) \
#str(1e8) \
#str(<intro>)
```

10
fa0
2.7
100000000
intro

```
str(
  int float decimal str label bytes version type,
  base: int,
) → str
```

/3.1 **value** `int` or `float` or `decimal` or `str` or `label` or `bytes` or `version` or `type`
*Obligatorio* *Posicional*

El valor que se debe convertir en cadena de texto.

/3.2 **base** `int` Por defecto: `10`

La base (raíz) en la que se muestran los enteros, entre 2 y 36.

## /4 Funciones

### /4.1 len Desde: siempre

La longitud de la cadena en bytes codificados en UTF-8.

```
self.len() → int
```

### /4.2 first Desde: siempre

Extrae el primer grupo de grafemas de la cadena.

Devuelve el valor por defecto indicado si la cadena está vacía, o falla con un error si no se especificó ningún valor por defecto.

```
self.first(default: str) → str
```

/4.2.1 **default** `str`

Un valor por defecto para devolver si la cadena está vacía.

### /4.3 last Desde: siempre

Extrae el último grupo de grafemas de la cadena.

Devuelve el valor por defecto indicado si la cadena está vacía, o falla con un error si no se especificó ningún valor por defecto.

```
self.last(default: str) → str
```

/4.3.1 **default** `str`

Un valor por defecto para devolver si la cadena está vacía.

### /4.4 at Desde: siempre

Extrae el primer grupo de grafemas después del índice especificado. Devuelve el valor por defecto si el índice está fuera de rango, o falla con un error si no se especificó ningún valor por defecto.

```
self.at(
  int,
  default: any,
) → any
```

/4.4.1 **index** int *Obligatorio* *Posicional*

El índice de bytes. Si es negativo, se indexa desde el final.

/4.4.2 **default** any

Un valor por defecto para devolver si el índice está fuera de rango.

## /4.5 slice Desde: siempre

Extrae una subcadena de la cadena. Falla con un error si el índice de inicio o de fin está fuera de rango.

```
self.slice(
  int,
  none int,
  count: int,
) → str
```

/4.5.1 **start** int *Obligatorio* *Posicional*

El índice de bytes inicial (inclusivo). Si es negativo, se indexa desde el final.

/4.5.2 **end** none or int *Posicional* Por defecto: none

El índice de bytes final (exclusivo). Si se omite, se extrae toda la porción hasta el final de la cadena. Si es negativo, se indexa desde el final.

/4.5.3 **count** int

La cantidad de bytes que se extraen. Equivale a pasar `start + count` como posición end. Es mutuamente excluyente con end.

## /4.6 clusters Desde: siempre

Devuelve los grupos de grafemas de la cadena como un array de subcadenas.

```
self.clusters() → array
```

## /4.7 codepoints Desde: siempre

Devuelve los puntos de código Unicode de la cadena como un array de subcadenas.

```
self.codepoints() → array
```

## /4.8 to-unicode Desde: 0.5.0

Convierte un carácter en su punto de código correspondiente.

```
#"a".to-unicode() \
#("a\u{0300}"
   .codepoints()
   .map(str.to-unicode))
```

```
97
(97, 768)
```

```
str.to-unicode(str) → int
```

/4.8.1 **character** str *Obligatorio* *Posicional*

El carácter que se debe convertir.

## /4.9 from-unicode Desde: 0.5.0

Convierte un punto de código Unicode en su cadena correspondiente.

```
#str.from-unicode(97)
```

a

```
str.from-unicode( int ) → str
```

/4.9.1 **value** int *Obligatorio* *Posicional*

El punto de código que se debe convertir.

## /4.10 normalize Desde: 0.14.0

Normaliza la cadena a la forma normal Unicode indicada.

Esto es útil al manipular cadenas que contienen caracteres combinados de Unicode.

```
#assert.eq("é".normalize(form: "nfd"), "e\u{0301}")
#assert.eq("ſ".normalize(form: "nfkc"), "s")
```

```
self.normalize(form: str ) → str
```

/4.10.1 **form** str Por defecto: "nfc"

| Variante | Detalles |
|---|---|
| "nfc" | Composición canónica, en la que, por ejemplo, las letras acentuadas se convierten en un único punto de código Unicode. |
| "nfd" | Descomposición canónica, en la que, por ejemplo, las letras acentuadas se dividen en una base y un signo diacrítico separados. |
| "nfkc" | Como NFC, pero usando las descomposiciones de compatibilidad de Unicode. |
| "nfkd" | Como NFD, pero usando las descomposiciones de compatibilidad de Unicode. |

## /4.11 contains Desde: siempre

Si la cadena contiene el patrón especificado.

Este método también tiene una sintaxis dedicada: podés escribir "bc" in "abcd" en lugar de "abcd".contains("bc").

```
self.contains( str regex ) → bool
```

/4.11.1 **pattern** str or regex *Obligatorio* *Posicional*

El patrón que se busca.

## /4.12 starts-with Desde: siempre

Si la cadena empieza con el patrón especificado.

```
self.starts-with( str regex ) → bool
```

/4.12.1 **pattern** str or regex *Obligatorio* *Posicional*

El patrón con el que podría empezar la cadena.

### /4.13 ends-with Desde: siempre

Si la cadena termina con el patrón especificado.

```
self.ends-with( str regex ) → bool
```

#### /4.13.1 pattern str or regex *Obligatorio* *Posicional*

El patrón con el que podría terminar la cadena.

### /4.14 find Desde: siempre

Busca el patrón especificado en la cadena y devuelve la primera coincidencia como cadena de texto, o none si no hay ninguna.

```
self.find( str regex ) → none str
```

#### /4.14.1 pattern str or regex *Obligatorio* *Posicional*

El patrón que se busca.

### /4.15 position Desde: siempre

Busca el patrón especificado en la cadena y devuelve el índice de la primera coincidencia como entero, o none si no hay ninguna.

```
self.position( str regex ) → none int
```

#### /4.15.1 pattern str or regex *Obligatorio* *Posicional*

El patrón que se busca.

### /4.16 match Desde: siempre

Busca el patrón especificado en la cadena y devuelve un diccionario con detalles sobre la primera coincidencia, o none si no hay ninguna.

El diccionario devuelto tiene las siguientes claves:
- `start`: El desplazamiento de inicio de la coincidencia
- end: El desplazamiento de fin de la coincidencia
- `text`: El texto que coincidió.
- `captures`: Un array con una cadena de texto por cada grupo de captura que coincidió. ¡El primer elemento del array contiene la primera captura, no la coincidencia completa! Está vacío, salvo que el `pattern` sea una regex con grupos de captura.

Example · Forma del diccionario devuelto

```
#let pat = regex("not (a|an) (apple|cat)")
#"I'm a doctor, not an apple.".match(pat) \
#"I am not a cat!".match(pat)
```

```
(
  start: 14,
  end: 26,
  text: "not an apple",
  captures: ("an", "apple"),
)
(
  start: 5,
  end: 14,
  text: "not a cat",
  captures: ("a", "cat"),
)
```

Example · Distintos tipos de patrones

```
#assert.eq("Is there a".match("for this?"), none)
#"The time of my life.".match(regex("[mit]+e"))
```

```
(start: 4, end: 8, text: "time",
captures: ())
```

```
self.match( str regex ) → none dictionary
```

/4.16.1 **pattern** `str` or `regex` *Obligatorio* *Posicional*

El patrón que se busca.

/4.17 **matches** Desde: siempre

Busca el patrón especificado en la cadena y devuelve un array de diccionarios con detalles sobre todas las coincidencias. Para más detalles sobre los diccionarios devueltos, mirá arriba[3.5.28/4.16].

```
#"Day by Day.".matches("Day")
```

```
(
  (start: 0, end: 3, text: "Day",
captures: ()),
  (start: 7, end: 10, text: "Day",
captures: ()),
)
```

```
self.matches( str regex ) → array
```

/4.17.1 **pattern** `str` or `regex` *Obligatorio* *Posicional*

El patrón que se busca.

/4.18 **replace** Desde: siempre

Reemplaza como máximo count apariciones del patrón indicado por una cadena o función de reemplazo (empezando desde el principio). Si no se indica una cantidad, se reemplazan todas las apariciones.

```
self.replace(
  str regex,
  str function,
  count: int,
) → str
```

/4.18.1 **pattern** str or regex *Obligatorio* *Posicional*

El patrón que se busca.

/4.18.2 **replacement** str or function *Obligatorio* *Posicional*

La cadena por la que se reemplazan las coincidencias, o una función que recibe un diccionario por cada coincidencia y puede devolver cadenas de reemplazo individuales.

El diccionario que se le pasa a la función tiene la misma forma que el diccionario devuelto por match[3.5.28/4.16].

/4.18.3 **count** int

Si se indica, solo se reemplazan las primeras count coincidencias del patrón.

/4.19 **trim** Desde: siempre

Elimina las coincidencias de un patrón de uno o ambos lados de la cadena, una vez o repetidamente, y devuelve la cadena resultante.

```
self.trim(
  none str regex,
  at: alignment,
  repeat: bool,
) → str
```

/4.19.1 **pattern** none or str or regex *Posicional* Por defecto: none

El patrón que se busca. Si es none, recorta los espacios en blanco.

/4.19.2 **at** alignment

Puede ser start o end para recortar solo el inicio o el final de la cadena. Si se omite, se recortan ambos lados.

/4.19.3 **repeat** bool Por defecto: true

Si se eliminan repetidamente las coincidencias del patrón o solo una vez. Por defecto es true.

/4.20 **split** Desde: siempre

Divide una cadena en las coincidencias de un patrón especificado y devuelve un array con las partes resultantes.

Cuando se usa la cadena vacía como separador, se separa cada carácter (es decir, cada punto de código Unicode) de la cadena, junto con el principio y el final. En la práctica, esto significa que la lista de partes resultante contendrá la cadena vacía al principio y al final.

```
self.split(none str regex) → array
```

/4.20.1 **pattern** none or str or regex *Posicional* Por defecto: none

El patrón por el que se divide. Por defecto son los espacios en blanco.

/4.21 **rev** Desde: 0.8.0

Invierte la cadena.

Más específicamente, devuelve una cadena con los mismos grupos de grafemas, en orden inverso.

```
#"Pirate flag: 🏴‍☠️".rev()
```

```
🏴‍☠️ :galf etariP
```

```
self.rev() → str
```

### 3.5.29 `symbol` Desde: siempre

Un símbolo Unicode.

Typst define símbolos comunes para que se puedan escribir fácilmente con teclados estándar. Los símbolos se definen en módulos, desde los cuales se puede acceder a ellos con la notación de acceso a campos[3.3.6]:

- Los símbolos generales se definen en el módulo `sym`[3.9.2] y, en modo matemático, son accesibles sin el prefijo `sym.`.
- Los emojis se definen en el módulo `emoji`[3.9.3]

Además, podés definir símbolos personalizados con la función constructora de este tipo.

```
#sym.arrow.r \
#sym.gt.eq.not \
$gt.eq.not$ \
#emoji.face.halo
```

→
≱
≱
😇

Muchos símbolos tienen distintas variantes, que se pueden seleccionar agregando los modificadores con la notación de punto. El orden de los modificadores no importa. Visitá las páginas de documentación de los módulos de símbolos y hacé clic en un símbolo para ver sus variantes disponibles.

```
$arrow.l$ \
$arrow.r$ \
$arrow.t.quad$
```

←
→
⟰

#### /1 Constructor

Crea un símbolo personalizado con modificadores.

```
#let envelope = symbol(
  "🖂",
  ("stamped", "🖃"),
  ("stamped.pen", "🖆"),
  ("lightning", "🖄"),
  ("fly", "🖅"),
)

#envelope
#envelope.stamped
#envelope.stamped.pen
#envelope.lightning
#envelope.fly
```

🖂 🖃 🖆 🖄 🖅

`symbol(.. str array ) → symbol`

##### /1.1 `variants` `str` or `array` *Obligatorio* *Posicional* *Variádico*

Las variantes del símbolo.

Puede ser simplemente una cadena de texto con un único carácter para la variante sin modificadores o un array con dos cadenas que especifican los modificadores y el símbolo. Los modificadores individuales deben separarse con puntos. Al mostrar un símbolo, Typst selecciona la primera de las variantes que tienen todos los modificadores adjuntos y la menor cantidad de otros modificadores.

### 3.5.30 `target` Contextual Desde: 0.13.0

Devuelve el destino de exportación actual.

Esta función devuelve
- `"paged"` en la exportación a PDF[3.14.4], PNG[3.14.5] y SVG[3.14.6], o dentro de un marco HTML[3.14.3/5]
- `"html"` en la exportación a HTML[3.14.3]
- `"bundle"` en la exportación a Bundle[3.14.2]

#### /1 Cuándo usarla

Esta función te permite dar formato correcto a tu documento en los destinos de exportación paginado, HTML y de varios archivos. Debería usarse principalmente en plantillas y reglas show, en lugar de directamente en el contenido. De esta manera, el contenido del documento puede ser totalmente independiente del destino de exportación y se puede compartir entre distintos destinos.

#### /2 Destinos variables

Esta función es contextual[3.4], ya que el destino puede variar dentro de una misma compilación: al exportar a HTML, el destino será `"paged"` dentro de un `html.frame`[3.14.3/5].

#### /3 Ejemplo

```
#let kbd(it) = context {
  if target() == "html" {
    html.elem("kbd", it)
  } else {
    set text(fill: rgb("#1f2328"))
    let r = 3pt
    box(
      fill: rgb("#f6f8fa"),
      stroke: rgb("#d1d9e0b3"),
      outset: (y: r),
      inset: (x: r),
      radius: r,
      raw(it)
    )
  }
}

Press #kbd("F1") for help.
```

Press `F1` for help.

#### /4 Parámetros

```
target() → str
```

### 3.5.31 `type` Desde: 0.8.0

Describe una clase de valor.

Para dar estilo a tu documento, tenés que trabajar con valores de distintas clases: longitudes que especifican el tamaño de tus elementos, colores para tu texto y tus formas, y más. Typst los categoriza en *tipos* claramente definidos y te indica dónde espera cada tipo de valor.

Además de los tipos básicos de valores numéricos y de los tipos[3.5.17] habituales[3.5.15] de[3.5.28] los[3.5.2] lenguajes de programación[3.5.12], Typst ofrece un tipo especial para el *contenido*.[3.5.8] Un valor de este tipo puede contener todo lo que podés ingresar en tu documento: texto, elementos como títulos y formas, e información de estilo.

#### /1 Ejemplo

```
#let x = 10
#if type(x) == int [
  #x is an integer!
] else [
  #x is another value...
]

An image is of type
#type(image("glacier.jpg")).
```

10 is an integer!

An image is of type content.

El tipo de `10` es `int`. Ahora bien, ¿cuál es el tipo de `int` o incluso de `type`?

```
#type(int) \
#type(type)
```

```
type
type
```

A diferencia de otros tipos como `int`, none[3.5.20] y auto[3.5.4] no tienen un nombre que los represente. Para comprobar si un valor es uno de ellos, comparalo directamente con ellos, por ejemplo:

```
#let val = none
#if val == none [
  Yep, it's none.
]
```

Yep, it's none.

Tené en cuenta que `type` devuelve `content`[3.5.8] para todos los elementos del documento. Para determinar programáticamente con qué clase de contenido estás tratando, mirá `content.func`[3.5.8/3.1].

#### /2 Constructor

Determina el tipo de un valor.

```
#type(12) \
#type(14.7) \
#type("hello") \
#type(<glacier>) \
#type([Hi]) \
#type(x => x + 1) \
#type(type)
```

```
int
float
str
label
content
function
type
```

`type( any ) → type`

/2.1 **value** any *Obligatorio* *Posicional*

El valor cuyo tipo se quiere determinar.

## 3.5.32 `version` Desde: 0.9.0

Una versión con una cantidad arbitraria de componentes.

Los primeros tres componentes tienen nombres que se pueden usar como campos: `major`, `minor`, `patch`. Todos los componentes siguientes no tienen nombre.

La lista de componentes se extiende semánticamente con una lista infinita de ceros. Esto significa que, por ejemplo, `0.8` es lo mismo que `0.8.0`. Como caso especial, la versión vacía (que no tiene ningún componente) es lo mismo que `0`, `0.0`, `0.0.0`, etcétera.

La versión actual del compilador de Typst está disponible como `sys.version`[3.5.27/1.1].

Podés convertir una versión en un array de componentes dados explícitamente usando el constructor de `array`[3.5.2].

### /1 Constructor

Crea una versión nueva.

Puede tener cualquier cantidad de componentes (incluso cero).

Example · Construcción de versiones

```
#version() \
#version(1) \
#version(1, 2, 3, 4) \
#version((1, 2, 3, 4)) \
#version((1, 2), 3)
```

1
1.2.3.4
1.2.3.4
1.2.3

Como caso de uso práctico, esto permite comparar la versión actual del compilador (disponible como `sys.version`[3.5.27/1.1]) con una específica.

Example · Comparación con la versión actual

```
Current version: #sys.version \
#(sys.version >= version(0, 14, 0)) \
#(version(3, 2, 0) > version(4, 1, 0))
```

Current version: 0.15.1
true
false

```
version(.. int array ) → version
```

#### /1.1 **components** `int` or `array` *Obligatorio* *Posicional* *Variádico*

Los componentes de la versión (los argumentos de tipo array se aplanan)

### /2 Funciones

#### /2.1 `at` Desde: 0.9.0

Obtiene un componente de una versión.

El entero devuelto nunca es negativo. Devuelve `0` si la versión no está especificada hasta la longitud necesaria.

```
self.at( int ) → int
```

##### /2.1.1 **index** `int` *Obligatorio* *Posicional*

El índice del componente que se quiere obtener. Si es negativo, se indexa desde el final de los componentes dados explícitamente.

# 3.6 Modelo

Estructuración del documento.

Acá podés encontrar funciones para estructurar tu documento e interactuar con esa estructura. Esto incluye los títulos de sección, las figuras, la gestión de bibliografía, las referencias cruzadas y más.

## Definiciones


3.6.1 Bibliography A bibliography / reference listing. . . . 153
3.6.2 Bullet List A bullet list. . . . 160
3.6.3 Cite Cite a work from the bibliography. . . . 163
3.6.4 Divider A thematic break that separates sections of content. . . . 168
3.6.5 Document Manages metadata and is used to add a document file to a bundle. . . . 169
3.6.6 Emphasis Emphasizes content by toggling italics. . . . 173
3.6.7 Figure A figure with an optional caption. . . . 174
3.6.8 Footnote A footnote. . . . 181
3.6.9 Heading A section heading. . . . 184
3.6.10 Link Links to a URL or a location in the document. . . . 189
3.6.11 Numbered List A numbered list. . . . 192
3.6.12 Numbering Applies a numbering to a sequence of numbers. . . . 196
3.6.13 Outline A table of contents, figures, or other elements. . . . 198
3.6.14 Paragraph A logical subdivison of textual content. . . . 204
3.6.15 Paragraph Break A paragraph break. . . . 213
3.6.16 Quote Displays a quote alongside an optional attribution. . . . 214
3.6.17 Reference A reference to a label or bibliography. . . . 218
3.6.18 Strong Emphasis Strongly emphasizes content by increasing the font weight. . . . 221
3.6.19 Table A table of items. . . . 222
3.6.20 Term List A list of terms and their descriptions. . . . 232
3.6.21 Title A document title. . . . 234

### 3.6.1 `bibliography` Elemento Desde: siempre

Una bibliografía / listado de referencias.

Podés crear una bibliografía nueva llamando a esta función con la ruta de un archivo de bibliografía en uno de estos dos formatos:

- Un archivo `.yaml`/`.yml` de Hayagriva. Hayagriva es un formato de archivo de bibliografía nuevo diseñado para usarse con Typst. Visitá su documentación[A] para más detalles.
- Un archivo `.bib` de BibLaTeX.

Apenas agregás una bibliografía en algún lugar de tu documento, podés empezar a citar cosas con la sintaxis de referencias (`@key`) o con llamadas explícitas a la función cite[3.6.3] (`#cite(<key>)`). La bibliografía solo mostrará las entradas de las obras que fueron referenciadas en el documento.

#### /1 Ejemplo

```
This was already noted by
pirates long ago. @arrgh

Multiple sources say ...
@arrgh @netwok.

#bibliography("works.bib")
```

This was already noted by pirates long ago. [1]

Multiple sources say ... [1], [2].

**Bibliografía**

[1] P. T. Leeson, «The Pirate Organization».

[2] R. Astley y L. Morris, «At-scale impact of the Net Wok: A culinarically holistic investigation of distributed dumplings», *Armenian Journal of Proceedings*, vol. 61, pp. 192-219, 2020.

#### /2 Estilos

Typst ofrece una amplia selección de estilos de citas y bibliografía[3.6.1/4.4] incorporados. Además de esos, podés agregar y usar archivos personalizados de CSL[B] (Citation Style Language). ¿Te preguntás qué estilo usar? Estos son algunos buenos valores por defecto según la disciplina en la que trabajes:

| Campos | Estilos típicos |
|---|---|
| Ingeniería, informática | `"ieee"` |
| Psicología, ciencias de la vida | `"apa"` |
| Ciencias sociales | `"chicago-author-date"` |
| Humanidades | `"mla"`, `"chicago-notes"`, `"harvard-cite-them-right"` |
| Economía | `"harvard-cite-them-right"` |
| Física | `"american-physics-society"` |

#### /3 Varias bibliografías

Cuando un documento de Typst contiene varias bibliografías, cada cita se asigna a una de ellas. Por defecto, Typst elegirá automáticamente una bibliografía adecuada (normalmente, la más cercana siguiente que contenga la clave de cita referenciada). Esto cubre casos comunes como las bibliografías por capítulo o temáticas. Para un control más fino, las citas pueden ser apuntadas explícitamente por una bibliografía mediante un selector `target`[3.6.1/4.5].

[A] https://github.com/typst/hayagriva/blob/main/docs/file-format.md
[B] https://citationstyles.org/

## /4 Parámetros

```
bibliography(
  str path bytes array,
  title: none auto content,
  full: bool,
  style: str path bytes,
  target: auto label selector location function,
  group: none auto str,
) → content
```

### /4.1 sources `str` or `path` or `bytes` or `array` *Obligatorio* *Posicional*

Una o varias rutas a archivos `.yaml` de Hayagriva y/o `.bib` de BibLaTeX, o sus bytes sin procesar.

Puede ser:
- Una cadena de ruta o path[3.5.22] desde donde cargar un archivo de bibliografía.
- Bytes sin procesar a partir de los cuales se debe decodificar la bibliografía.
- Un array en el que cada elemento es una de las opciones anteriores.

### /4.2 title `none` or `auto` or `content` *Configurable con set* Por defecto: `auto`

El título de la bibliografía.

- Cuando se establece en `auto`, se usará un título apropiado para el idioma del texto[3.7.12/2.16]. Este es el valor por defecto.
- Cuando se establece en `none`, la bibliografía no tendrá título.
- Se puede establecer un título personalizado pasando contenido.

El título de la bibliografía no estará numerado por defecto, pero podés forzarlo con una regla show-set: `show bibliography: set heading(numbering: "1.")`

### /4.3 full `bool` *Configurable con set* Por defecto: `false`

Si se incluyen todas las obras de los archivos de bibliografía dados, incluso las que no se citaron en el documento.

Para agregar selectivamente obras citadas individuales sin mostrarlas, también podés usar la función `cite` con form[3.6.3/3.3] establecido en `none`.

### /4.4 style `str` or `path` or `bytes` *Configurable con set* Por defecto: `"ieee"`

El estilo de bibliografía.

Puede ser:
- Una cadena de texto con el nombre de uno de los estilos incorporados (ver abajo). Algunos de los estilos listados abajo aparecen dos veces: una con su nombre completo y otra con un alias corto.
- Una cadena de ruta o path[3.5.22] a un archivo CSL[A].
- Bytes sin procesar a partir de los cuales se debe decodificar un estilo CSL.

| Variante | Detalles |
|---|---|
| `"alphanumeric"` | Alfanumérico |
| `"american-anthropological-association"` | American Anthropological Association |
| `"american-chemical-society"` | ACS Guide 2022 revision |
| `"american-geophysical-union"` | American Geophysical Union |
| `"american-institute-of-aeronautics-and-astronautics"` | American Institute of Aeronautics and Astronautics |
| `"american-institute-of-physics"` | AIP Style Manual 4th edition |
| `"american-medical-association"` | AMA Manual of Style 11th edition |

[A] https://citationstyles.org/

| Variante | Detalles |
|---|---|
| `"american-meteorological-society"` | American Meteorological Society |
| `"american-physics-society"` | American Physical Society |
| `"american-physiological-society"` | American Physiological Society |
| `"american-political-science-association"` | APSA Style Manual revised 2018 edition |
| `"american-psychological-association"` | APA Style 7th edition |
| `"apa"` | Un alias corto de `american-psychological-association` |
| `"american-society-for-microbiology"` | American Society for Microbiology |
| `"american-society-of-civil-engineers"` | American Society of Civil Engineers |
| `"american-society-of-mechanical-engineers"` | American Society of Mechanical Engineers |
| `"american-sociological-association"` | ASA Style Guide 6th/7th edition |
| `"angewandte-chemie"` | Angewandte Chemie International Edition |
| `"annual-reviews"` | Annual Reviews (sorted by order of appearance) |
| `"annual-reviews-author-date"` | Annual Reviews (author-date) |
| `"associacao-brasileira-de-normas-tecnicas"` | Associação Brasileira de Normas Técnicas (Português - Brasil) |
| `"association-for-computing-machinery"` | Association for Computing Machinery |
| `"biomed-central"` | BioMed Central |
| `"bristol-university-press"` | Bristol University Press |
| `"british-medical-journal"` | BMJ |
| `"bmj"` | Un alias corto de `british-medical-journal` |
| `"cell"` | Celda |
| `"chicago-author-date"` | Chicago Manual of Style 18th edition (author--date) |
| `"chicago-notes"` | Chicago Manual of Style 18th edition (notes and bibliography) |
| `"chicago-fullnotes"` | Un alias corto de `chicago-notes` |
| `"chicago-shortened-notes"` | Chicago Manual of Style 18th edition (shortened notes and bibliography) |
| `"copernicus"` | Copernicus Publications |
| `"cse-citation-sequence-brackets-8th-edition"` | CSE Manual: Scientific Style and Format 8th edition (citation-sequence, brackets) |
| `"council-of-science-editors"` | Un alias corto de `cse-citation-sequence-brackets-8th-edition` |
| `"cse-name-year"` | CSE Manual: Scientific Style and Format 9th edition (name-year) |
| `"council-of-science-editors-author-date"` | Un alias corto de `cse-name-year` |
| `"current-opinion"` | Current Opinion journals |
| `"deutsche-gesellschaft-für-psychologie"` | Deutsche Gesellschaft für Psychologie 5. Auflage (Deutsch) |
| `"deutsche-sprache"` | Deutsche Sprache (Deutsch) |
| `"elsevier-harvard"` | Elsevier (author-date/Harvard, with titles) |
| `"elsevier-vancouver"` | Elsevier - NLM/Vancouver (citation-sequence) |
| `"elsevier-with-titles"` | Elsevier (numeric, with titles) |
| `"frontiers"` | Frontiers journals |
| `"future-medicine"` | Future Medicine journals |

| Variante | Detalles |
| --- | --- |
| `"future-science"` | Future Science Group |
| `"gb-7714-2005-numeric"` | China National Standard GB/T 7714-2005 (numeric, 中文) |
| `"gb-7714-2015-author-date"` | China National Standard GB/T 7714-2015 (author-date, 中文) |
| `"gb-7714-2015-note"` | China National Standard GB/T 7714-2015 (note, 中文) |
| `"gb-7714-2015-numeric"` | China National Standard GB/T 7714-2015 (numeric, 中文) |
| `"gost-r-705-2008-numeric"` | Russian GOST R 7.0.5-2008 (numeric) |
| `"harvard-cite-them-right"` | Cite Them Right 12th edition (author-date/ Harvard) |
| `"institute-of-electrical-and-electronics-engineers"` | IEEE Reference Guide version 11.29.2023 |
| `"ieee"` | Un alias corto de `institute-of-electrical-and-electronics-engineers` |
| `"institute-of-physics-numeric"` | Institute of Physics (numeric) |
| `"iso-690-author-date"` | ISO-690 (author-date, English) |
| `"iso-690-numeric"` | ISO-690 (numeric, English) |
| `"karger"` | Karger journals |
| `"mary-ann-liebert-vancouver"` | Mary Ann Liebert - NLM/Vancouver |
| `"modern-humanities-research-association-notes"` | MHRA Style Guide 4th edition (notes) |
| `"modern-humanities-research-association"` | Un alias corto de `modern-humanities-research-association-notes` |
| `"modern-language-association"` | MLA Handbook 9th edition (in-text citations) |
| `"mla"` | Un alias corto de `modern-language-association` |
| `"modern-language-association-8"` | Un alias corto de `modern-language-association` |
| `"mla-8"` | Un alias corto de `modern-language-association` |
| `"multidisciplinary-digital-publishing-institute"` | Multidisciplinary Digital Publishing Institute |
| `"nature"` | Nature |
| `"nlm-citation-sequence"` | NLM/Vancouver: Citing Medicine 2nd edition (citation-sequence) |
| `"vancouver"` | Un alias corto de `nlm-citation-sequence` |
| `"nlm-citation-sequence-superscript"` | NLM/Vancouver: Citing Medicine 2nd edition (citation-sequence, superscript) |
| `"vancouver-superscript"` | Un alias corto de `nlm-citation-sequence-superscript` |
| `"pensoft"` | Pensoft Journals |
| `"public-library-of-science"` | Public Library of Science |
| `"plos"` | Un alias corto de `public-library-of-science` |
| `"royal-society-of-chemistry"` | Royal Society of Chemistry |
| `"sage-vancouver"` | SAGE - NLM/Vancouver |
| `"sist02"` | SIST02 (日本語) |
| `"spie"` | SPIE journals |
| `"springer-basic"` | Springer - Basic (numeric, brackets) |
| `"springer-basic-author-date"` | Springer - Basic (author-date) |

| Variante | Detalles |
|---|---|
| `"springer-fachzeitschriften-medizin-psychologie"` | Springer - Fachzeitschriften Medizin Psychologie (Deutsch) |
| `"springer-humanities-author-date"` | Springer - Humanities (author-date) |
| `"springer-lecture-notes-in-computer-science"` | Springer - Lecture Notes in Computer Science |
| `"springer-mathphys"` | Springer - MathPhys (numeric, brackets) |
| `"springer-socpsych-author-date"` | Springer - SocPsych (author-date) |
| `"springer-vancouver"` | Springer - NLM/Vancouver (citation-sequence, brackets) |
| `"taylor-and-francis-chicago-author-date"` | Taylor & Francis Journals Standard Reference Style Guide: Chicago author-date version 2.0 |
| `"taylor-and-francis-national-library-of-medicine"` | Taylor & Francis - NLM/Vancouver (citation--sequence, brackets) |
| `"the-institution-of-engineering-and-technology"` | The Institution of Engineering and Technology |
| `"the-lancet"` | The Lancet |
| `"thieme"` | Thieme-German (Deutsch) |
| `"trends"` | Trends journals |
| `"turabian-author-date"` | Chicago Manual of Style 17th edition (author--date) |
| `"turabian-fullnote-8"` | Chicago Manual of Style 17th edition (notes and bibliography, subsequent author-title) |

/4.5 **target** `auto` or `label` or `selector` or `location` or `function` *Configurable con set*

Por defecto: `auto`

Define qué citas se incluyen en la bibliografía.

Typst asignará automáticamente cada cita del documento a una bibliografía. Concretamente, una cita se asignará a (en orden de precedencia)

1. la primera bibliografía que la incluya en su selector `target`; o, si no existe tal bibliografía,
2. la bibliografía *siguiente* más cercana con `target: auto` que contenga su clave; o, si no sigue ninguna bibliografía así,
3. la bibliografía *anterior* más cercana con `target: auto` que contenga su clave.

Example · Bibliografía local

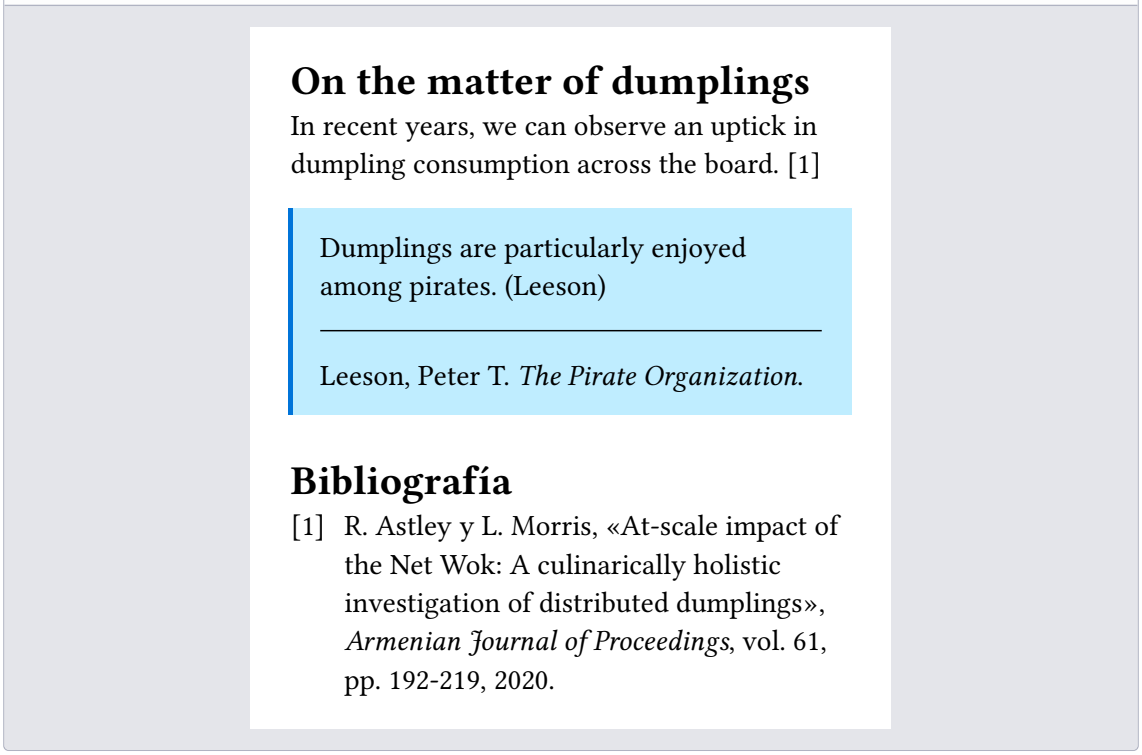
```
#let info(body) = block(
  stroke: (left: 1.5pt + blue),
  fill: aqua.lighten(50%),
  inset: 1em,
  context {
    body
    show divider: set block(spacing: 1.2em)
    divider()
    bibliography(
      "works.bib",
      title: none,
      target: selector(cite).within(here()),
      style: "mla",
    )
  }
)

= On the matter of dumplings
In recent years, we can observe an uptick in
```

```
dumpling consumption across the board. @netwok

#info[
  Dumplings are particularly enjoyed
  among pirates. @arrgh
]

#bibliography("works.bib")
```



## On the matter of dumplings

In recent years, we can observe an uptick in dumpling consumption across the board. [1]

> Dumplings are particularly enjoyed among pirates. (Leeson)
>
> ———
>
> Leeson, Peter T. *The Pirate Organization.*

## Bibliografía

[1] R. Astley y L. Morris, «At-scale impact of the Net Wok: A culinarically holistic investigation of distributed dumplings», *Armenian Journal of Proceedings*, vol. 61, pp. 192-219, 2020.

/4.6 **group** `none` or `auto` or `str` *Configurable con set* Por defecto: `auto`

Agrupa conceptualmente esta bibliografía con otras bibliografías a los fines de la numeración. Las bibliografías del mismo grupo asignarán números de cita consecutivos.

Puede ser:

- `none`: La bibliografía se numerará de forma aislada.
- `auto`: La bibliografía se numerará consecutivamente con todas las demás bibliografías del grupo `auto`.
- Una cadena de texto[3.5.28]: La bibliografía se numerará consecutivamente con todas las demás bibliografías con el mismo valor de `group`.

El grupo `auto` funciona igual que cualquier grupo de cadena, pero es el grupo por defecto canónico.

Example · Números de cita consecutivos

```
#show bibliography: set heading(
  offset: 1,
)

= First part
Starts at one: @netwok @arrgh
#bibliography(
  "works.bib",
  style: "ieee",
)

= Second part
Continues with three: @distress
#bibliography(
  "works.bib",
  style: "nlm-citation-sequence",
)
```

## First part

Starts at one: [1], [2]

### Bibliografía

[1] R. Astley y L. Morris, «At-scale impact of the Net Wok: A culinarically holistic investigation of distributed dumplings», *Armenian Journal of Proceedings*, vol. 61, pp. 192-219, 2020.

[2] P. T. Leeson, «The Pirate Organization».

## Second part

Continues with three: (3)

### Bibliografía

3. An Insight into Bibliographical Distress.

Example · Números de cita separados

```
#show bibliography: set heading(
  offset: 1,
)
#set bibliography(group: none)

= First part
Starts at one: @netwok @arrgh
#bibliography(
  "works.bib",
  style: "ieee",
)

= Second part
Resets to one: @distress
#bibliography(
  "works.bib",
  style: "nlm-citation-sequence",
)
```

## First part

Starts at one: [1], [2]

### Bibliografía

[1] R. Astley y L. Morris, «At-scale impact of the Net Wok: A culinarically holistic investigation of distributed dumplings», *Armenian Journal of Proceedings*, vol. 61, pp. 192-219, 2020.

[2] P. T. Leeson, «The Pirate Organization».

## Second part

Resets to one: (1)

### Bibliografía

1. An Insight into Bibliographical Distress.

### 3.6.2 `list` Elemento Desde: siempre

Una lista con viñetas.

Muestra verticalmente una secuencia de elementos, cada uno introducido por un marcador.

#### /1 Ejemplo

```
Normal list.
- Text
- Math
- Layout
- ...

Multiple lines.
- This list item spans multiple
  lines because it is indented.

Function call.
#list(
  [Foundations],
  [Calculate],
  [Construct],
  [Data Loading],
)
```

Normal list.
- Text
- Math
- Layout
- ...

Multiple lines.
- This list item spans multiple lines because it is indented.

Function call.
- Foundations
- Calculate
- Construct
- Data Loading

#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: empezá una línea con un guion seguido de un espacio para crear un elemento de lista. Un elemento de lista puede contener varios párrafos y otro contenido de bloque. Todo el contenido que tenga más sangría que el marcador de un elemento pasa a formar parte de ese elemento.

#### /3 Parámetros

```
list(
  tight: bool,
  marker: content array function,
  indent: length,
  body-indent: length,
  spacing: auto length,
  marker-align: alignment,
  ..content,
) → content
```

##### /3.1 `tight` `bool` *Configurable con set* Por defecto: `true`

Define el espaciado[3.6.2/3.5] por defecto de la lista. Si es `false`, los elementos se separan con el espaciado de párrafo[3.6.14/4.2]. Si es `true`, en cambio usan el interlineado de párrafo[3.6.14/4.1]. Esto hace que la lista sea más compacta, lo que puede verse mejor si los elementos son cortos.

En modo marcado, el valor de este parámetro se determina según si los elementos están separados por una línea en blanco. Si los elementos se siguen directamente, se establece en `true`; si los elementos están separados por una línea en blanco, se establece en `false`. La compacidad definida en el marcado no se puede reemplazar con reglas set.

```
- If a list has a lot of text, and
  maybe other inline content, it
  should not be tight anymore.

- To make a list wide, simply insert
  a blank line between the items.
```

- If a list has a lot of text, and maybe other inline content, it should not be tight anymore.
- To make a list wide, simply insert a blank line between the items.

/3.2 **marker** `content` or `array` or `function` *Configurable con set* Por defecto: `([•], [‣], [–])`

El marcador que introduce cada elemento.

En lugar de contenido simple, también podés pasar un array con varios marcadores que se usarán para las listas anidadas. Si la profundidad de anidamiento de la lista supera la cantidad de marcadores, los marcadores se repiten cíclicamente. Para tener control total, podés pasar una función que asigne la profundidad de anidamiento de la lista (empezando desde `0`) al marcador deseado.

```
#set list(marker: [--])
- A more classic list
- With en-dashes

#set list(marker: ([•], [--]))
- Top-level
  - Nested
  - Items
- Items
```

– A more classic list
– With en-dashes

- Top-level
  – Nested
  – Items
- Items

/3.3 **indent** `length` *Configurable con set* Por defecto: `0pt`

La sangría de cada elemento.

/3.4 **body-indent** `length` *Configurable con set* Por defecto: `0.5em`

El espaciado entre el marcador y el cuerpo de cada elemento.

/3.5 **spacing** `auto` or `length` *Configurable con set* Por defecto: `auto`

El espaciado entre los elementos de la lista.

Si se establece en `auto`, usa el `leading`[3.6.14/4.1] del párrafo para las listas compactas y el `spacing`[3.6.14/4.2] del párrafo para las listas amplias (no compactas).

/3.6 **marker-align** `alignment` *Configurable con set* Por defecto: end

Alineación que se usa para los marcadores de lista.

La alineación vertical siempre es relativa a la altura de los elementos de la lista a los que están unidos. Por defecto no está especificada, lo que significa que cada marcador se alineará verticalmente con la línea base del elemento al que está unido (que suele ser su primera línea de texto o, si no, su parte superior).

La alineación horizontal, en cambio, es relativa a los otros marcadores del mismo nivel de la lista. Por defecto se establece en `end`, lo que significa que los marcadores se alinean hacia el final de la dirección de texto actual (`right` para texto LTR, `left` para RTL). Sin embargo, como los marcadores de cada nivel suelen ser idénticos, se espera que la alineación horizontal no tenga un efecto real la mayoría de las veces. De todos modos, todavía es posible que marque una diferencia en algunos casos, en particular si los marcadores usan contadores u otras formas de estado para mostrar contenido distinto en cada elemento.

```
- Baseline \
  aligned

#set list(marker-align: horizon)

- Horizon \
  aligned
```

- Baseline aligned
- Horizon aligned

### /3.7 children `content` *Obligatorio* *Posicional* *Variádico*

Los hijos de la lista con viñetas.

Al usar la sintaxis de lista, los elementos adyacentes se reúnen automáticamente en listas, incluso a través de construcciones como los bucles for.

```
#for letter in "ABC" [
  - Letter #letter
]
```

- Letter A
- Letter B
- Letter C

## /4 Funciones

### /4.1 item Elemento Desde: 0.4.0

Un elemento de una lista con viñetas.

```
list.item( content ) → content
```

#### /4.1.1 body `content` *Obligatorio* *Posicional*

El cuerpo del elemento.

### 3.6.3 `cite` Elemento Desde: siempre

Cita una obra de la bibliografía.

Antes de empezar a citar, necesitás agregar una bibliografía[3.6.1] en algún lugar de tu documento.

#### /1 Ejemplo

```
This was already noted by
pirates long ago. @arrgh

Multiple sources say ...
@arrgh @netwok.

You can also call `cite`
explicitly. #cite(<arrgh>)

#bibliography("works.bib")
```

This was already noted by pirates long ago. [1]

Multiple sources say ... [1], [2].

You can also call `cite` explicitly. [1]

**Bibliografía**

[1] P. T. Leeson, «The Pirate Organization».

[2] R. Astley y L. Morris, «At-scale impact of the Net Wok: A culinarically holistic investigation of distributed dumplings», *Armenian Journal of Proceedings*, vol. 61, pp. 192-219, 2020.

Si el nombre de tu fuente contiene ciertos caracteres, como barras, que la sintaxis <> no reconoce, podés llamar explícitamente a `label` en su lugar.

```
Computer Modern is an example of a modernist serif typeface.
#cite(label("DBLP:books/lib/Knuth86a")).
```

#### /2 Sintaxis

Esta función tiene indirectamente una sintaxis dedicada. Las referencias[3.6.17] se pueden usar para citar obras de la bibliografía. La etiqueta corresponde entonces a la clave de la cita.

#### /3 Parámetros

```
cite(
  label,
  supplement: none content,
  form: none str,
  style: auto str path bytes,
) → content
```

##### /3.1 key `label` *Obligatorio* *Posicional*

La clave de cita que identifica la entrada de la bibliografía que se debe citar, como etiqueta.

```
// All the same
@netwok \
#cite(<netwok>) \
#cite(label("netwok"))
```

(Astley & Morris, 2020)
(Astley & Morris, 2020)
(Astley & Morris, 2020)

##### /3.2 supplement `none` or `content` *Configurable con set* Por defecto: none

Un suplemento de la cita, como el número de página o de capítulo.

En la sintaxis de referencias, el suplemento se puede agregar entre corchetes:

```
This has been proven. @distress[p.~7]

#bibliography("works.bib")
```

This has been proven. [1, p. 7]

## Bibliografía

[1] B. Aldrin, «An Insight into Bibliographical Distress».

/3.3 **form** `none` or `str` *Configurable con set* Por defecto: `"normal"`

El tipo de cita que se produce. Las distintas formas son útiles en distintos escenarios: una cita normal es útil como fuente al final de una oración, mientras que una cita "prose" es más adecuada para incluirla en el flujo del texto.

Si se establece en none, la obra citada se incluye en la bibliografía, pero no se mostrará nada.

```
#cite(<netwok>, form: "prose")
show the outsized effects of
pirate life on the human psyche.
```

Astley & Morris (2020) show the outsized effects of pirate life on the human psyche.

| Variante | Detalles |
|---|---|
| `"normal"` | Se muestra de la forma estándar del estilo activo. |
| `"prose"` | Produce una cita adecuada para incluirla dentro de una oración. |
| `"full"` | Imita una entrada de bibliografía, con la información completa de la obra citada. |
| `"author"` | Muestra solo el o los autores de la obra citada. |
| `"year"` | Muestra solo el año de la obra citada. |

/3.4 **style** `auto` or `str` or `path` or `bytes` *Configurable con set* Por defecto: `auto`

El estilo de cita.

Puede ser:

- `auto` para usar automáticamente el estilo de la bibliografía[3.6.1/4.4] para las citas.
- Una cadena de texto con el nombre de uno de los estilos incorporados (ver abajo). Algunos de los estilos listados abajo aparecen dos veces: una con su nombre completo y otra con un alias corto.
- Una cadena de ruta o path[3.5.22] a un archivo CSL[A].
- Bytes sin procesar a partir de los cuales se debe decodificar un estilo CSL.

| Variante | Detalles |
|---|---|
| `"alphanumeric"` | Alfanumérico |
| `"american-anthropological-association"` | American Anthropological Association |
| `"american-chemical-society"` | ACS Guide 2022 revision |
| `"american-geophysical-union"` | American Geophysical Union |
| `"american-institute-of-aeronautics-and-astronautics"` | American Institute of Aeronautics and Astronautics |
| `"american-institute-of-physics"` | AIP Style Manual 4th edition |
| `"american-medical-association"` | AMA Manual of Style 11th edition |
| `"american-meteorological-society"` | American Meteorological Society |
| `"american-physics-society"` | American Physical Society |
| `"american-physiological-society"` | American Physiological Society |
| `"american-political-science-association"` | APSA Style Manual revised 2018 edition |

[A] https://citationstyles.org/

| Variante | Detalles |
|---|---|
| `"american-psychological-association"` | APA Style 7th edition |
| `"apa"` | Un alias corto de `american-psychological-association` |
| `"american-society-for-microbiology"` | American Society for Microbiology |
| `"american-society-of-civil-engineers"` | American Society of Civil Engineers |
| `"american-society-of-mechanical-engineers"` | American Society of Mechanical Engineers |
| `"american-sociological-association"` | ASA Style Guide 6th/7th edition |
| `"angewandte-chemie"` | Angewandte Chemie International Edition |
| `"annual-reviews"` | Annual Reviews (sorted by order of appearance) |
| `"annual-reviews-author-date"` | Annual Reviews (author-date) |
| `"associacao-brasileira-de-normas-tecnicas"` | Associação Brasileira de Normas Técnicas (Português - Brasil) |
| `"association-for-computing-machinery"` | Association for Computing Machinery |
| `"biomed-central"` | BioMed Central |
| `"bristol-university-press"` | Bristol University Press |
| `"british-medical-journal"` | BMJ |
| `"bmj"` | Un alias corto de `british-medical-journal` |
| `"cell"` | Celda |
| `"chicago-author-date"` | Chicago Manual of Style 18th edition (author--date) |
| `"chicago-notes"` | Chicago Manual of Style 18th edition (notes and bibliography) |
| `"chicago-fullnotes"` | Un alias corto de `chicago-notes` |
| `"chicago-shortened-notes"` | Chicago Manual of Style 18th edition (shortened notes and bibliography) |
| `"copernicus"` | Copernicus Publications |
| `"cse-citation-sequence-brackets-8th-edition"` | CSE Manual: Scientific Style and Format 8th edition (citation-sequence, brackets) |
| `"council-of-science-editors"` | Un alias corto de `cse-citation-sequence-brackets-8th-edition` |
| `"cse-name-year"` | CSE Manual: Scientific Style and Format 9th edition (name-year) |
| `"council-of-science-editors-author-date"` | Un alias corto de `cse-name-year` |
| `"current-opinion"` | Current Opinion journals |
| `"deutsche-gesellschaft-für-psychologie"` | Deutsche Gesellschaft für Psychologie 5. Auflage (Deutsch) |
| `"deutsche-sprache"` | Deutsche Sprache (Deutsch) |
| `"elsevier-harvard"` | Elsevier (author-date/Harvard, with titles) |
| `"elsevier-vancouver"` | Elsevier - NLM/Vancouver (citation-sequence) |
| `"elsevier-with-titles"` | Elsevier (numeric, with titles) |
| `"frontiers"` | Frontiers journals |
| `"future-medicine"` | Future Medicine journals |
| `"future-science"` | Future Science Group |
| `"gb-7714-2005-numeric"` | China National Standard GB/T 7714-2005 (numeric, 中文) |
| `"gb-7714-2015-author-date"` | China National Standard GB/T 7714-2015 (author-date, 中文) |

| Variante | Detalles |
|---|---|
| `"gb-7714-2015-note"` | China National Standard GB/T 7714-2015 (note, 中文) |
| `"gb-7714-2015-numeric"` | China National Standard GB/T 7714-2015 (numeric, 中文) |
| `"gost-r-705-2008-numeric"` | Russian GOST R 7.0.5-2008 (numeric) |
| `"harvard-cite-them-right"` | Cite Them Right 12th edition (author-date/ Harvard) |
| `"institute-of-electrical-and-electronics-engineers"` | IEEE Reference Guide version 11.29.2023 |
| `"ieee"` | Un alias corto de `institute-of-electrical-and-electronics-engineers` |
| `"institute-of-physics-numeric"` | Institute of Physics (numeric) |
| `"iso-690-author-date"` | ISO-690 (author-date, English) |
| `"iso-690-numeric"` | ISO-690 (numeric, English) |
| `"karger"` | Karger journals |
| `"mary-ann-liebert-vancouver"` | Mary Ann Liebert - NLM/Vancouver |
| `"modern-humanities-research-association-notes"` | MHRA Style Guide 4th edition (notes) |
| `"modern-humanities-research-association"` | Un alias corto de `modern-humanities-research-association-notes` |
| `"modern-language-association"` | MLA Handbook 9th edition (in-text citations) |
| `"mla"` | Un alias corto de `modern-language-association` |
| `"modern-language-association-8"` | Un alias corto de `modern-language-association` |
| `"mla-8"` | Un alias corto de `modern-language-association` |
| `"multidisciplinary-digital-publishing-institute"` | Multidisciplinary Digital Publishing Institute |
| `"nature"` | Nature |
| `"nlm-citation-sequence"` | NLM/Vancouver: Citing Medicine 2nd edition (citation-sequence) |
| `"vancouver"` | Un alias corto de `nlm-citation-sequence` |
| `"nlm-citation-sequence-superscript"` | NLM/Vancouver: Citing Medicine 2nd edition (citation-sequence, superscript) |
| `"vancouver-superscript"` | Un alias corto de `nlm-citation-sequence-superscript` |
| `"pensoft"` | Pensoft Journals |
| `"public-library-of-science"` | Public Library of Science |
| `"plos"` | Un alias corto de `public-library-of-science` |
| `"royal-society-of-chemistry"` | Royal Society of Chemistry |
| `"sage-vancouver"` | SAGE - NLM/Vancouver |
| `"sist02"` | SIST02 (日本語) |
| `"spie"` | SPIE journals |
| `"springer-basic"` | Springer - Basic (numeric, brackets) |
| `"springer-basic-author-date"` | Springer - Basic (author-date) |
| `"springer-fachzeitschriften-medizin-psychologie"` | Springer - Fachzeitschriften Medizin Psychologie (Deutsch) |
| `"springer-humanities-author-date"` | Springer - Humanities (author-date) |
| `"springer-lecture-notes-in-computer-science"` | Springer - Lecture Notes in Computer Science |

| Variante | Detalles |
| --- | --- |
| `"springer-mathphys"` | Springer - MathPhys (numeric, brackets) |
| `"springer-socpsych-author-date"` | Springer - SocPsych (author-date) |
| `"springer-vancouver"` | Springer - NLM/Vancouver (citation-sequence, brackets) |
| `"taylor-and-francis-chicago-author-date"` | Taylor & Francis Journals Standard Reference Style Guide: Chicago author-date version 2.0 |
| `"taylor-and-francis-national-library-of-medicine"` | Taylor & Francis - NLM/Vancouver (citation--sequence, brackets) |
| `"the-institution-of-engineering-and-technology"` | The Institution of Engineering and Technology |
| `"the-lancet"` | The Lancet |
| `"thieme"` | Thieme-German (Deutsch) |
| `"trends"` | Trends journals |
| `"turabian-author-date"` | Chicago Manual of Style 17th edition (author--date) |
| `"turabian-fullnote-8"` | Chicago Manual of Style 17th edition (notes and bibliography, subsequent author-title) |

### 3.6.4 `divider` Elemento Desde: 0.15.0

Una pausa temática que separa secciones de contenido.

Por defecto, se renderiza como una línea horizontal, pero se puede personalizar mediante reglas show.

#### /1 Ejemplo

```
She left without a word.
#divider()
Three days later, she returned.
```

She left without a word.

---

Three days later, she returned.

#### /2 Estilos

El divisor se puede estilizar mediante reglas show.

Como el divisor se muestra por defecto como una línea, podés usar una regla set para ajustar el trazo de la línea:

```
#show divider: set line(stroke: 2pt + red)
First part
#divider()
Second part
```

First part

---

Second part

También podés reemplazar por completo el divisor con contenido personalizado, como un adorno floral o asteriscos, pero en ese caso conviene envolverlo en un bloque para conservar el espaciado:

```
#show divider: set align(center)
#show divider: block[* * *]
Chapter 1
#divider()
Chapter 2
```

Chapter 1

* * *

Chapter 2

#### /3 Parámetros

```
divider() → content
```

## 3.6.5 `document` Elemento Desde: siempre

Administra metadatos y se usa para agregar un archivo de documento a un bundle.

### /1 Metadatos

El elemento document es la única fuente de verdad de los metadatos del documento. Con él, podés especificar el título, los autores, la fecha, etcétera, del documento en un solo lugar. Normalmente, el elemento se usa con una regla set[3.2.1] como esta:

```
#set document(title: [My doc])

Title is _not_ rendered, but
embedded in PDF metadata.
```

Title is *not* rendered, but embedded in PDF metadata.

Por defecto, los metadatos se incrustan en la salida, pero no se renderizan visiblemente en el documento. Sin embargo, pasan a estar disponibles contextualmente[3.4] en todo el documento y los elementos y plantillas pueden usarlos. Por ejemplo, el elemento incorporado `title`[3.6.21] toma automáticamente el título de documento configurado:

```
#set document(title: [My doc])

#title()
Title is now rendered _and_
embedded in PDF metadata.
```

**My doc**

Title is now rendered *and* embedded in PDF metadata.

De manera similar al elemento `title`, también podés acceder vos mismo a los metadatos con una expresión de contexto[3.4].

```
// In the document.
#set document(
  keywords: ("Typst", "Metadata")
)

// Somewhere in your template.
_Keywords:_
#context document.keywords.join(", ")
```

*Keywords:* Typst, Metadata

En los formatos de exportación de un solo documento, esta función solo se usa con reglas set[3.2.1]. Esas reglas set solo deben aparecer en el nivel superior, no dentro de ningún contenedor de diseño. También podés crear explícitamente un elemento `document`, pero eso solo es relevante en la exportación a bundle[3.6.5/2].

#### /1.1 Consideraciones específicas de cada formato

Los metadatos se incrustan en la salida en distinta medida:

- La exportación a PDF admite toda la gama de metadatos y los emite en el *diccionario de información del documento* del PDF, así como en metadatos XMP.
- La exportación a HTML solo admite las propiedades `title`, `description`, `author` y `keywords`. La propiedad `date` no se admite, ya que el estándar HTML no la prevé.
- La exportación a SVG y PNG no tiene ninguna compatibilidad con metadatos.

### /2 Documentos en la exportación a bundle

En la exportación a bundle[3.14.2], un elemento document representa un único archivo de la salida del bundle, en uno de los otros formatos de exportación de Typst. Al crear un documento, tenés que proporcionar una ruta de salida y algo de contenido. Typst compilará y exportará el contenido proporcionado con el formato

apropiado. Por defecto, el formato se infiere de la extensión de archivo de la ruta que especificaste, pero también podés configurar explícitamente el format[3.6.5/3.2].

```
#document("index.html", title: [Home])[
  #title()
  View #link(<list>)[my famous list].
]

#document("list.html", title: [My Famous List])[
  #title()
  - My
  - Famous
  - List
] <list>
```

## /2.1 Metadatos

Los elementos document toman los metadatos de las reglas `set document` de nivel superior que hay dentro de ellos. Esto significa que los documentos escritos para la exportación de un solo documento se pueden usar con elementos `document` explícitos y conservar correctamente los metadatos.

```
// Will pick up the title defined in `paper.typ`.
#document("paper.pdf", include "paper.typ")
```

```
// paper.typ
#set document(title: [My Paper])
...
```

Tené en cuenta que las reglas set de document dentro de un `document` reemplazan los argumentos explícitos pasados al elemento `document`.

Además, las propiedades configuradas como argumentos explícitos de `document` quedan disponibles contextualmente:

```
#document("index.html", title: [My title])[
  // Both of these will pick up `[My title]`
  #title()
  #context document.title
]
```

## /2.2 Configuración de exportación

Igual que con los metadatos del documento, podés usar reglas set sobre los elementos de formato[3.14] para configurar los ajustes de exportación de un documento en particular.

```
// Pretty-print all HTML.
#set format.html(pretty: true)

#document("index.html")[
  ...
]

#document("archive.pdf")[
  // And use a specific PDF standard
  // for `archive.pdf`.
  #set pdf(standard: "a-2a")
  ...
]
```

## /3 Parámetros

```
document(
  str,
  format: auto str,
  title: none content,
  author: str array,
  description: none content,
  keywords: str array,
  date: none auto datetime,
  content,
) → content
```

### /3.1 path `str` *Obligatorio* *Posicional*

La ruta dentro del bundle donde se ubicará el documento exportado.

Puede contener barras interiores, en cuyo caso los directorios intermedios se crearán automáticamente.

Esta propiedad solo se admite en el destino bundle[3.14.2].

### /3.2 format `auto` or `str` *Configurable con set* Por defecto: `auto`

En qué formato exportar.

Si es `auto`, Typst intenta inferir el formato de exportación a partir de la extensión de archivo de la path[3.6.5/3.1].

Esta propiedad solo se admite en el destino bundle[3.14.2].

| Variante | Detalles |
|---|---|
| `"pdf"` | Formato de documentos y gráficos de alta fidelidad, centrado en la reproducción exacta para impresión. |
| `"png"` | Formato de trama para ilustraciones y gráficos con transparencia. |
| `"svg"` | El formato de gráficos vectoriales de la web. |
| `"html"` | El formato de documentos de la web. |

### /3.3 title `none` or `content` *Configurable con set* Por defecto: `none`

El título del documento. Se renderiza como el título de la ventana del lector de PDF o de la pestaña del navegador de la página.

Por defecto, el título configurado no se renderiza visiblemente en el documento. Podés agregar el título al contenido del documento usando el elemento title[3.6.21], que tomará automáticamente el título configurado acá.

Agregar un título es importante para la accesibilidad, ya que facilita navegar hasta tu documento e identificarlo entre otros documentos abiertos. Al exportar a PDF/UA, el título es obligatorio.

Si bien puede ser contenido arbitrario, los lectores de PDF solo admiten títulos en texto plano, por lo que la conversión podría perder información.

### /3.4 author `str` or `array` *Configurable con set* Por defecto: ()

Los autores del documento.

### /3.5 description `none` or `content` *Configurable con set* Por defecto: `none`

La descripción del documento.

### /3.6 keywords `str` or `array` *Configurable con set* Por defecto: ()

Las palabras clave del documento.

/3.7 **date** `none` or `auto` or `datetime` *Configurable con set* Por defecto: `auto`

La fecha de creación del documento.

Si es `auto` (por defecto), Typst usa la fecha y hora actuales. Establecerlo en `none` evita que Typst incruste cualquier fecha de creación en los metadatos del PDF.

El componente del año debe ser al menos cero para poder incrustarse en un PDF.

Si querés crear PDF reproducibles byte por byte, establecelo en algo distinto de `auto`.

/3.8 **body** `content` *Obligatorio* *Posicional*

El contenido que compone el documento.

Esta propiedad solo se admite en el destino bundle[3.14.2].

### 3.6.6 emph Elemento Desde: siempre

Enfatiza contenido alternando la cursiva.

- Si el estilo de texto[3.7.12/2.3] actual es `"normal"`, lo convierte en `"italic"`.
- Si ya es `"italic"` u `"oblique"`, lo vuelve a convertir en `"normal"`.

#### /1 Ejemplo

```
This is _emphasized._ \
This is #emph[too.]

#show emph: it => {
  text(blue, it.body)
}

This is _emphasized_ differently.
```

This is *emphasized.*
This is *too.*

This is emphasized differently.

#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: para enfatizar contenido, simplemente encerralo entre guiones bajos (_). Tené en cuenta que esto solo funciona en los límites de las palabras. Para enfatizar una parte de una palabra, tenés que usar la función.

#### /3 Parámetros

```
emph(content) → content
```

##### /3.1 body `content` *Obligatorio* *Posicional*

El contenido que se enfatiza.

### 3.6.7 `figure` Elemento Desde: siempre

Una figura con un epígrafe opcional.

Detecta automáticamente su tipo para seleccionar la pista de conteo correcta. Por ejemplo, las figuras que contienen imágenes se numerarán por separado de las que contienen tablas.

## /1 Ejemplos

El siguiente ejemplo muestra una figura básica con una imagen:

```
@glacier shows a glacier. Glaciers
are complex systems.

#figure(
  image("glacier.jpg", width: 80%),
  caption: [A curious figure.],
) <glacier>
```

Figura 1 shows a glacier. Glaciers are complex systems.



Figura 1: A curious figure.

También podés insertar tablas[3.6.19] en figuras para darles un epígrafe. La figura lo detectará y usará automáticamente un contador separado.

```
#figure(
  table(
    columns: 4,
    [t], [1], [2], [3],
    [y], [0.3s], [0.4s], [0.8s],
  ),
  caption: [Timing results],
)
```

| t | 1 | 2 | 3 |
|---|---|---|---|
| y | 0.3s | 0.4s | 0.8s |

Tabla 1: Timing results

Este comportamiento se puede reemplazar especificando explícitamente el `kind` de la figura. Todas las figuras del mismo tipo comparten un contador común.

## /2 Comportamiento de las figuras

Por defecto, las figuras se ubican dentro del flujo del contenido. Para hacer que floten hacia la parte superior o inferior de la página, podés usar el argumento `placement`[3.6.7/5.3].

Si tu figura es demasiado grande y su contenido se puede cortar entre páginas (por ejemplo, si contiene una tabla grande), podés hacer que la propia figura también se pueda cortar entre páginas con esta regla show:

```
#show figure: set block(breakable: true)
```

Mirá la documentación de block[3.10.4/2.3] para más información sobre los bloques que se pueden cortar y los que no.

## /3 Personalización del epígrafe

Podés modificar el aspecto del epígrafe de la figura con su función asociada `caption`[3.6.7/6.1]. En el siguiente ejemplo, enfatizamos todos los epígrafes:

```
#show figure.caption: emph

#figure(
  rect[Hello],
  caption: [I am emphasized!],
)
```

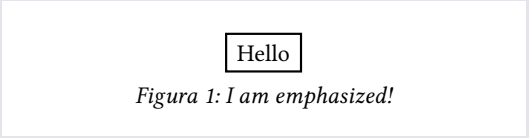



*Figura 1: I am emphasized!*

Con un selector `where`[3.5.16/8.2], podemos limitar esas reglas a tipos específicos de figuras. Por ejemplo, para ubicar el epígrafe encima de las tablas, pero mantenerlo debajo en todos los demás tipos de figuras, podríamos escribir la siguiente regla show-set:

```
#show figure.where(
  kind: table
): set figure.caption(position: top)

#figure(
  table(columns: 2)[A][B][C][D],
  caption: [I'm up here],
)
```

Tabla 1: I'm up here

| | |
|---|---|
| A | B |
| C | D |

## /4 Accesibilidad

Podés usar el parámetro `alt`[3.6.7/5.2] para proporcionar una descripción alternativa[4.4.7] de la figura para lectores de pantalla y otras tecnologías de asistencia (TA). Consultá su documentación[3.6.7/5.2] para aprender más.

Podés usar figuras para agregar descripciones alternativas a rutas, formas o visualizaciones que no tienen su propio parámetro `alt`. Si tu gráfico es puramente decorativo y no tiene un significado semántico, considerá envolverlo en cambio en `pdf.artifact`[3.14.4/4], lo que lo ocultará a las TA al exportar a PDF.

Las TA siempre leerán la figura en el punto donde aparece en el documento, sin importar su `placement`[3.6.7/5.3]. Poné su marcado donde tenga más sentido en el orden de lectura.

## /5 Parámetros

```
figure(
  content,
  alt: none str,
  placement: none auto alignment,
  scope: str,
  caption: none content,
  kind: auto str function,
  supplement: none auto content function,
  numbering: none str function,
  gap: length,
  outlined: bool,
) → content
```

### /5.1 body content *Obligatorio* *Posicional*

El contenido de la figura. A menudo, una imagen[3.11.7].

### /5.2 alt none or str *Configurable con set* Por defecto: none

Una descripción alternativa de la figura.

Cuando agregás una descripción alternativa, las TA leerán tanto ella como el epígrafe (si lo hay). Sin embargo, se omitirá el contenido de la figura en sí.

Cuando el cuerpo de tu figura es una imagen[3.11.7] con su propio texto `alt` establecido, este parámetro no debería usarse en el elemento figure. Del mismo modo, no uses este parámetro cuando la figura contiene

una tabla, código u otro contenido que ya es accesible. En esos casos, las TA leerán el contenido de la figura y agregar una descripción alternativa llevaría a una pérdida de información.

Podés aprender a escribir buenas descripciones alternativas en la guía de accesibilidad[4.4.7].

/5.3 **placement** none or auto or alignment *Configurable con set* Por defecto: none

La ubicación de la figura en la página.

- none: La figura permanece en el flujo, exactamente donde se la especificó, como cualquier otro contenido.
- auto: La figura elige top o bottom según cuál esté más cerca.
- top: La figura flota hacia la parte superior de la página.
- bottom: La figura flota hacia la parte inferior de la página.

El espacio entre el contenido del flujo principal y la figura flotante se controla con el argumento clearance[3.10.19/4.4] de la función place.

```
#set page(height: 200pt)
#show figure: set place(
  clearance: 1em,
)

= Introduction
#figure(
  placement: bottom,
  caption: [A glacier],
  image("glacier.jpg", width: 60%),
)
#lorem(60)
```



Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam

Figura 1: A glacier

aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At.

/5.4 **scope** str *Configurable con set* Por defecto: "column"

En relación con qué ámbito contenedor se ubica la figura.

Establecelo en "parent" para crear una figura de ancho completo en un documento de dos columnas.

No tiene efecto si placement es none.

```
#set page(height: 250pt, columns: 2)

= Introduction
#figure(
  placement: bottom,
  scope: "parent",
  caption: [A glacier],
  image("glacier.jpg", width: 60%),
)
#lorem(60)
```

## Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod



Figura 1: A glacier

aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At.

| Variante | Detalles |
|---|---|
| `"column"` | Se ubica en la columna actual. |
| `"parent"` | Se ubica respecto del elemento padre, y el contenido abarca todas las columnas. |

/5.5 **caption** `none` or `content` *Configurable con set* Por defecto: `none`

El epígrafe de la figura.

/5.6 **kind** `auto` or `str` or `function` *Configurable con set* Por defecto: `auto`

El tipo de figura que es.

Todas las figuras del mismo tipo comparten un contador común.

Si se establece en `auto`, la figura intentará determinar automáticamente su tipo según el tipo de su cuerpo. Los tipos detectados automáticamente son las tablas[3.6.19] y el código[3.7.6]. En los demás casos, el tipo inferido es el de una imagen[3.11.7].

Establecerlo en algo distinto de `auto` reemplazará la detección automática. Esto puede ser útil si
- querés crear un tipo de figura personalizado que no sea una imagen[3.11.7], una tabla[3.6.19] ni código[3.7.6],
- querés forzar a la figura a usar un contador específico sin importar su contenido.

Podés establecer el tipo como una función de elemento o como una cadena de texto. Si lo establecés como una función de elemento distinta de table[3.6.19], raw[3.7.6] o image[3.11.7], vas a tener que especificar manualmente el suplemento de la figura.

Example · Personalizar el tipo de figura

```
#figure(
  circle(radius: 10pt),
  caption: [A curious atom.],
  kind: "atom",
  supplement: [Atom],
)
```



Atom 1: A curious atom.

Si querés modificar un contador para saltear un número o reiniciarlo, podés acceder al contador[3.12.1] de cada tipo de figura con un selector where[3.5.16/8.2]:

- Para tablas[3.6.19]: `counter(figure.where(kind: table))`
- Para imágenes[3.11.7]: `counter(figure.where(kind: image))`
- Para un tipo personalizado: `counter(figure.where(kind: kind))`

Example · Modificar el contador de figuras para tipos específicos

```
#figure(
  table(columns: 2, $n$, $1$),
  caption: [The first table.],
)

#counter(
  figure.where(kind: table)
).update(41)

#figure(
  table(columns: 2, $n$, $42$),
  caption: [The 42nd table],
)

#figure(
  rect[Image],
  caption: [Does not affect images],
)
```

| $n$ | 1 |
|---|---|

Tabla 1: The first table.

| $n$ | 42 |
|---|---|

Tabla 42: The 42nd table

Image

Figura 1: Does not affect images

Para usar cómodamente el contador correcto en una regla show, podés acceder al campo `counter`. Hay un ejemplo de esto en la documentación del campo body del elemento `figure.caption`[3.6.7/6.1.3].

/5.7 **supplement** `none` or `auto` or `content` or `function` *Configurable con set* Por defecto: `auto`

El suplemento de la figura.

Si se establece en `auto`, la figura intentará determinar automáticamente el suplemento correcto según el `kind` y el idioma del texto[3.7.12/2.16] activo. Si usás un tipo de figura personalizado, vas a tener que especificar manualmente el suplemento.

Si se especifica una función, se le pasa el primer descendiente del `kind` especificado (normalmente, el cuerpo de la figura) y debe devolver contenido.

```
#figure(
  [The contents of my figure!],
  caption: [My custom figure],
  supplement: [Bar],
  kind: "foo",
)
```

The contents of my figure!
Bar 1: My custom figure

/5.8 **numbering** `none` or `str` or `function` *Configurable con set* Por defecto: `"1"`

Cómo numerar la figura. Acepta un patrón o función de numeración[3.6.12] que toma un único número.

/5.9 **gap** `length` *Configurable con set* Por defecto: `0.65em`

El espacio vertical entre el cuerpo y el epígrafe.

/5.10 **outlined** `bool` *Configurable con set* Por defecto: `true`

Si la figura debe aparecer en un outline[3.6.13] de figuras.

## /6 Funciones

### /6.1 caption Elemento Desde: 0.8.0

El epígrafe de una figura. Este elemento se puede usar en reglas set y show para personalizar el aspecto de los epígrafes de todas las figuras o de las figuras de un tipo específico.

Además de su position y su body, el caption también proporciona como campos el kind, el supplement, el counter y la numbering de la figura. Estas partes se pueden usar en selectores where[3.5.16/8.2] y en reglas show para construir un epígrafe completamente personalizado.

```
#show figure.caption: emph

#figure(
  rect[Hello],
  caption: [A rectangle],
)
```

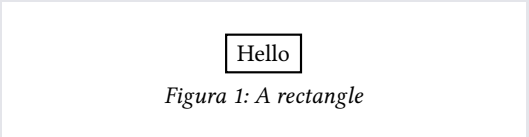



*Figura 1: A rectangle*

```
figure.caption(
  position: alignment,
  separator: auto content,
  content,
) → content
```

#### /6.1.1 position `alignment` *Configurable con set* Por defecto: `bottom`

La posición del epígrafe en la figura. Puede ser top o bottom.

```
#show figure.where(
  kind: table
): set figure.caption(position: top)

#figure(
  table(columns: 2)[A][B],
  caption: [I'm up here],
)

#figure(
  rect[Hi],
  caption: [I'm down here],
)

#figure(
  table(columns: 2)[A][B],
  caption: figure.caption(
    position: bottom,
    [I'm down here too!]
  )
)
```

Tabla 1: I'm up here

| A | B |
|---|---|




Figura 1: I'm down here

| A | B |
|---|---|

Tabla 2: I'm down here too!

#### /6.1.2 separator `auto` or `content` *Configurable con set* Por defecto: `auto`

El separador que aparecerá entre el número y el cuerpo.

Si se establece en auto, el separador se adaptará al idioma[3.7.12/2.16] y a la región[3.7.12/2.17] actuales.

```
#set figure.caption(separator: [ --- ])

#figure(
  rect[Hello],
  caption: [A rectangle],
)
```

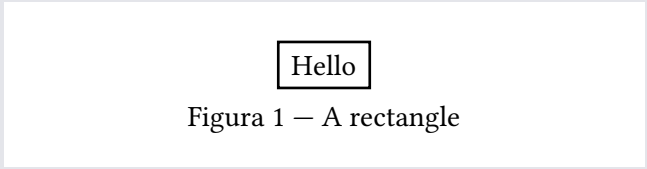



Figura 1 — A rectangle

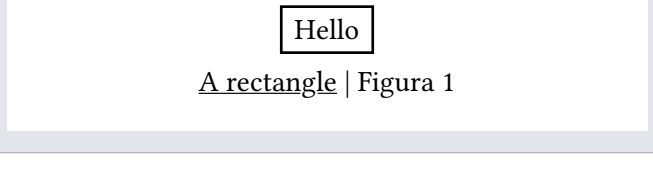
/6.1.3 **body** `content` *Obligatorio* *Posicional*

El cuerpo del epígrafe.

Se puede usar junto con `kind`, `supplement`, `counter`, `numbering` y `location` para personalizar por completo el epígrafe.

```
#show figure.caption: it => [
  #underline(it.body) |
  #it.supplement
  #context it.counter.display(it.numbering)
]

#figure(
  rect[Hello],
  caption: [A rectangle],
)
```




A rectangle | Figura 1

### 3.6.8 `footnote` Elemento Desde: 0.4.0

Una nota al pie.

Incluye observaciones y referencias adicionales en la misma página mediante notas al pie. Una nota al pie insertará un número en superíndice que enlaza con la nota al final de la página. Las notas se numeran secuencialmente a lo largo de todo el documento y se pueden cortar entre varias páginas.

Para personalizar el aspecto de la entrada en el listado de notas al pie, mirá `footnote.entry`[3.6.8/4.1]. La nota al pie en sí se materializa como un superíndice normal, por lo que podés usar una regla set sobre la función `super`[3.7.11] para personalizarla. También podés aplicar una regla show para personalizar solo el marcador de la nota al pie (el número en superíndice) en el texto corrido.

#### /1 Ejemplo

```
Check the docs for more details.
#footnote[https://typst.app/docs]
```

Check the docs for more details.[1]

[1] https://typst.app/docs

La nota al pie se adjunta automáticamente a la palabra anterior, incluso si hay un espacio antes en el marcado. Para forzar un espacio, podés usar la cadena `#" "` o un `espaciado horizontal`[3.10.26] explícito.

Al darle una etiqueta a una nota al pie, podés tener varias referencias a ella.

```
You can edit Typst documents online.
#footnote[https://typst.app/app] <fn>
Checkout Typst's website. @fn
And the online app. #footnote(<fn>)
```

You can edit Typst documents online.[1] Checkout Typst's website.[1] And the online app.[1]

[1] https://typst.app/app

*Nota:* Es posible que las reglas set y show del ámbito donde se llama a `footnote` no se apliquen al contenido de la nota al pie. Mirá `acá`[A] para más información.

#### /2 Accesibilidad

Las tecnologías de asistencia (TA) leerán las notas al pie inmediatamente después del lugar del texto donde se las referencia, tal como aparecen en el marcado.

#### /3 Parámetros

```
footnote(
  numbering: str function,
  label content,
) → content
```

##### /3.1 `numbering` `str` or `function` *Configurable con set* Por defecto: `"1"`

Cómo numerar las notas al pie. Acepta un `patrón o función de numeración`[3.6.12] que toma un único número.

Por defecto, la numeración de las notas al pie continúa a lo largo de todo el documento. Si preferís una numeración de notas al pie por página, podés reiniciar el `contador`[3.12.1] de notas al pie en el `encabezado`[3.10.17/4.13] de la página. En el futuro, podría haber una forma más simple de lograrlo.

[A] https://github.com/typst/typst/issues/1467#issuecomment-1588799440

```
#set footnote(numbering: "*")

Footnotes:
#footnote[Star],
#footnote[Dagger]
```

Footnotes:*,†

*Star
†Dagger

/3.2 **body** `label` or `content` *Obligatorio* *Posicional*

El contenido que se pone en la nota al pie. También puede ser la etiqueta de otra nota al pie a la que esta debe apuntar.

## /4 Funciones

/4.1 **entry** Elemento Desde: 0.4.0

Una entrada en una lista de notas al pie.

Esta función no está pensada para llamarse directamente. En cambio, se usa en reglas set y show para personalizar los listados de notas al pie.

```
#show footnote.entry: set text(red)

My footnote listing
#footnote[It's down here]
has red text!
```

My footnote listing[1] has red text!

[1] It's down here

*Nota:* Las propiedades de las entradas de notas al pie deben ser uniformes en cada serie de páginas (una serie de páginas es una secuencia de páginas sin un salto de página explícito entre ellas). Por este motivo, las reglas set y show de las entradas de notas al pie deberían definirse antes de cualquier contenido de página, normalmente al comienzo del documento.

```
footnote.entry(
  content,
  separator: content,
  clearance: length,
  gap: length,
  indent: length,
) → content
```

/4.1.1 **note** `content` *Obligatorio* *Posicional*

La nota al pie de esta entrada. Su ubicación se puede usar para determinar el estado del contador de notas al pie.

```
#show footnote.entry: it => {
  let loc = it.note.location()
  counter(footnote).display(at: loc, "1: ")
  it.note.body
}

Customized #footnote[Hello]
listing #footnote[World! 🌍]
```

Customized[1] listing[2]

1: Hello
2: World!

### /4.1.2 separator content *Configurable con set* Por defecto: `line(length: 30% + 0pt, stroke: 0.05em)`

El separador entre el cuerpo del documento y el listado de notas al pie.

```
#set footnote.entry(
  separator: repeat[.]
)

Testing a different separator.
#footnote[
  Unconventional, but maybe
  not that bad?
]
```

Testing a different separator.[1]

[1] Unconventional, but maybe not that bad?

### /4.1.3 clearance length *Configurable con set* Por defecto: `1em`

La cantidad de espacio libre entre el cuerpo del documento y el separador.

```
#set footnote.entry(clearance: 3em)

Footnotes also need ...
#footnote[
  ... some space to breathe.
]
```

Footnotes also need ...[1]

[1] ... some space to breathe.

### /4.1.4 gap length *Configurable con set* Por defecto: `0.5em`

El espacio entre las entradas de notas al pie.

```
#set footnote.entry(gap: 0.8em)

Footnotes:
#footnote[Spaced],
#footnote[Apart]
```

Footnotes:[1],[2]

[1] Spaced
[2] Apart

### /4.1.5 indent length *Configurable con set* Por defecto: `1em`

La sangría de cada entrada de nota al pie.

```
#set footnote.entry(indent: 0em)

Footnotes:
#footnote[No],
#footnote[Indent]
```

Footnotes:[1],[2]

[1] No
[2] Indent

## 3.6.9 `heading` Elemento Desde: siempre

Un título de sección.

Con los títulos, podés estructurar tu documento en secciones. Cada título tiene un *nivel*, que empieza en uno y no tiene límite superior. Este nivel indica el rol lógico del contenido que sigue (sección, subsección, etcétera). Un título de nivel superior indica una sección de nivel superior del documento (no el título del documento). Para insertar un título de documento, usá en su lugar el elemento `title`[3.6.21].

Typst puede numerar tus títulos automáticamente. Para habilitar la numeración, especificá cómo querés que se numeren tus títulos con un patrón o función de numeración[3.6.12].

Independientemente de la numeración, Typst también puede generar automáticamente un índice[3.6.13] de todos los títulos. Para excluir uno o más títulos de este índice, podés establecer el parámetro `outlined` en `false`.

Al escribir una regla show[3.2.2] que accede al campo body[3.6.9/5.9] para crear un aspecto completamente personalizado de los títulos, asegurate de envolver el contenido en un `block`[3.10.4] (que es implícitamente sticky[3.10.4/2.13] para los títulos mediante una regla show-set incorporada). Esto evita que los títulos queden "huérfanos", es decir, que queden al final de la página con el contenido siguiente en la página próxima.

### /1 Ejemplo

```
#set heading(numbering: "1.a)")

= Introduction
In recent years, ...

== Preliminaries
To start, ...
```

**1) Introduction**

In recent years, ...

**1.a) Preliminaries**

To start, ...

### /2 Sintaxis

Los títulos tienen una sintaxis dedicada: se pueden crear empezando una línea con uno o varios signos igual, seguidos de un espacio. La cantidad de signos igual determina la profundidad lógica de anidamiento del título. El campo `offset` se puede establecer para configurar la profundidad inicial.

### /3 Accesibilidad

Los títulos son importantes para la accesibilidad, ya que ayudan a los usuarios de tecnologías de asistencia (TA), como los lectores de pantalla, a navegar por tu documento. Los usuarios de lectores de pantalla podrán saltar de título en título u obtener un panorama de todos los títulos del documento.

Para que tus títulos sean accesibles, no deberías saltear niveles de títulos. Esto significa que deberías empezar con un título de primer nivel. Además, cuando el título anterior era de nivel 3, el siguiente debería ser de nivel 3 (manteniendo la misma profundidad), de nivel 4 (bajando exactamente un nivel) o de nivel 1 o 2 (nuevos títulos jerárquicamente superiores).

### /4 Exportación a HTML

Como se mencionó arriba, un título de nivel superior indica una sección de nivel superior del documento y no su título. Esto contrasta con el elemento HTML `<h1>`, del cual debería haber solo uno por documento.

Por este motivo, en la exportación a HTML, un elemento `title`[3.6.21] se convierte en un `<h1>` y los títulos se convierten en `<h2>` y menores (un título de nivel 1 pasa a ser `<h2>`, un título de nivel 2, `<h3>`, etcétera).

### /5 Parámetros

```
heading(
  level: auto int,
```

```
  depth: int,
  offset: int,
  numbering: none str function,
  supplement: none auto content function,
  outlined: bool,
  bookmarked: auto bool,
  hanging-indent: auto length,
  content,
) → content
```

/5.1 **level** `auto` or `int` *Configurable con set* Por defecto: `auto`

La profundidad de anidamiento absoluta del título, empezando desde uno. Si se establece en `auto`, se calcula a partir de `offset + depth`.

Esto es útil principalmente para usar en reglas show[3.2.2] (ya sea con selectores `where`[3.5.16/8.2] o accediendo directamente al nivel de un título mostrado).

```
#show heading.where(level: 2): set text(red)

= Level 1
== Level 2

#set heading(offset: 1)
= Also level 2
== Level 3
```




/5.2 **depth** `int` *Configurable con set* Por defecto: `1`

La profundidad de anidamiento relativa del título, empezando desde uno. Se combina con `offset` para calcular el `level` real.

Esto lo establece la sintaxis de títulos, de modo que **`== Heading`** crea un título con profundidad lógica 2, pero nivel real `offset + 2`. Si construís un título manualmente, normalmente conviene preferir esto en lugar de establecer el nivel absoluto.

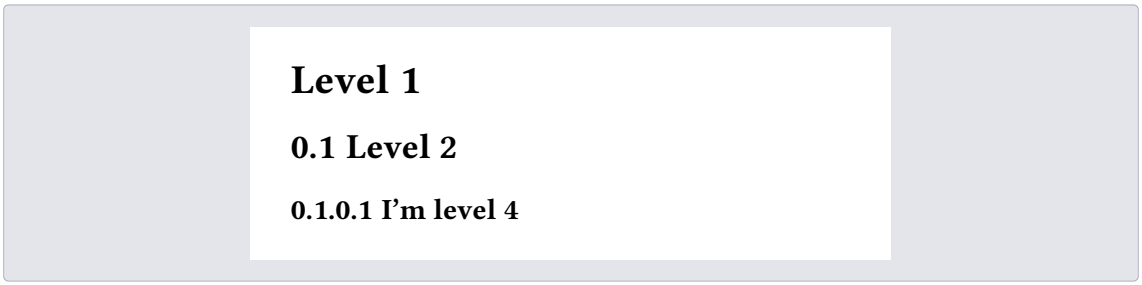
/5.3 **offset** `int` *Configurable con set* Por defecto: `0`

El desplazamiento inicial del `level` de cada título, que se usa para convertir su `depth` relativo en su `level` absoluto.

```
= Level 1

#set heading(offset: 1, numbering: "1.1")
= Level 2

#heading(offset: 2, depth: 2)[
  I'm level 4
]
```

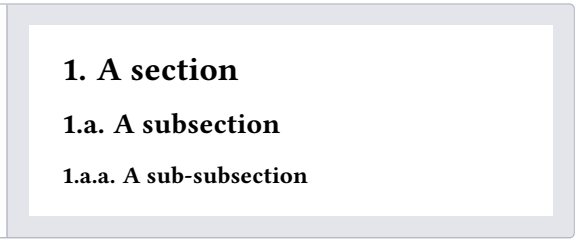



/5.4 **numbering** `none` or `str` or `function` *Configurable con set* Por defecto: `none`

Cómo numerar el título. Acepta un patrón o función de numeración[3.6.12] que toma varios números.

```
#set heading(numbering: "1.a.")

= A section
== A subsection
=== A sub-subsection
```




/5.5 **supplement** `none` or `auto` or `content` or `function` *Configurable con set* Por defecto: `auto`

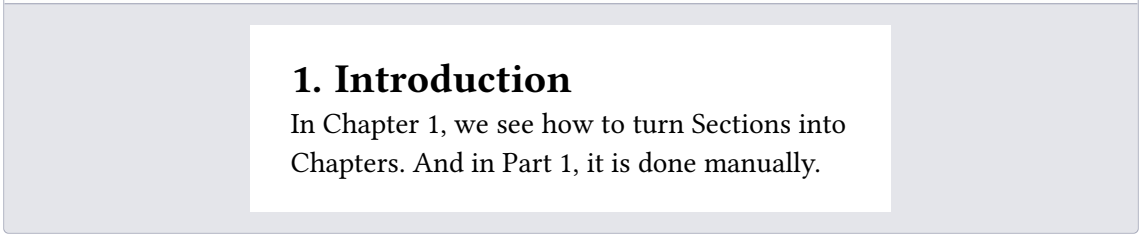
Un suplemento para el título.

En las referencias a títulos, se agrega antes del número referenciado.

Si se especifica una función, se le pasa el título referenciado y debe devolver contenido.

```
#set heading(numbering: "1.", supplement: [Chapter])

= Introduction <intro>
In @intro, we see how to turn
Sections into Chapters. And
in @intro[Part], it is done
manually.
```




/5.6 **outlined** `bool` *Configurable con set* Por defecto: `true`

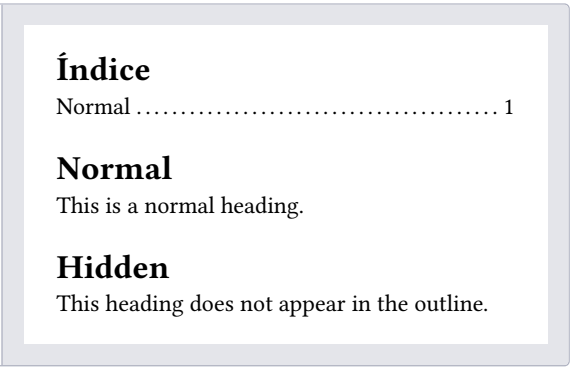
Si el título debe aparecer en el índice[3.6.13].

Tené en cuenta que esta propiedad, si se establece en `true`, asegura que el título también se muestre como marcador en el esquema del PDF exportado (al exportar a PDF). Para cambiar ese comportamiento, usá la propiedad `bookmarked`.

```
#outline()

#heading[Normal]
This is a normal heading.

#heading(outlined: false)[Hidden]
This heading does not appear
in the outline.
```



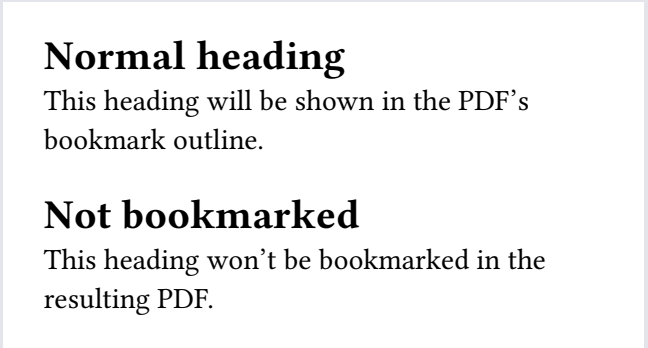

/5.7 **bookmarked** `auto` or `bool` *Configurable con set* Por defecto: `auto`

Si el título debe aparecer como marcador en el esquema del PDF exportado. No afecta a otros formatos de exportación, como PNG.

El valor por defecto `auto` indica que el título solo aparecerá en el esquema del PDF exportado si su propiedad `outlined` está establecida en `true`, es decir, si también aparecería en el índice[3.6.13] de Typst. Establecer esta propiedad en `true` (con marcador) o en `false` (sin marcador) omite ese comportamiento.

```
#heading[Normal heading]
This heading will be shown in
the PDF's bookmark outline.

#heading(bookmarked: false)[Not bookmarked]
This heading won't be
bookmarked in the resulting
PDF.
```



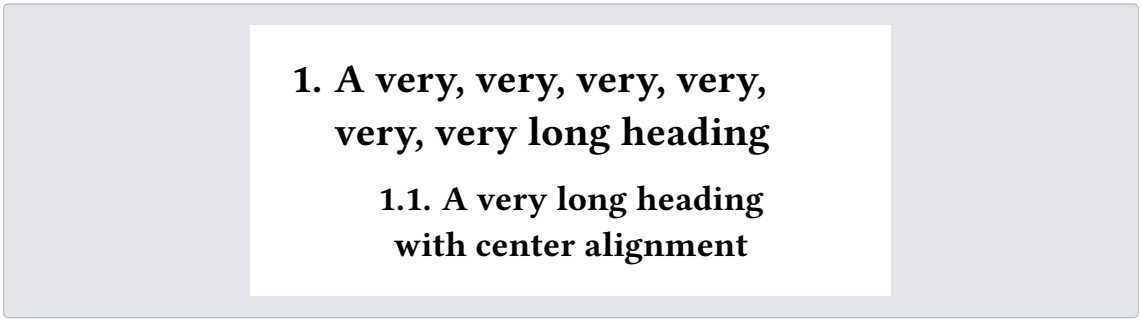

/5.8 **hanging-indent** `auto` or `length` *Configurable con set* Por defecto: `auto`

La sangría que deben tener todas las líneas de un título excepto la primera.

El valor por defecto `auto` usa el ancho de la numeración como sangría si el título está alineado al inicio[3.10.8/2.5] de la dirección del texto[3.7.12/2.19], y ninguna sangría para la alineación centrada y otras alineaciones.

```
#set heading(numbering: "1.")
= A very, very, very, very, very, very long heading

#show heading: set align(center)
== A very long heading\ with center alignment
```




/5.9 **body** `content` *Obligatorio* *Posicional*

El título del encabezado.

### 3.6.10 `link` Elemento Desde: siempre

Enlaces a una URL o a una ubicación del documento.

Por defecto, los enlaces no se ven distintos del texto normal. Sin embargo, podés aplicar fácilmente un estilo a tu elección con una regla show.

#### /1 Ejemplo

```
#show link: underline

https://example.com \

#link("https://example.com") \
#link("https://example.com")[
  See example.com
]
```

https://example.com

https://example.com
See example.com

#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: el texto que empieza con `http://` o `https://` se convierte automáticamente en un enlace.

Para evitar la creación automática de un enlace, podés poner el texto en una cadena de texto[3.5.28]. Para incrustar la cadena en el marcado, anteponele un numeral. Como alternativa, si el texto con forma de enlace es código informático, también podés ponerlo en un elemento `raw`[3.7.6]. Tené en cuenta que, en ambos casos, el texto puede seguir siendo cliqueable en el PDF, porque algunos lectores de PDF detectan enlaces automáticamente.

```
#show link: set text(blue)

// Automatic link
https://example.com

// String, not a link
#"https://example.com"

// Raw, not a link
`https://*.com`
```

https://example.com

https://example.com

`https://*.com`

#### /3 Separación en sílabas

Si activás la separación en sílabas o la justificación, por defecto no se aplicará a los enlaces, para evitar separaciones no deseadas en las URL. Podés desactivar este comportamiento por defecto con `show link: set text(hyphenate: true)`.

#### /4 Accesibilidad

El destino de un enlace debería quedar claro a partir del propio texto del enlace o, al menos, del texto que lo rodea inmediatamente. En la exportación a PDF, Typst generará automáticamente una descripción emergente para los enlaces según su destino. Para los enlaces a URL, se usará la propia URL como descripción emergente.

#### /5 Enlaces en la exportación a HTML

En la exportación a HTML[3.14.3], un enlace a una etiqueta[3.5.18] o ubicación[3.12.4] se convertirá en un enlace de fragmento a un punto de anclaje con nombre. Para admitirlo, los destinos sin un ID existente recibirán automáticamente un ID en el DOM. Cómo funciona esto varía según el tipo de nodo(s) HTML en que se convirtió el destino del enlace:

- Si el destino del enlace se convirtió en un único elemento HTML, ese elemento recibirá el ID. Este es, por ejemplo, el caso típico cuando se enlaza a un título de nivel superior (que se convierte en un único elemento `<h2>`).
- Si el destino del enlace se convirtió en un único nodo de texto, el nodo se envolverá en un `<span>`, que recibirá el ID.
- Si el destino del enlace se convirtió en varios nodos, el primer nodo recibirá el ID.
- Si el destino del enlace no se convirtió en ningún nodo, se generará un span vacío que sirva como destino del enlace.

Si dependés de una estructura de DOM específica, deberías asegurarte de que el destino del enlace se convierta en uno o varios elementos, ya que el compilador no da garantías sobre la segmentación precisa del texto en nodos de texto.

Si está presente, la generación automática de ID intenta reutilizar la etiqueta del destino del enlace para crear un ID legible. Una etiqueta se puede reutilizar si:

- Todos los caracteres son alfabéticos o numéricos según Unicode, o un guion, o un guion bajo.
- La etiqueta no empieza con un dígito ni con un guion.

Estas reglas aseguran que la etiqueta sea tanto un identificador CSS válido como un fragmento de URL válido para enlazar.

Como los ID deben ser únicos en el DOM, las etiquetas duplicadas podrían necesitar desambiguación al reutilizarlas como ID. Las reglas precisas para esto son las siguientes:

- Si una etiqueta se puede reutilizar y es única en el documento, se usará directamente como ID.
- Si es reutilizable, pero no única, se agregará un sufijo formado por un guion y un entero. Por ejemplo, si la etiqueta `<mylabel>` existe dos veces, se convertiría en `mylabel-1` y `mylabel-2`.
- De lo contrario, se generará un ID único con la forma `loc-` seguido de un entero.

## /6 Enlaces en la exportación a bundle

En la exportación a bundle[3.14.2], los enlaces siguen funcionando como siempre. Por ejemplo, si le adjuntás una etiqueta a un elemento de un documento, los enlaces de otros documentos pueden hacer referencia a esa etiqueta. Además, los documentos y los recursos (assets) también se pueden enlazar directamente. Para enlazar a un documento o recurso completo, podés adjuntarle una etiqueta o hacer una consulta[3.12.6] de él y extraer su ubicación[3.12.4].

```
#document("index.html")[
  // Link to document.
  #link(<appendix>)[To appendix]

  // Link into document.
  See the #link(<glossary>)[Glossary]
  for more information.
]

#document("appendix.html")[
  = Definitions
  ...

  = Glossary <glossary>
  ...
] <appendix>
```

Los enlaces entre documentos se emiten como rutas relativas (potencialmente con fragmentos). Typst asigna automáticamente nombres de anclaje por documento según las mismas reglas que en la exportación a HTML. En los documentos HTML y SVG, se emiten como atributos `id` en los elementos. En los documentos PDF, se emiten como *destinos con nombre*. Los documentos PNG no admiten enlaces.

Tené en cuenta que los enlaces siempre usan rutas relativas completas. En algunos escenarios (principalmente en sitios web de varias páginas), esto puede no ser deseable. Por ejemplo, podrías querer generar un documento `/blog/index.html` y a la vez enlazarlo simplemente como `/blog`. Además, tu servidor web podría tratar `/blog` y `/blog/` como intercambiables y servir `/blog/index.html` en ambos casos. Si entonces un usuario navega a `/blog`, los enlaces relativos a otras páginas generadas por Typst dejarán de funcionar. Actualmente, Typst no tiene una forma de engancharse directamente en el manejo de enlaces incorporado. Dicho esto, en la exportación a HTML, según tu caso de uso, puede ser posible ajustar el manejo de enlaces incorporado con una regla show sobre `html.elem.where(tag: "a")`.

## /7 Parámetros

```
link(
  str label location dictionary,
  content,
) → content
```

### /7.1 dest `str` or `label` or `location` or `dictionary` *Obligatorio* *Posicional*

El destino al que apunta el enlace.

- Para enlazar a páginas web, `dest` debe ser una cadena de URL válida. Si la URL tiene el esquema `mailto:` o `tel:` y se omite el parámetro `body`, el cuerpo del enlace será la dirección de correo electrónico o el número de teléfono, sin el esquema.
- Para enlazar a otra parte del documento, `dest` puede tomar una de tres formas:
  - Una etiqueta[3.5.18] adjunta a un elemento. Si además querés texto automático para el enlace basado en el elemento, considerá usar en cambio una referencia[3.6.17].
  - Una `location`[3.12.4] (normalmente obtenida de `here`[3.12.2], `locate`[3.12.3] o `query`[3.12.6]).
  - Un diccionario con una clave `page` de tipo entero[3.5.17] y coordenadas `x` e `y` de tipo longitud[3.10.13]. Las páginas se cuentan desde uno y las coordenadas son relativas a la esquina superior izquierda de la página.

```
= Introduction <intro>
#link("mailto:hello@typst.app") \
#link(<intro>)[Go to intro] \
#link((page: 1, x: 0pt, y: 0pt))[
  Go to top
]
```

**Introduction**
hello@typst.app
Go to intro
Go to top

### /7.2 body `content` *Obligatorio* *Posicional*

El contenido que debe convertirse en un enlace.

Si `dest` es una cadena de URL, el parámetro se puede omitir. En ese caso, la URL se mostrará como enlace.

### 3.6.11 `enum` Elemento Desde: siempre

Una lista numerada.

Muestra verticalmente una secuencia de elementos y los numera consecutivamente.

## /1 Ejemplo

```
Automatically numbered:
+ Preparations
+ Analysis
+ Conclusions

Manually numbered:
2. What is the first step?
5. I am confused.
+  Moving on ...

Multiple lines:
+ This enum item has multiple
  lines because the next line
  is indented.

Function call.
#enum[First][Second]
```

Automatically numbered:
1. Preparations
2. Analysis
3. Conclusions

Manually numbered:
2. What is the first step?
5. I am confused.
6. Moving on ...

Multiple lines:
1. This enum item has multiple lines because the next line is indented.

Function call.
1. First
2. Second

Podés cambiar fácilmente todas tus enumeraciones a otro estilo de numeración con una regla set.

```
#set enum(numbering: "a)")

+ Starting off ...
+ Don't forget step two
```

a) Starting off ...
b) Don't forget step two

También podés usar enum.item[3.6.11/4.1] para personalizar programáticamente el número de cada elemento de la enumeración:

```
#enum(
  enum.item(1)[First step],
  enum.item(5)[Fifth step],
  enum.item(10)[Tenth step]
)
```

1. First step
5. Fifth step
10. Tenth step

## /2 Sintaxis

Esta función también tiene una sintaxis dedicada:

- Empezar una línea con un signo más crea un elemento de enumeración numerado automáticamente.
- Empezar una línea con un número seguido de un punto crea un elemento de enumeración numerado explícitamente.

Los elementos de una enumeración pueden contener varios párrafos y otro contenido de bloque. Todo el contenido que tenga más sangría que el marcador de un elemento pasa a formar parte de ese elemento.

## /3 Parámetros

```
enum(
  tight: bool,
  numbering: str function,
```

```
  start: auto int,
  full: bool,
  reversed: bool,
  indent: length,
  body-indent: length,
  spacing: auto length,
  number-align: alignment,
  ..content,
) → content
```

### /3.1 tight bool *Configurable con set* Por defecto: true

Define el espaciado[3.6.11/3.8] por defecto de la enumeración. Si es `false`, los elementos se separan con el espaciado de párrafo[3.6.14/4.2]. Si es `true`, en cambio usan el interlineado de párrafo[3.6.14/4.1]. Esto hace que la lista sea más compacta, lo que puede verse mejor si los elementos son cortos.

En modo marcado, el valor de este parámetro se determina según si los elementos están separados por una línea en blanco. Si los elementos se siguen directamente, se establece en `true`; si los elementos están separados por una línea en blanco, se establece en `false`. La compacidad definida en el marcado no se puede reemplazar con reglas set.

```
+ If an enum has a lot of text, and
  maybe other inline content, it
  should not be tight anymore.

+ To make an enum wide, simply
  insert a blank line between the
  items.
```

1. If an enum has a lot of text, and maybe other inline content, it should not be tight anymore.
2. To make an enum wide, simply insert a blank line between the items.

### /3.2 numbering str or function *Configurable con set* Por defecto: "1."

Cómo numerar la enumeración. Acepta un patrón o función de numeración[3.6.12].

Si el patrón de numeración contiene varios símbolos de conteo, se aplican a las enumeraciones anidadas. Si se da una función, la función recibe un argumento si `full` es `false` y varios argumentos si `full` es `true`.

```
#set enum(numbering: "1.a)")
+ Different
+ Numbering
  + Nested
  + Items
+ Style

#set enum(numbering: n => super[#n])
+ Superscript
+ Numbering!
```

1) Different
2) Numbering
   a) Nested
   b) Items
3) Style

[1] Superscript
[2] Numbering!

### /3.3 start auto or int *Configurable con set* Por defecto: auto

Con qué número empieza la enumeración.

```
#enum(
  start: 3,
  [Skipping],
  [Ahead],
)
```

3. Skipping
4. Ahead

/3.4 **full** `bool` *Configurable con set* Por defecto: `false`

Si se muestra la numeración completa, incluidos los números de todas las enumeraciones padre.

```
#set enum(numbering: "1.a)", full: true)
+ Cook
  + Heat water
  + Add ingredients
+ Eat
```

1) Cook
   1.a) Heat water
   1.b) Add ingredients
2) Eat

/3.5 **reversed** `bool` *Configurable con set* Por defecto: `false`

Si se invierte la numeración de esta enumeración.

```
#set enum(reversed: true)
+ Coffee
+ Tea
+ Milk
```

3. Coffee
2. Tea
1. Milk

/3.6 **indent** `length` *Configurable con set* Por defecto: `0pt`

La sangría de cada elemento.

/3.7 **body-indent** `length` *Configurable con set* Por defecto: `0.5em`

El espacio entre la numeración y el cuerpo de cada elemento.

/3.8 **spacing** `auto` or `length` *Configurable con set* Por defecto: `auto`

El espaciado entre los elementos de la enumeración.

Si se establece en `auto`, usa el `leading`[3.6.14/4.1] del párrafo para las enumeraciones compactas y el `spacing`[3.6.14/4.2] del párrafo para las enumeraciones amplias (no compactas).

/3.9 **number-align** `alignment` *Configurable con set* Por defecto: end

La alineación que deben tener los números de la enumeración.

Por defecto se establece en end, lo que alinea los números de la enumeración horizontalmente hacia el final de la dirección de texto actual (en una escritura de izquierda a derecha, por ejemplo, equivale a `right`). Además, la ausencia de una alineación vertical ubica cada número verticalmente justo encima de la línea base del elemento, como si fuera parte de su primera línea de texto.

La elección de end para la alineación horizontal de los números de la enumeración suele preferirse a `start`, ya que así los números crecen alejándose del texto en lugar de hacia él. Sin embargo, esta opción te permite reemplazar ese comportamiento.

En cuanto a la alineación vertical, se puede reemplazar si no se desea la alineación con la línea base. Por ejemplo, una alineación de end `+` top siempre ubicaría el marcador verticalmente cerca de la parte superior del elemento, mientras que end `+`
bottom lo movería cerca de la parte inferior.

También hay que notar que la lista sin orden[3.6.2] tiene una opción similar, llamada en cambio `marker-align`, que también controla ambos ejes de la alineación del marcador exactamente de la misma manera que los números de enum.

```
#set enum(number-align: start + bottom)

Here are some powers of two:
1. One
2. Two
4. Four
8. Eight
16. Sixteen
32. Thirty two
```

Here are some powers of two:

1. One
2. Two
4. Four
8. Eight
16. Sixteen
32. Thirty two

/3.10 **children** `content` *Obligatorio* *Posicional* *Variádico*

Los elementos de la lista numerada.

Al usar la sintaxis de enum, los elementos adyacentes se reúnen automáticamente en enumeraciones, incluso a través de construcciones como los bucles for.

```
#for phase in (
  "Launch",
  "Orbit",
  "Descent",
) [+ #phase]
```

1. Launch
2. Orbit
3. Descent

## /4 Funciones

/4.1 **item** Elemento Desde: 0.4.0

Un elemento de enumeración.

```
enum.item(
  auto int,
  content,
) → content
```

/4.1.1 **number** `auto` or `int` *Posicional* *Configurable con set* Por defecto: `auto`

El número del elemento.

/4.1.2 **body** `content` *Obligatorio* *Posicional*

El cuerpo del elemento.

## 3.6.12 `numbering` Desde: siempre

Aplica una numeración a una secuencia de números.

Una numeración define cómo se debe mostrar como contenido una secuencia de números. Se define mediante una cadena de patrón o una función arbitraria.

Un patrón de numeración consiste en símbolos de conteo, en los que se sustituye el número real, sus prefijos y un sufijo. Los prefijos y el sufijo se muestran tal cual.

### /1 Ejemplo

```
#numbering("1.1)", 1, 2, 3) \
#numbering("1.a.i", 1, 2) \
#numbering("I - 1", 12, 2) \
#numbering(
  (..nums) => nums
    .pos()
    .map(str)
    .join(".") + ")",
  1, 2, 3,
)
```

1.2.3)
1.b
XII – 2
1.2.3)

### /2 Patrones de numeración y funciones de numeración

Hay varias situaciones en las que podés proporcionar un patrón o una función de numeración en Typst. Por ejemplo, al definir cómo numerar los títulos[3.6.9] o las figuras[3.6.7]. Cada vez, el formato esperado es el mismo que se describe abajo para el parámetro `numbering`[3.6.12/3.1].

El siguiente ejemplo ilustra que una función de numeración es simplemente una función[3.5.16] normal que acepta números y devuelve `content`[3.5.8].

```
#let unary(.., last) = "|" * last
#set heading(numbering: unary)
= First heading
= Second heading
= Third heading
```

| First heading

|| Second heading

||| Third heading

### /3 Parámetros

```
numbering(
  str function,
  .. int,
) → any
```

#### /3.1 `numbering` `str` or `function` *Obligatorio* *Posicional*

Define cómo funciona la numeración.

Los **símbolos de conteo** son `1`, `a`, `A`, `i`, `I`, α, Α, 一, 壹, あ, い, ア, イ, ㄅ, 가, ㄱ, *, ١, ۱, ٩, ๑, ①, ⓵ y ⓪. Se reemplazan por el número de la secuencia, conservando las mayúsculas y minúsculas originales.

El carácter `*` significa que se deben usar símbolos para contar, en el orden *, †, ‡, §, ¶, ‖. Si hay más de seis elementos, el número se representa con símbolos repetidos.

Los **sufijos** son todos los caracteres posteriores al último símbolo de conteo. Se muestran tal cual al final de cualquier número renderizado.

Los **prefijos** son todos los caracteres que no son ni símbolos de conteo ni sufijos. Se muestran tal cual delante del equivalente renderizado de su símbolo de conteo.

Este parámetro también puede ser una función arbitraria que reciba cada número como un argumento individual. Cuando se le da una función, la función `numbering` simplemente le reenvía los argumentos. Si bien esto no es particularmente útil en sí mismo, significa que podés darle numeraciones arbitrarias a la función `numbering` sin preocuparte de si están definidas como patrón o como función.

/3.2 **`numbers`** `int` *Obligatorio* *Posicional* *Variádico*

Los números a los que se aplica la numeración. Deben ser no negativos.

En general, los números se cuentan desde uno. Un número cero indica que el primer elemento todavía no apareció.

Si `numbering` es un patrón y se dan más números que símbolos de conteo, se repite el último símbolo de conteo con su prefijo.

### 3.6.13 `outline` Elemento Desde: siempre

Un índice de contenidos, de figuras u otros elementos.

Esta función genera una lista de todas las apariciones de un elemento en el documento, hasta una `depth`[3.6.13/4.3] dada. La numeración y el número de página del elemento se mostrarán en el índice junto con su título o epígrafe.

#### /1 Ejemplo

```
#set heading(numbering: "1.")
#outline()

= Introduction
#lorem(5)

= Methods
== Setup
#lorem(10)
```

**Índice**


1. Introduction ..... 1
2. Methods ..... 1
   2.1. Setup ..... 1


## 1. Introduction

Lorem ipsum dolor sit amet.

## 2. Methods

### 2.1. Setup

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

#### /2 Índices alternativos

En su configuración por defecto, esta función genera un índice de contenidos. Al establecer el parámetro `target`, el índice se puede usar para generar una lista de otros tipos de elementos distintos de los títulos.

En el siguiente ejemplo, listamos todas las figuras que contienen imágenes estableciendo `target` en `figure.where(kind: image)`. De la misma manera, podríamos haberlo establecido en `figure.where(kind: table)` para generar una lista de tablas.

También podríamos establecerlo simplemente en `figure`, sin usar un selector `where`[3.5.16/8.2], pero entonces la lista contendría *todas* las figuras, ya sean las que contienen imágenes, tablas u otro material.

```
#outline(
  title: [List of Figures],
  target: figure.where(kind: image),
)

#figure(
  image("tiger.jpg"),
  caption: [A nice figure!],
)
```

**List of Figures**


Figura 1 A nice figure! ..... 1




Figura 1: A nice figure!

## /3 Dar estilo al índice

En el nivel más básico, podés dar estilo al índice estableciendo propiedades en él y en sus entradas. De esta manera, podés personalizar el título[3.6.13/4.1] del índice, cómo se indentan[3.6.13/4.4] las entradas del índice y cómo debe rellenarse[3.6.13/5.1.3] el espacio entre el texto de una entrada y su número de página.

Es posible una personalización más rica mediante la configuración de las entradas[3.6.13/5.1] del índice. El índice genera una entrada por cada elemento incluido.

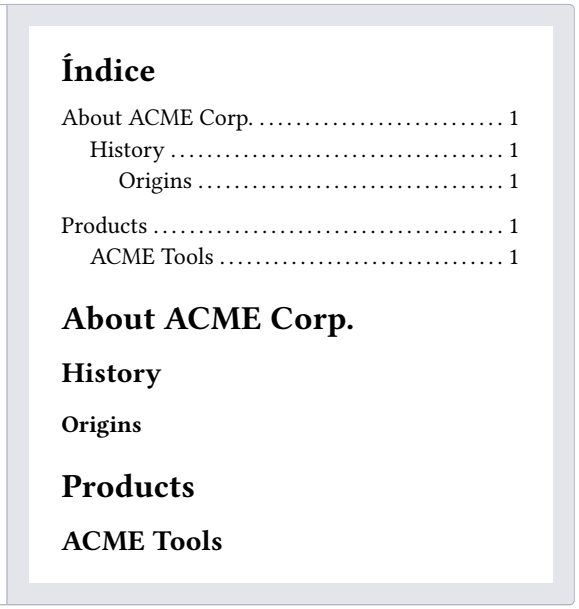
### /3.1 Espaciar las entradas

Las entradas del índice son bloques[3.10.4], así que podés ajustar el espaciado entre ellas con las reglas normales de espaciado de bloques:

```
#show outline.entry.where(
  level: 1
): set block(above: 1.2em)

#outline()

= About ACME Corp.
== History
=== Origins
= Products
== ACME Tools
```




### /3.2 Armar una entrada del índice a partir de sus partes

Para tener control total, también podés escribir una regla show de transformación sobre `outline.entry`. Sin embargo, la lógica para dar formato e indentar correctamente las entradas del índice es bastante compleja y la entrada del índice en sí solo contiene dos campos: el nivel y el elemento incluido.

Por este motivo, se proporcionan varias funciones auxiliares. Podés combinarlas para componer una entrada solo con las partes que te gusten.

La regla show por defecto de una entrada del índice se ve así[A]:

```
#show outline.entry: it => link(
  it.element.location(),
  it.indented(it.prefix(), it.inner()),
)
```

- La función `indented`[3.6.13/5.1.4.1] toma un prefijo opcional y un contenido interno y les aplica automáticamente la sangría adecuada, de modo que las distintas entradas queden bien alineadas y los títulos largos se ajusten correctamente.
- La función `prefix`[3.6.13/5.1.4.2] da formato a la numeración del elemento (si la hay). También agrega un suplemento para ciertos elementos.
- La función `inner`[3.6.13/5.1.4.3] combina el body[3.6.13/5.1.4.4] del elemento, el relleno y el número de page[3.6.13/5.1.4.5].

Podés usar estas funciones individuales para dar formato a la entrada del índice de distintas maneras. Supongamos que querés eliminar por completo el relleno y los números de página. Para lograrlo, podrías escribir una regla show como esta:

[A] El índice de ecuaciones es la excepción a esta regla, ya que no tiene cuerpo y, por lo tanto, no usa un diseño con sangría.

```typ
#show outline.entry: it => link(
  it.element.location(),
  // Keep just the body, dropping
  // the fill and the page.
  it.indented(it.prefix(), it.body()),
)

#outline()

= About ACME Corp.
== History
```




## /4 Parámetros

```typ
outline(
  title: none auto content,
  target: label selector location function,
  depth: none int,
  indent: auto relative function,
) → content
```

### /4.1 title

none or auto or content — *Configurable con set* — Por defecto: auto

El título del índice.

- Cuando se establece en auto, se usará un título apropiado para el idioma del texto[3.7.12/2.16].
- Cuando se establece en none, el índice no tendrá título.
- Se puede establecer un título personalizado pasando contenido.

El título del índice no estará numerado por defecto, pero podés forzarlo con una regla show-set: `show outline: set heading(numbering: "1.")`

### /4.2 target

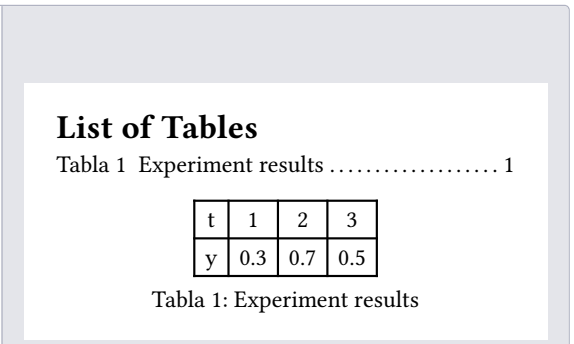
label or selector or location or function — *Configurable con set* — Por defecto: heading

El tipo de elemento que se incluye en el índice.

Para listar figuras que contienen un tipo específico de elemento, como una imagen o una tabla, podés especificar el tipo deseado en un selector where[3.5.16/8.2]. Mirá la sección sobre índices alternativos[3.6.13/2] para más detalles.

```typ
#outline(
  title: [List of Tables],
  target: figure.where(kind: table),
)

#figure(
  table(
    columns: 4,
    [t], [1], [2], [3],
    [y], [0.3], [0.7], [0.5],
  ),
  caption: [Experiment results],
)
```




### /4.3 depth

none or int — *Configurable con set* — Por defecto: none

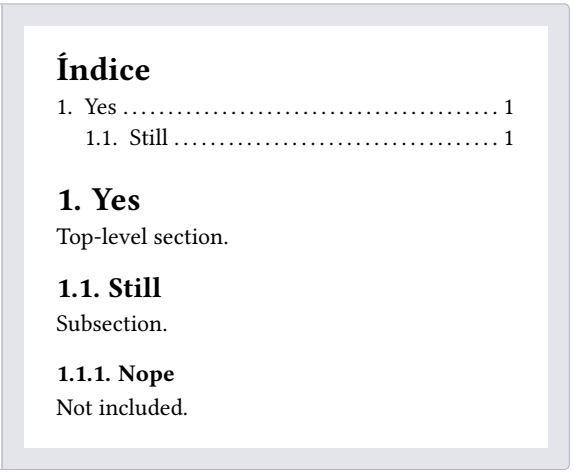
El nivel máximo hasta el cual se incluyen elementos en el índice. Cuando este argumento es none, se incluyen todos los elementos.

```
#set heading(numbering: "1.")
#outline(depth: 2)

= Yes
Top-level section.

== Still
Subsection.

=== Nope
Not included.
```




/4.4 **indent** `auto` or `relative` or `function` *Configurable con set* Por defecto: `auto`

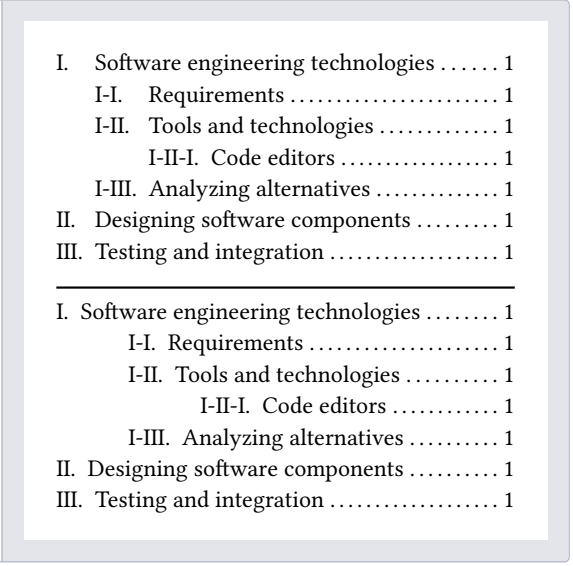
Cómo indentar las entradas del índice.

- `auto`: Indenta la numeración/prefijo de una entrada anidada con el título de su entrada padre. Si las entradas no están numeradas (por ejemplo, mediante la numeración de títulos[3.6.9/5.4]), en cambio simplemente inserta una sangría fija de `1.2em` por nivel.
- Longitud relativa[3.10.21]: Indenta la entrada con la longitud especificada por nivel de anidamiento. Especificar `2em`, por ejemplo, indentaría los títulos de nivel superior `0em` (no anidados), los títulos de segundo nivel `2em` (anidados una vez), los de tercer nivel `4em` (anidados dos veces), y así sucesivamente.
- Función[3.5.16]: Podés personalizar aún más esta configuración con una función. Esa función recibe el nivel de anidamiento como parámetro (empezando en 0 para los títulos/elementos de nivel superior) y debe devolver una longitud (relativa). Por ejemplo, `n => n * 2em` equivaldría a especificar simplemente `2em`.

```
#set heading(numbering: "I-I.")
#set outline(title: none)

#outline()
#line(length: 100%)
#outline(indent: 3em)

= Software engineering technologies
== Requirements
== Tools and technologies
=== Code editors
== Analyzing alternatives
= Designing software components
= Testing and integration
```




## /5 Funciones

### /5.1 `entry` Elemento Desde: 0.6.0

Representa una línea de entrada en un índice.

Con reglas show-set y show sobre las entradas del índice, podés personalizar ampliamente el aspecto del índice. Mirá la sección sobre cómo dar estilo al índice[3.6.13/3] para más detalles.

```
outline.entry(
  int,
```

```
  content,
  fill: none content,
) → content
```

/5.1.1 **level** int *Obligatorio* *Posicional*

El nivel de anidamiento de esta entrada del índice. Empieza en 1 para las entradas de nivel superior.

/5.1.2 **element** content *Obligatorio* *Posicional*

El elemento al que se refiere esta entrada. Su ubicación estará disponible mediante el método location[3.5.8/3.5] del contenido y se puede enlazar[3.6.10].

/5.1.3 **fill** none or content *Configurable con set* Por defecto: repeat(body: [.], gap: 0.15em)

Contenido que llena el espacio entre el título y el número de página. Se puede establecer en none para desactivar el relleno.

El fill se colocará en un box de tamaño fraccionario que abarca el espacio entre el cuerpo de la entrada y el número de página. Por lo tanto, al usar reglas show para reemplazar entradas del índice, se recomienda envolver el relleno en un box[3.10.5] con ancho fraccionario, es decir, box(width: 1fr, it.fill).

Al usar repeat[3.10.22], la propiedad gap[3.10.22/3.2] puede ser útil para ajustar el peso visual del relleno.

```
#set outline.entry(fill: line(length: 100%))
#outline()

= A New Beginning
```




/5.1.4 **Funciones on entry**

/5.1.4.1 **indented** Contextual Desde: 0.13.0

Una función auxiliar para producir un diseño de entrada con sangría: dispone un prefijo y el resto de la entrada teniendo en cuenta la sangría.

Si la indent[3.6.13/4.4] del índice padre es auto, el contenido interno de todas las entradas del nivel N se alinea con el prefijo de todas las entradas del nivel N + 1, dejando al menos un espacio gap entre el prefijo y las partes internas. Además, los contenidos inner de todas las entradas del mismo nivel se alinean.

Si la sangría del índice es un valor fijo o una función, los prefijos se indentan, pero los contenidos internos simplemente se desplazan del prefijo el gap especificado, en lugar de alinearse en todo el índice. Para una explicación visual, mirá outline.indent[3.6.13/4.4].

```
self.indented(
  none content,
  content,
  gap: length,
) → content
```

/5.1.4.1.1 **prefix** none or content *Obligatorio* *Posicional*

El prefix se alinea con el contenido inner de las entradas que tienen un nivel menos.

En la regla show por defecto, es simplemente it.prefix(), pero se puede personalizar libremente.

/5.1.4.1.2 **inner** `content` *Obligatorio* *Posicional*

El contenido interno de la entrada, con formato.

En la regla show por defecto, es simplemente `it.inner()`, pero se puede personalizar libremente.

/5.1.4.1.3 **gap** `length` Por defecto: `0.5em`

El espacio entre el prefijo y el contenido interno.

/5.1.4.2 **prefix** Contextual Desde: 0.13.0

Da formato a la numeración del elemento (si la hay).

Esto también agrega el suplemento del elemento en el caso de figuras o ecuaciones. Por ejemplo, produciría `1.1` para un título, pero `Figure 1` para una figura, como es habitual en los índices.

```
self.prefix() → none content
```

/5.1.4.3 **inner** Contextual Desde: 0.13.0

Crea el contenido interno por defecto de la entrada.

Esto incluye el cuerpo, el relleno y el número de página.

```
self.inner() → content
```

/5.1.4.4 **body** Desde: 0.13.0

El contenido que se muestra en lugar del elemento referido en su entrada del índice. Para un título, es su body[3.6.9/5.9]; para una figura, un epígrafe y, para las ecuaciones, está vacío.

```
self.body() → content
```

/5.1.4.5 **page** Contextual Desde: 0.13.0

El número de página del elemento de esta entrada, con el formato de la numeración establecida para la página referida.

```
self.page() → content
```

### 3.6.14 `par` Elemento Desde: siempre

Una subdivisión lógica de contenido textual.

Typst reúne automáticamente los elementos *en línea* en párrafos. Los elementos en línea incluyen el texto[3.7.12], el espaciado horizontal[3.10.26], los boxes[3.10.5] y las ecuaciones en línea[3.8.16].

Para separar párrafos, usá una línea en blanco (o un `parbreak`[3.6.15] explícito). Los párrafos también se interrumpen automáticamente con cualquier elemento de bloque (como `block`[3.10.4], `place`[3.10.19] o cualquier cosa que se muestre como uno de estos).

El elemento `par` se usa principalmente en reglas set para afectar las propiedades de los párrafos, pero también se puede usar para mostrar explícitamente su argumento como un párrafo propio. En ese caso, el cuerpo del párrafo no puede contener contenido de bloque.

#### /1 Boxes y bloques

Como se explicó arriba, normalmente los párrafos solo contienen contenido en línea. Sin embargo, podés integrar cualquier tipo de contenido de bloque en un párrafo envolviéndolo en un `box`[3.10.5].

Inversamente, podés separar contenido en línea de un párrafo envolviéndolo en un `block`[3.10.4]. En ese caso, no pasará a formar parte de ningún párrafo. Leé la siguiente sección para ver una explicación de por qué esto importa y en qué se diferencia de simplemente agregar saltos de párrafo alrededor del contenido.

#### /2 ¿Qué se convierte en un párrafo?

Cuando agregás contenido en línea a tu documento, Typst lo envolverá automáticamente en párrafos. Sin embargo, un documento típico también contiene algo de texto que, semánticamente, no es parte de un párrafo, por ejemplo en un título o un epígrafe.

Las reglas sobre cuándo Typst envuelve contenido en línea en un párrafo son las siguientes:

- Todo el texto en la raíz de un documento se envuelve en párrafos.
- El texto dentro de un contenedor (como un `block`) solo se envuelve en un párrafo si el contenedor tiene contenido de bloque. Si todo el contenido es en línea, no se crea ningún párrafo.

En el documento ya dispuesto, no se ve de inmediato si el texto pasó a formar parte de un párrafo. Sin embargo, sigue siendo importante por varias razones:

- Cierto estilo de párrafo, como `first-line-indent`, solo se aplicará a párrafos propiamente dichos, no a cualquier texto. De manera similar, las reglas show de `par` por supuesto solo se activan con párrafos.
- Una distinción adecuada entre párrafos y otro texto ayuda a las personas que dependen de tecnologías de asistencia (TA) (como los lectores de pantalla) a navegar y entender el documento correctamente.
- La exportación a PDF generará una etiqueta `P` solo para los párrafos.
- La exportación a HTML generará una etiqueta `<p>` solo para los párrafos.

Al crear componentes reutilizables personalizados, podés y deberías hacerte cargo de si Typst crea o no párrafos. Al envolver texto en un `block`[3.10.4] en lugar de simplemente agregar saltos de párrafo alrededor, podés forzar la ausencia de un párrafo. Inversamente, al agregar un `parbreak`[3.6.15] después de algún contenido en un contenedor, podés forzar que se convierta en un párrafo aunque sea una sola palabra. Esto es, por ejemplo, lo que hacen las listas no `tight`[3.6.2/3.1] para forzar que sus elementos se conviertan en párrafos.

## /3 Ejemplo

```
#set par(
  first-line-indent: 1em,
  spacing: 0.65em,
  justify: true,
)

We proceed by contradiction.
Suppose that there exists a set
of positive integers $a$, $b$, and
$c$ that satisfies the equation
$a^n + b^n = c^n$ for some
integer value of $n > 2$.

Without loss of generality,
let $a$ be the smallest of the
three integers. Then, we ...
```

We proceed by contradiction. Suppose that there exists a set of positive integers $a$, $b$, and $c$ that satisfies the equation $a^n + b^n = c^n$ for some integer value of $n > 2$.

Without loss of generality, let $a$ be the smallest of the three integers. Then, we ...

## /4 Parámetros

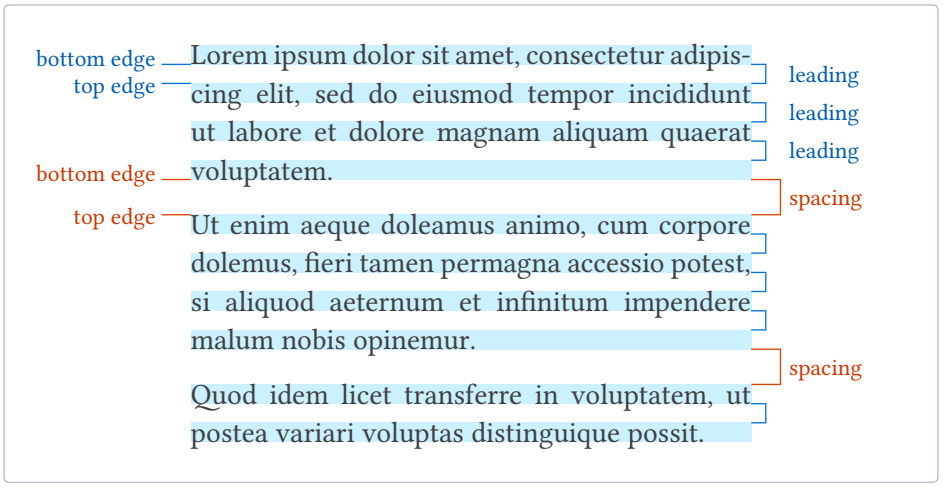
```
par(
  leading: length,
  spacing: length,
  justify: bool,
  justification-limits: dictionary,
  linebreaks: auto str,
  first-line-indent: length dictionary,
  hanging-indent: length,
  content,
) → content
```

### /4.1 **leading** length *Configurable con set* Por defecto: 0.65em

El espaciado entre líneas.

El interlineado (leading) define el espacio entre el borde inferior[3.7.12/2.15] de una línea y el borde superior[3.7.12/2.14] de la línea siguiente. Por defecto, estas dos propiedades dependen de la fuente, pero también se pueden configurar manualmente con una regla set de text.

Al establecer el borde superior, el borde inferior y el interlineado, también podés configurar una distancia constante de línea base a línea base. Podrías, por ejemplo, establecer el interlineado en 1em, el borde superior en 0.8em y el borde inferior en -0.2em para obtener un espacio entre líneas base de exactamente 2em. La distribución exacta de los valores de los bordes superior e inferior afecta los límites de la primera y la última línea.




### /4.2 **spacing** `length` *Configurable con set* Por defecto: `1.2em`

El espaciado entre párrafos.

Igual que el interlineado, define el espacio entre el borde inferior de la última línea de un párrafo y el borde superior de la primera línea del párrafo siguiente. El espaciado actúa tanto por encima como por debajo y colapsa al mayor de los valores definidos por los párrafos adyacentes.

Cuando un párrafo es adyacente a un `block`[3.10.4], la propiedad `above`[3.10.4/2.10] o `below`[3.10.4/2.11] de ese bloque tiene prioridad sobre el espaciado del párrafo. Los títulos, por ejemplo, reducen por defecto el espaciado que tienen debajo para lograr un mejor aspecto.

### /4.3 **justify** `bool` *Configurable con set* Por defecto: `false`

Si se justifica el texto en su línea.

La separación en sílabas se habilitará en los párrafos justificados si la propiedad `hyphenate` de la función `text`[3.7.12/2.20] está establecida en `auto` y se conoce el idioma actual.

Tené en cuenta que la alineación[3.10.1/5.1] actual igualmente tiene un efecto en la ubicación de la última línea, salvo que termine con un salto de línea justificado[3.7.2/3.1].

Por defecto, Typst solo cambia el espaciado entre palabras para lograr la justificación. Sin embargo, también podés permitirle ajustar el espaciado entre caracteres individuales con la propiedad `justification-limits`[3.6.14/4.4].

### /4.4 **justification-limits** `dictionary` *Configurable con set*

Por defecto: (
```
  spacing: (min: 66.67% + 0pt, max: 150% + 0pt),
  tracking: (min: 0pt, max: 0pt),
)
```

Cuánto se puede ajustar el espaciado entre palabras y caracteres durante la justificación.

Al justificar texto, Typst necesita estirar o comprimir una línea hasta el ancho completo de la medida. Para lograrlo, por defecto ajusta el espaciado entre palabras. Además, también puede ajustar el espaciado entre caracteres individuales. Esta propiedad te permite configurar los límites inferior y superior de estos ajustes.

La propiedad acepta un diccionario con dos entradas, `spacing` y `tracking`, cada una con un diccionario con las claves `min` y `max`. Las claves `min` definen hasta qué límite inferior se pueden comprimir los espacios, mientras que las claves `max` definen hasta qué límite superior se pueden estirar.

- La entrada `spacing` define cuánto se puede ajustar el ancho de los espacios entre palabras. Está estrechamente relacionada con `text.spacing`[3.7.12/2.10] y sus claves `min` y `max` aceptan longitudes relativas[3.10.21], igual que la propiedad `spacing`.

  Un valor `min` de `100%` significa que los espacios deben conservar su tamaño normal (es decir, no comprimirse), mientras que un valor de `90% - 0.01em` indicaría que un espacio se puede comprimir hasta un ancho del 90 % de su ancho normal menos 0,01 veces el tamaño de fuente actual. De manera similar, un valor `max` de `100% + 0.02em` significa que el ancho de un espacio se puede aumentar en 0,02 veces

el tamaño de fuente actual. La parte de proporción siempre debe ser positiva. La parte de longitud, en cambio, no debe ser positiva para `min` ni negativa para `max`.

Tené en cuenta que los espacios igualmente pueden expandirse más allá del valor `max` si no hay otra manera de justificar la línea. Sin embargo, los demás medios de justificación (por ejemplo, separar los caracteres si la entrada `tracking` está configurada en consecuencia) se usan primero hasta su máximo.

– La entrada `tracking` define cuánto se puede ajustar el espaciado entre letras. Está estrechamente relacionada con `text.tracking`[3.7.12/2.9] y sus claves `min` y `max` aceptan longitudes[3.10.13], igual que la propiedad `tracking`. A diferencia de `spacing`, no acepta longitudes relativas, porque la base de la longitud relativa variaría para cada carácter, lo que llevaría a un aspecto visual desparejo. El comportamiento en comparación con `spacing` es como si la base fuera `100%`.

  Por lo demás, los valores `min` y `max` funcionan igual que para `spacing`. Un valor `max` de `0.01em` significa que se puede insertar un espaciado adicional de 0,01 veces el tamaño de fuente actual entre cada par de caracteres. Tené en cuenta que esto también incluye los espacios entre espacios y caracteres, por lo que, para los espacios, los valores de `tracking` actúan además de los valores de `spacing`.

Si solo especificás uno de `spacing` o `tracking`, el otro conserva su valor establecido previamente (o el valor por defecto si no se había establecido).

Si querés habilitar la justificación por carácter, un buen valor para las claves `min` y `max` ronda entre `0.01em` y `0.02em` (negado para `min`). Usar el mismo valor para ambas da una buena base, pero ajustar los dos valores individualmente puede producir resultados más equilibrados, como demuestra el ejemplo de abajo. Tené cuidado de no establecer los límites demasiado amplios, ya que rápidamente se ve poco natural.

La justificación por carácter es una técnica microtipográfica de gran impacto que puede mejorar el aspecto del texto justificado, especialmente en columnas angostas. Tené en cuenta, sin embargo, que la justificación por carácter no funciona con todas las fuentes ni idiomas. Por ejemplo, las fuentes cursivas conectan las letras. Usar la justificación por carácter llevaría a conexiones irregulares.

Example · Justificación por carácter

```
#let example(name) = columns(2, gutter: 10pt)[
  #place(top, float: true, scope: "parent", strong(name))
  /* Text from https://en.wikipedia.org/wiki/Anne_Bayley */
]

#set page(width: 440pt, height: 21em, margin: 15pt)
#set par(justify: true)
#set text(size: 0.8em)

#grid(
  columns: (1fr, 1fr),
  gutter: 20pt,
  {
    // These are Typst's default limits.
    set par(justification-limits: (
      spacing: (min: 100% * 2 / 3, max: 150%),
      tracking: (min: 0em, max: 0em),
    ))
    example[Word-level justification]
  },
  {
    // These are our custom character-level limits.
    set par(justification-limits: (
      tracking: (min: -0.01em, max: 0.02em),
    ))
    example[Character-level justification]
  },
)
```

**Word-level justification**

Anne Christine Bayley (1 June 1934 – 31 December 2024) was an English surgeon. She was awarded the Order of the British Empire for her research into HIV/AIDS patients in Zambia and for documenting the spread of the disease among heterosexual patients in Africa. In addition to her clinical work, she was a lecturer and head of the surgery department at the University of Zambia School of Medicine. In the 1990s, she returned to England, where she was ordained as an Anglican priest. She continued to be active in Africa throughout her retirement years.

**Character-level justification**

Anne Christine Bayley (1 June 1934 – 31 December 2024) was an English surgeon. She was awarded the Order of the British Empire for her research into HIV/AIDS patients in Zambia and for documenting the spread of the disease among heterosexual patients in Africa. In addition to her clinical work, she was a lecturer and head of the surgery department at the University of Zambia School of Medicine. In the 1990s, she returned to England, where she was ordained as an Anglican priest. She continued to be active in Africa throughout her retirement years.

### /4.5 `linebreaks` `auto` or `str` *Configurable con set*

Por defecto: `auto`

Cómo determinar los saltos de línea.

Cuando esta propiedad se establece en `auto`, su valor por defecto, se usarán saltos de línea optimizados para los párrafos justificados. Habilitar los saltos de línea optimizados para párrafos sin justificar también puede valer la pena para mejorar el aspecto del texto.

```
#set page(width: 207pt)
#set par(linebreaks: "simple")
Some texts feature many longer
words. Those are often exceedingly
challenging to break in a visually
pleasing way.

#set par(linebreaks: "optimized")
Some texts feature many longer
words. Those are often exceedingly
challenging to break in a visually
pleasing way.
```

Some texts feature many longer words. Those are often exceedingly challenging to break in a visually pleasing way.

Some texts feature many longer words. Those are often exceedingly challenging to break in a visually pleasing way.

| Variante | Detalles |
|---|---|
| `"simple"` | Determina los saltos de línea con un estilo simple de primer ajuste (first-fit). |
| `"optimized"` | Optimiza los saltos de línea para todo el párrafo.<br>Typst intenta producir líneas de texto más parejas considerando todo el párrafo al calcular los saltos de línea. |

### /4.6 `first-line-indent` `length` or `dictionary` *Configurable con set*

Por defecto: `(amount: 0pt, all: false)`

La sangría que debe tener la primera línea de un párrafo.

Por defecto, solo se indentará la primera línea de un párrafo consecutivo (no el primero del documento o del contenedor, ni los párrafos que siguen inmediatamente a otros elementos de bloque).

Si en cambio querés indentar todos los párrafos, podés pasar un diccionario que contenga la cantidad (`amount`) de sangría como longitud y el par `all: true`. Cuando se omite `all` del diccionario, su valor por defecto es `false`.

Por convención tipográfica, los saltos de párrafo se indican con algo de espacio entre párrafos o con primeras líneas con sangría. Considerá
– reducir el spacing del párrafo[3.6.14/4.2] al `leading`[3.6.14/4.1] usando `set par(spacing: 0.65em)`

– aumentar el spacing del bloque[3.10.4/2.9] (que hereda por defecto el espaciado del párrafo) al espaciado de párrafo original usando `set block(spacing: 1.2em)`

```
#set block(spacing: 1.2em)
#set par(
  first-line-indent: 1.5em,
  spacing: 0.65em,
)

The first paragraph is not affected
by the indent.

But the second paragraph is.

#line(length: 100%)

#set par(first-line-indent: (
  amount: 1.5em,
  all: true,
))

Now all paragraphs are affected
by the first line indent.

Even the first one.
```

The first paragraph is not affected by the indent.

But the second paragraph is.

---

Now all paragraphs are affected by the first line indent.

Even the first one.

### /4.7 hanging-indent `length` *Configurable con set* Por defecto: `0pt`

La sangría que deben tener todas las líneas de un párrafo excepto la primera.

```
#set par(hanging-indent: 1em)

#lorem(15)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

### /4.8 body `content` *Obligatorio* *Posicional*

El contenido del párrafo.

## /5 Funciones

### /5.1 line Elemento Desde: 0.12.0

Una línea de párrafo.

Este elemento se usa exclusivamente para configurar los números de línea mediante reglas set y no se puede ubicar.

La opción numbering[3.6.14/5.1.1] se usa para habilitar los números de línea especificando un formato de numeración.

```
#set par.line(numbering: "1")

Roses are red. \
Violets are blue. \
Typst is there for you.
```

1 Roses are red.
2 Violets are blue.
3 Typst is there for you.

La opción `numbering` toma un patrón de numeración[3.6.12] predefinido o una función que devuelva contenido con estilo. Podés desactivar los números de línea del texto dentro de ciertos elementos estableciendo la numeración en `none` con reglas show-set.

```
// Styled red line numbers.
#set par.line(
  numbering: n => text(red)[#n]
)

// Disable numbers inside figures.
#show figure: set par.line(
  numbering: none
)

Roses are red. \
Violets are blue.

#figure(
  caption: [Without line numbers.]
)[
  Lorem ipsum \
  dolor sit amet
]

The text above is a sample \
originating from distant times.
```

1 Roses are red.
2 Violets are blue.

Lorem ipsum
dolor sit amet
Figura 1: Without line numbers.

3 The text above is a sample
4 originating from distant times.

Este elemento expone más opciones que se pueden usar para controlar otros aspectos de la numeración de líneas, como su alineación[3.6.14/5.1.2] o su margen[3.6.14/5.1.3]. Además, podés controlar si la numeración se reinicia en cada página mediante la opción `numbering-scope`[3.6.14/5.1.5].

```
par.line(
  numbering: none str function,
  number-align: auto alignment,
  number-margin: alignment,
  number-clearance: auto length,
  numbering-scope: str,
) → content
```

/5.1.1 **numbering** `none` or `str` or `function` *Configurable con set* Por defecto: none

Cómo numerar cada línea. Acepta un patrón o función de numeración[3.6.12] que toma un único número.

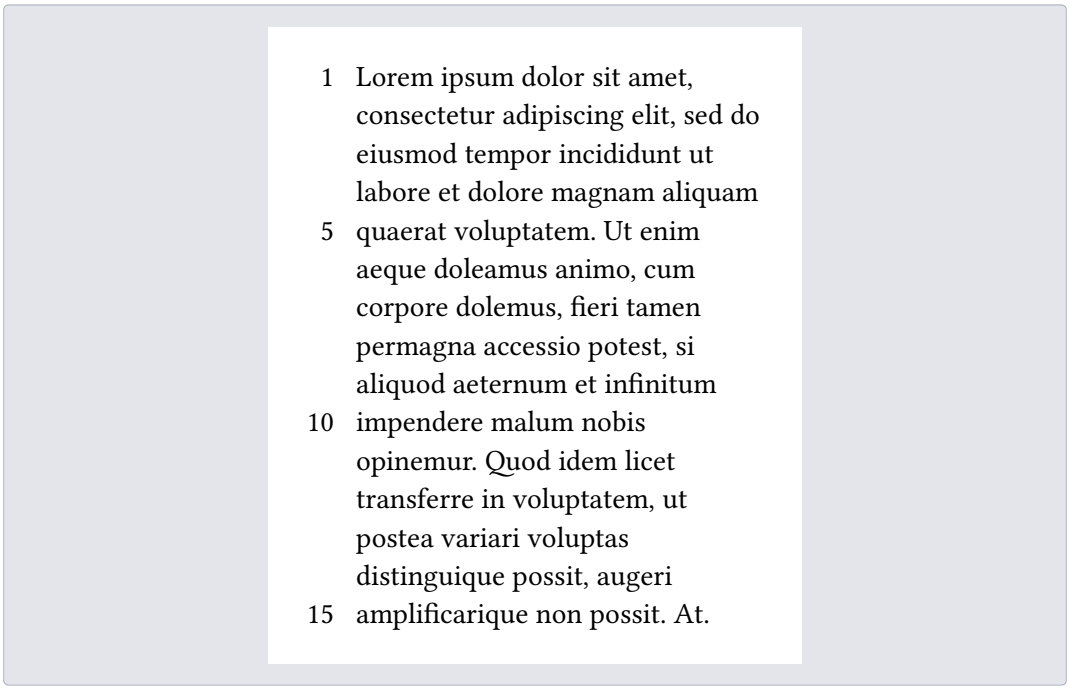
```
#set par.line(numbering: "I")

Roses are red. \
Violets are blue. \
Typst is there for you.
```

I Roses are red.
II Violets are blue.
III Typst is there for you.

```
#set par.line(
  numbering: i => if calc.rem(i, 5) == 0 or i == 1 { i },
)

#lorem(60)
```




/5.1.2 **number-align** `auto` or `alignment` *Configurable con set* Por defecto: `auto`

La alineación de los números de línea asociados a cada línea.

El valor por defecto `auto` indica un valor inteligente en el que los números crecen horizontalmente alejándose del texto, teniendo en cuenta el margen en el que están y la dirección de texto actual.

```
#set par.line(
  numbering: "I",
  number-align: left,
)

Hello world! \
Today is a beautiful day \
For exploring the world.
```

I Hello world!
II Today is a beautiful day
III For exploring the world.

/5.1.3 **number-margin** `alignment` *Configurable con set* Por defecto: `start`

El margen en el que aparecen los números de línea.

*Nota:* En un documento de varias columnas, los números de línea de los párrafos dentro de la última columna siempre aparecerán en el margen end (margen derecho para texto de izquierda a derecha y margen izquierdo para texto de derecha a izquierda), sin importar esta configuración. Ese comportamiento no se puede cambiar por el momento.

```
#set par.line(
  numbering: "1",
  number-margin: right,
)

= Report
- Brightness: Dark, yet darker
- Readings: Negative
```

**Report** 1
• Brightness: Dark, yet darker 2
• Readings: Negative 3

/5.1.4 **number-clearance** `auto` or `length` *Configurable con set* Por defecto: `auto`

La distancia entre los números de línea y el texto.

El valor por defecto `auto` produce un espacio que se adapta al ancho de la página y da resultados razonables en la mayoría de los casos.

```
#set par.line(
  numbering: "1",
  number-clearance: 4pt,
)

Typesetting \
Styling \
Layout
```

1 Typesetting
2 Styling
3 Layout

### /5.1.5 numbering-scope `str` *Configurable con set*

Por defecto: `"document"`

Controla cuándo se reinicia la numeración de líneas.

*Nota:* El ámbito de la numeración de líneas debe ser uniforme en cada serie de páginas (una serie de páginas es una secuencia de páginas sin un salto de página explícito entre ellas). Por este motivo, las reglas set para él deberían definirse antes de cualquier contenido de página, normalmente al comienzo del documento.

```
#set par.line(
  numbering: "1",
  numbering-scope: "page",
)

First line \
Second line
#pagebreak()
First line again \
Second line again
```

1 First line
2 Second line

1 First line again
2 Second line again

| Variante | Detalles |
|---|---|
| `"document"` | Indica que el contador de números de línea abarca todo el documento, es decir, nunca se reinicia automáticamente. |
| `"page"` | Indica que el contador de números de línea se reinicia al comienzo de cada página nueva. |

### 3.6.15 `parbreak` Elemento Desde: siempre

Un salto de párrafo.

Esto inicia un párrafo nuevo. Es especialmente útil cuando se usa dentro de código, como los bucles for[3.3.5]. Varios saltos de párrafo consecutivos colapsan en uno solo.

#### /1 Ejemplo

```
#for i in range(3) {
  [Blind text #i: ]
  lorem(5)
  parbreak()
}
```

Blind text 0: Lorem ipsum dolor sit amet.

Blind text 1: Lorem ipsum dolor sit amet.

Blind text 2: Lorem ipsum dolor sit amet.

#### /2 Sintaxis

En lugar de llamar a esta función, podés insertar una línea en blanco en tu marcado para crear un salto de párrafo.

#### /3 Parámetros

```
parbreak() → content
```

### 3.6.16 `quote` Elemento Desde: 0.9.0

Muestra una cita junto con una atribución opcional.

#### /1 Ejemplo

```
Plato is often misquoted as the author of #quote[I know that I know
nothing], however, this is a derivation form his original quote:

#set quote(block: true)

#quote(attribution: [Plato])[
  ... ἔοικα γοῦν τούτου γε σμικρῷ τινι αὐτῷ τούτῳ σοφώτερος εἶναι, ὅτι
  ἃ μὴ οἶδα οὐδὲ οἴομαι εἰδέναι.
]
#quote(attribution: [from the Henry Cary literal translation of 1897])[
  ... I seem, then, in just this little thing to be wiser than this man at
  any rate, that what I do not know I do not think I know either.
]
```

Plato is often misquoted as the author of “I know that I know nothing”, however, this is a derivation form his original quote:

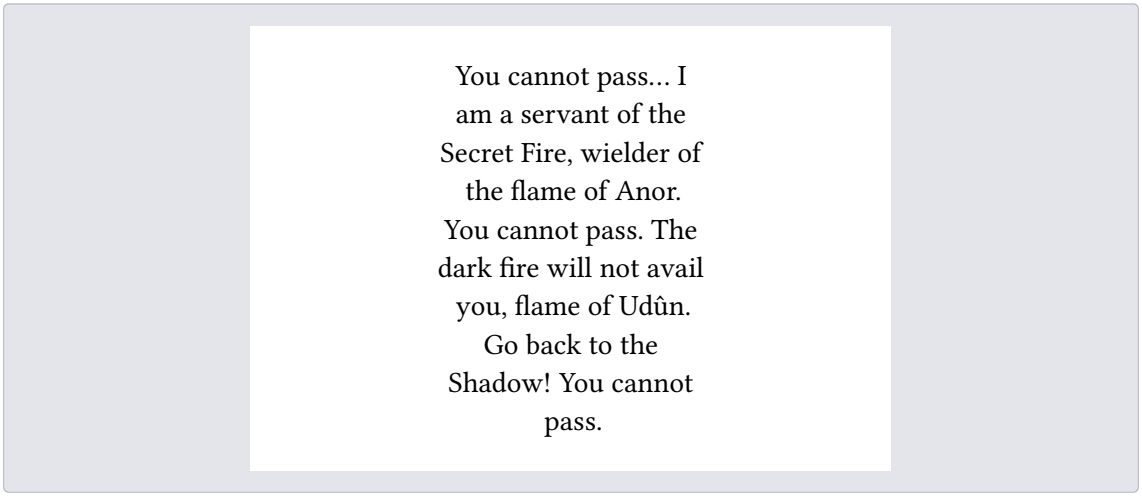
> ... ἔοικα γοῦν τούτου γε σμικρῷ τινι αὐτῷ τούτῳ σοφώτερος εἶναι, ὅτι ἃ μὴ οἶδα οὐδὲ οἴομαι εἰδέναι.
>
> — Plato

> ... I seem, then, in just this little thing to be wiser than this man at any rate, that what I do not know I do not think I know either.
>
> — from the Henry Cary literal translation of 1897

Por defecto, las citas en bloque tienen un relleno de `1em` a izquierda y derecha; la alineación y el relleno se pueden controlar con reglas show:

```
#set quote(block: true)
#show quote: set align(center)
#show quote: set pad(x: 5em)

#quote[
  You cannot pass... I am a servant of the Secret Fire, wielder of the
  flame of Anor. You cannot pass. The dark fire will not avail you,
  flame of Udûn. Go back to the Shadow! You cannot pass.
]
```




## /2 Parámetros

```
quote(
  block: bool,
  quotes: auto bool,
  attribution: none label content,
  content,
) → content
```

### /2.1 block `bool` *Configurable con set* Por defecto: `false`

Si esta es una cita en bloque.

```
An inline citation would look like
this: #quote(
  attribution: [René Descartes]
)[
  cogito, ergo sum
], and a block equation like this:
#quote(
  block: true,
  attribution: [JFK]
)[
  Ich bin ein Berliner.
]
```

An inline citation would look like this: “cogito, ergo sum”, and a block equation like this:

> Ich bin ein Berliner.
>
> — JFK

### /2.2 quotes `auto` or `bool` *Configurable con set* Por defecto: `auto`

Si se deben agregar comillas dobles alrededor de esta cita.

Las comillas dobles que se usan se infieren de la propiedad `quotes` de smartquote[3.7.8], que se ve afectada por la propiedad `lang` de text[3.7.12].

- `true`: Envuelve esta cita entre comillas dobles.
- `false`: No envuelve esta cita entre comillas dobles.
- `auto`: Infiere si se debe envolver esta cita entre comillas dobles según la propiedad `block`. Si `block` es `false`, se agregan comillas dobles automáticamente.

```
#set text(lang: "de")

Ein deutsch-sprechender Author
zitiert unter umständen JFK:
#quote[Ich bin ein Berliner.]

#set text(lang: "en")

And an english speaking one may
translate the quote:
#quote[I am a Berliner.]
```

Ein deutsch-sprechender Author zitiert unter umständen JFK: „Ich bin ein Berliner.“

And an english speaking one may translate the quote: “I am a Berliner.”

/2.3 **attribution** `none` or `label` or `content` *Configurable con set* Por defecto: none

La atribución de esta cita, generalmente el autor o la fuente. Puede ser una etiqueta que apunte a una entrada de bibliografía o cualquier contenido. Por defecto solo se muestra para las citas en bloque, pero se puede cambiar con una regla show.

```
#quote(attribution: [René Descartes])[
  cogito, ergo sum
]

#show quote.where(block: false): it => {
  ["] + h(0pt, weak: true) + it.body + h(0pt, weak: true) + ["]
  if it.attribution != none [ (#it.attribution)]
}

#quote(
  attribution: link("https://typst.app/home")[typst.app]
)[
  Compose papers faster
]

#set quote(block: true)

#quote(attribution: <tolkien54>)[
  You cannot pass... I am a servant
  of the Secret Fire, wielder of the
  flame of Anor. You cannot pass. The
  dark fire will not avail you, flame
  of Udûn. Go back to the Shadow! You
  cannot pass.
]

#bibliography("works.bib", style: "apa")
```

“cogito, ergo sum”

“Compose papers faster” (typst.app)

> You cannot pass… I am a servant of the Secret Fire, wielder of the flame of Anor. You cannot pass. The dark fire will not avail you, flame of Udûn. Go back to the Shadow! You cannot pass.
>
> — Tolkien (1954)

## Bibliografía

Tolkien, J. R. R. (1954). *The Fellowship of the Ring* (Vol. 1, Pt. 1, Chapter 1). Allen & Unwin.

/2.4 **body** `content` *Obligatorio* *Posicional*

La cita.

### 3.6.17 `ref` Elemento Desde: siempre

Una referencia a una etiqueta o a la bibliografía.

Toma una etiqueta y hace una referencia cruzada a ella. Hay dos tipos de referencias, determinados por su `form`[3.6.17/4.3]: `"normal"` y `"page"`.

La referencia por defecto, `"normal"`, produce una referencia textual a una etiqueta. Por ejemplo, una referencia a un título dará una cadena apropiada como "Sección 1" para una referencia al primer título. La palabra "Sección" depende de la configuración de `lang`[3.7.12/2.16] y se localiza en consecuencia. Las referencias también son enlaces al elemento respectivo. La sintaxis de referencias también se puede usar para citar[3.6.3] de una bibliografía.

Como la forma por defecto requiere un suplemento y una numeración, la etiqueta debe estar adjunta a un *elemento referenciable*. Los elementos referenciables incluyen los títulos[3.6.9], las figuras[3.6.7], las ecuaciones[3.8.16] y las notas al pie[3.6.8]. Para crear un elemento referenciable personalizado, como un teorema, podés crear una figura de un `kind`[3.6.7/5.6] personalizado y escribir una regla show para ella. En el futuro, podría haber una forma más directa de definir un elemento referenciable personalizado.

Si solo querés enlazar a un elemento etiquetado y no obtener una referencia textual automática, considerá usar en cambio la función `link`[3.6.10].

Una referencia `"page"` produce una referencia de página a una etiqueta, mostrando el número de página en su ubicación. Podés usar el suplemento de la página[3.10.17/4.11] para modificar el texto que va antes del número de página. A diferencia de una referencia `"normal"`, la etiqueta se puede adjuntar a cualquier elemento siempre que esté establecida la numeración de la página[3.10.17/4.10] correspondiente.

#### /1 Ejemplo

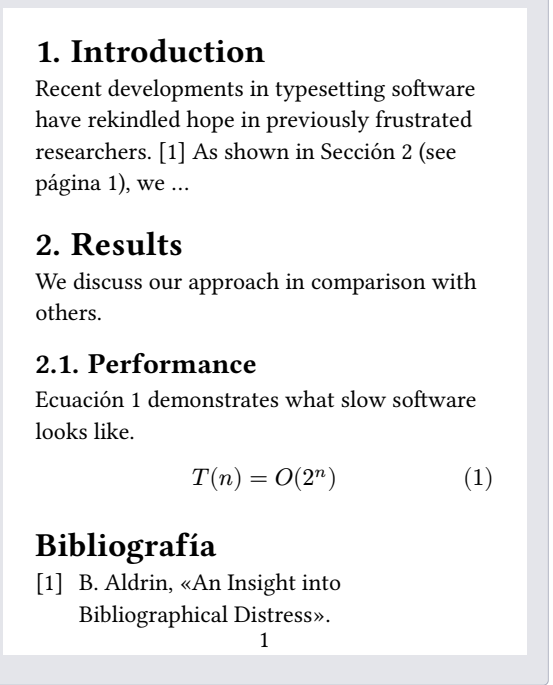
```
#set page(numbering: "1")
#set heading(numbering: "1.")
#set math.equation(numbering: "(1)")

= Introduction <intro>
Recent developments in
typesetting software have
rekindled hope in previously
frustrated researchers. @distress
As shown in @results (see
#ref(<results>, form: "page")),
we ...

= Results <results>
We discuss our approach in
comparison with others.

== Performance <perf>
@slow demonstrates what slow
software looks like.
$ T(n) = O(2^n) $ <slow>

#bibliography("works.bib")
```




#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: una referencia `"normal"` a una etiqueta se puede crear escribiendo una `@` seguida del nombre de la etiqueta (por ejemplo, `= Introduction <intro>` se puede referenciar escribiendo `@intro`).

Para personalizar el suplemento, agregá contenido entre corchetes después de la referencia: `@intro[Chapter]`.

## /3 Personalización

Cuando en un documento solo necesitás referenciar páginas de una figura/tabla/título/etcétera, el valor por defecto del campo `form` se puede cambiar a `"page"` con una regla set. Si preferís un suplemento corto "p." en lugar de "página", se puede usar el campo `page.supplement`[3.10.17/4.11] para cambiarlo:

```
#set page(
  numbering: "1",
  supplement: "p.",
)
#set ref(form: "page")

#figure(
  stack(
    dir: ltr,
    spacing: 1em,
    circle(),
    square(),
  ),
  caption: [Shapes],
) <shapes>

#pagebreak()

See @shapes for examples
of different shapes.
```



Figura 1: Shapes

1

See p. 1 for examples of different shapes.

2

Si escribís una regla show para referencias, podés acceder al elemento referenciado a través del campo `element` de la referencia. El `element` puede ser `none` aunque exista, si Typst todavía no lo descubrió, por lo que siempre necesitás manejar ese caso en tu código.

```
#set heading(numbering: "1.")
#set math.equation(numbering: "(1)")

#show ref: it => {
  let eq = math.equation
  let el = it.element
  // Skip all other references.
  if el == none or el.func() != eq { return it }
  // Override equation references.
  link(el.location(), counter(eq).display(at: el.location()))
}

= Beginnings <beginning>
In @beginning we prove @pythagoras.
$ a^2 + b^2 = c^2 $ <pythagoras>
```

# 1. Beginnings

In Sección 1 we prove (1).

$$a^2 + b^2 = c^2 \qquad (1)$$

## /4 Parámetros

```
ref(
  label,
```

```
  supplement: none auto content function,
  form: str,
) → content
```

### /4.1 target label *Obligatorio* *Posicional*

La etiqueta de destino que se debe referenciar.

Puede ser una etiqueta definida en el documento o, si la form[3.6.17/4.3] se establece en "normal", una entrada de la bibliography[3.6.1].

### /4.2 supplement none or auto or content or function *Configurable con set* Por defecto: auto

Un suplemento para la referencia.

Si la form[3.6.17/4.3] se establece en "normal":
- En las referencias a títulos o figuras, se agrega antes del número referenciado.
- En las citas, se puede usar para agregar un número de página.

Si la form[3.6.17/4.3] se establece en "page", se agrega antes del número de página de la etiqueta referenciada.

Si se especifica una función, se le pasa el elemento referenciado y debe devolver contenido.

```
#set heading(numbering: "1.")
#show ref.where(
  form: "normal"
): set ref(supplement: it => {
  if it.func() == heading {
    "Chapter"
  } else {
    "Thing"
  }
})

= Introduction <intro>
In @intro, we see how to turn
Sections into Chapters. And
in @intro[Part], it is done
manually.
```

**1. Introduction**
In Chapter 1, we see how to turn Sections into Chapters. And in Part 1, it is done manually.

### /4.3 form str *Configurable con set* Por defecto: "normal"

El tipo de referencia que se produce.

```
#set page(numbering: "1")

Here <here> we are on
#ref(<here>, form: "page").
```

Here we are on página 1.
1

| Variante | Detalles |
|---|---|
| "normal" | Produce una referencia textual a una etiqueta. |
| "page" | Produce una referencia de página a una etiqueta. |

### 3.6.18 `strong` Elemento Desde: siempre

Enfatiza fuertemente el contenido aumentando el peso de la fuente.

Aumenta el peso de fuente actual en un `delta` dado.

#### /1 Ejemplo

```
This is *strong.* \
This is #strong[too.] \

#show strong: set text(red)
And this is *evermore.*
```

This is **strong.**
This is **too.**

And this is **evermore.**

#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: para enfatizar fuertemente contenido, simplemente encerralo entre estrellas/asteriscos (*). Tené en cuenta que esto solo funciona en los límites de las palabras. Para enfatizar fuertemente una parte de una palabra, tenés que usar la función.

#### /3 Parámetros

```
strong(
  delta: int,
  content,
) → content
```

##### /3.1 **delta** `int` *Configurable con set* Por defecto: `300`

El delta que se aplica al peso de la fuente.

```
#set strong(delta: 0)
No *effect!*
```

No effect!

##### /3.2 **body** `content` *Obligatorio* *Posicional*

El contenido que se enfatiza fuertemente.

### 3.6.19 `table` Elemento Desde: siempre

Una tabla de elementos.

Las tablas se usan para disponer contenido en celdas. Las celdas pueden contener contenido arbitrario, incluidos varios párrafos, y se especifican por filas. Para una explicación práctica de todas las formas en que podés usar y personalizar tablas en Typst, consultá la guía de tablas[4.3].

Como las tablas son simplemente grillas con distintos valores por defecto para algunas propiedades de las celdas (notablemente `stroke` e `inset`), consultá la documentación de grid[3.10.10/1] para más información sobre cómo dar tamaño a las pistas de la tabla y especificar las propiedades de aspecto de las celdas.

Si no estás seguro de si deberías usar una tabla o una grilla, considerá si el contenido que estás disponiendo pertenece semánticamente junto como un conjunto de datos relacionados o similares, o si simplemente querés mejorar tu presentación disponiendo contenido no relacionado en una grilla. En el primer caso, una tabla es la elección correcta; en el segundo, una grilla es más apropiada. Además, las tecnologías de asistencia (TA), como los lectores de pantalla, anunciarán el contenido de una `table` como tabular, mientras que el contenido de una grilla se anunciará igual que varios bloques de contenido en el flujo del documento. Los usuarios de TA podrán navegar las tablas bidimensionalmente por celda.

Tené en cuenta que, para reemplazar las propiedades de una celda en particular o aplicar reglas show sobre las celdas de una tabla, podés usar el elemento `table.cell`[3.6.19/4.1]. Consultá su documentación para más información.

Aunque `table` y `grid` comparten la mayoría de las propiedades, las reglas set y show sobre una de ellas no afectan a la otra. Se recomienda ubicar la mayor parte de tus estilos en reglas set y show, ya que mantiene limpios y fáciles de leer los usos reales de la tabla. También te permite cambiar fácilmente el aspecto de todas las tablas en un solo lugar.

Para darle un epígrafe a una tabla y hacerla referenciable[3.6.17], ponela en una figura[3.6.7].

#### /1 Ejemplo

El siguiente ejemplo demuestra algunas de las opciones de tabla más comunes.

```
#table(
  columns: (1fr, auto, auto),
  inset: 10pt,
  align: horizon,
  table.header(
    [], [*Volume*], [*Parameters*],
  ),
  image("cylinder.svg"),
  $ pi h (D^2 - d^2) / 4 $,
  [
    $h$: height \
    $D$: outer radius \
    $d$: inner radius
  ],
  image("tetrahedron.svg"),
  $ sqrt(2) / 12 a^3 $,
  [$a$: edge length]
)
```

| | **Volume** | **Parameters** |
|---|---|---|
|  | $\pi h \frac{D^2 - d^2}{4}$ | $h$: height<br>$D$: outer radius<br>$d$: inner radius |
|  | $\frac{\sqrt{2}}{12} a^3$ | $a$: edge length |

Igual que con las grillas, podés usar `table.cell`[3.6.19/4.1] para personalizar el aspecto y la posición de cada celda.

```
#set table(
  stroke: none,
  gutter: 0.2em,
  fill: (x, y) =>
    if x == 0 or y == 0 { gray },
  inset: (right: 1.5em),
)

#show table.cell: it => {
  if it.x == 0 or it.y == 0 {
    set text(white)
    strong(it)
  } else if it.body == [] {
    // Replace empty cells with 'N/A'
    pad(..it.inset)[_N/A_]
  } else {
    it
  }
}

#let a = table.cell(
  fill: green.lighten(60%),
)[A]
#let b = table.cell(
  fill: aqua.lighten(60%),
)[B]

#table(
  columns: 4,
  [], [Exam 1], [Exam 2], [Exam 3],

  [John], [], a, [],
  [Mary], [], a, a,
  [Robert], b, a, b,
)
```

| | Exam 1 | Exam 2 | Exam 3 |
|---|---|---|---|
| John | N/A | A | N/A |
| Mary | N/A | A | A |
| Robert | B | A | B |

## /2 Accesibilidad

Las tablas son difíciles de consumir para los usuarios de tecnologías de asistencia (TA). Para hacerles la vida más fácil a los usuarios de TA, te recomendamos enfáticamente que uses `table.header`[3.6.19/4.4] y `table.footer`[3.6.19/4.5] para marcar las secciones de encabezado y pie de tu tabla. Esto permitirá que las TA anuncien las etiquetas de columna de cada celda.

Como navegar una tabla por celdas es más engorroso que leerla visualmente, deberías considerar hacer que la información central de tu tabla también esté disponible como texto. Podés hacerlo envolviendo tu tabla en una figura[3.6.7] y usando su epígrafe para resumir el contenido de la tabla.

## /3 Parámetros

```
table(
  columns: auto int relative fraction array,
  rows: auto int relative fraction array,
  gutter: auto int relative fraction array,
  column-gutter: auto int relative fraction array,
  row-gutter: auto int relative fraction array,
  inset: relative array dictionary function,
  align: auto array function alignment,
  fill: none color gradient tiling array function,
  stroke: none length color gradient stroke tiling array dictionary function,
```

```
  ..content,
) → content
```

/3.1 **columns** auto or int or relative or fraction or array *Configurable con set* Por defecto: ()

Los tamaños de las columnas. Mirá la documentación de grid[3.10.10/1] para más información sobre el dimensionamiento de pistas.

/3.2 **rows** auto or int or relative or fraction or array *Configurable con set* Por defecto: ()

Los tamaños de las filas. Mirá la documentación de grid[3.10.10/1] para más información sobre el dimensionamiento de pistas.

/3.3 **gutter** auto or int or relative or fraction or array Por defecto: ()

Los espacios entre filas y columnas. Es un atajo para establecer `column-gutter` y `row-gutter` en el mismo valor. Mirá la documentación de grid[3.10.10/5.3] para más información sobre los gutters.

/3.4 **column-gutter** auto or int or relative or fraction or array *Configurable con set*
Por defecto: ()

Los espacios entre columnas. Tiene precedencia sobre `gutter`. Mirá la documentación de grid[3.10.10/5.3] para más información sobre los gutters.

/3.5 **row-gutter** auto or int or relative or fraction or array *Configurable con set*
Por defecto: ()

Los espacios entre filas. Tiene precedencia sobre `gutter`. Mirá la documentación de grid[3.10.10/5.3] para más información sobre los gutters.

/3.6 **inset** relative or array or dictionary or function *Configurable con set*
Por defecto: `0% + 5pt`

Cuánto relleno se agrega al contenido de las celdas.

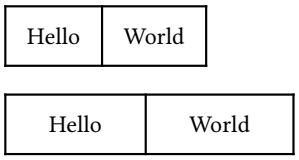
Para especificar el mismo inset para todas las celdas, usá una única longitud para todos los lados o un diccionario de longitudes para lados individuales. Mirá la documentación de box[3.10.5/2.7] para más detalles.

Para especificar un inset distinto para distintas celdas, podés:
- usar un único inset uniforme para todas las celdas
- usar un array de insets para cada columna
- usar una función que asigne la posición X/Y de una celda (ambas desde cero) a su inset

Mirá la documentación de grid[3.10.10/3] para más detalles.

```
#table(
  columns: 2,
  inset: 10pt,
  [Hello],
  [World],
)

#table(
  columns: 2,
  inset: (x: 20pt, y: 10pt),
  [Hello],
  [World],
)
```




/3.7 **align** auto or array or function or alignment *Configurable con set* Por defecto: auto

Cómo alinear el contenido de las celdas.

Si se establece en `auto`, se usa la alineación exterior.

Podés especificar la alineación de cualquiera de las siguientes maneras:
- usar una única alineación para todas las celdas
- usar un array de alineaciones correspondientes a cada columna
- usar una función que asigne la posición X/Y de una celda (ambas desde cero) a su alineación

Mirá la guía de tablas[4.3.6] para más detalles.

```
#table(
  columns: 3,
  align: (left, center, right),
  [Hello], [Hello], [Hello],
  [A], [B], [C],
)
```

| Hello | Hello | Hello |
|---|---|---|
| A | B | C |

/3.8 **fill** none or color or gradient or tiling or array or function *Configurable con set*

Por defecto: none

Cómo rellenar las celdas.

Puede ser:
- un único relleno para todas las celdas
- un array de rellenos correspondientes a cada columna
- una función que asigne la posición de una celda a su relleno

Sobre todo, los arrays y las funciones son útiles para crear tablas con rayas. Mirá la guía de tablas[4.3.4] para más detalles.

```
#table(
  fill: (x, _) =>
    if calc.odd(x) { luma(240) }
    else { white },
  align: (x, y) =>
    if y == 0 { center }
    else if x == 0 { left }
    else { right },
  columns: 4,
  [], [*Q1*], [*Q2*], [*Q3*],
  [Revenue:], [1000 €], [2000 €], [3000 €],
  [Expenses:], [500 €], [1000 €], [1500 €],
  [Profit:], [500 €], [1000 €], [1500 €],
)
```

| | Q1 | Q2 | Q3 |
|---|---|---|---|
| Revenue: | 1000 € | 2000 € | 3000 € |
| Expenses: | 500 € | 1000 € | 1500 € |
| Profit: | 500 € | 1000 € | 1500 € |

/3.9 **stroke** none or length or color or gradient or stroke or tiling or array or dictionary or function *Configurable con set*

Por defecto: stroke()

Cómo trazar[3.11.12] las celdas.

Los trazos se pueden desactivar estableciendo esto en none.

Si hace falta ubicar líneas que puedan cruzar el espacio entre celdas producido por la opción gutter[3.6.19/3.3], o reemplazar el trazo entre varias celdas específicas, considerá especificar uno o más de table.hline[3.6.19/4.2] y table.vline[3.6.19/4.3] junto con las celdas de tu tabla.

Para especificar el mismo trazo para todas las celdas, usá un único trazo[3.11.12] para todos los lados o un diccionario de trazos[3.11.12] para lados individuales. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

Para especificar trazos distintos para distintas celdas, podés:
- usar un único trazo para todas las celdas
- usar un array de trazos correspondientes a cada columna
- usar una función que asigne la posición de una celda a su trazo

Mirá la guía de tablas[4.3.5] para más detalles.

/3.10 **children** `content` *Obligatorio* *Posicional* *Variádico*

El contenido de las celdas de la tabla, más las líneas de tabla adicionales especificadas con los elementos `table.hline`[3.6.19/4.2] y `table.vline`[3.6.19/4.3].

## /4 Funciones

### /4.1 cell Elemento Desde: 0.11.0

Una celda de la tabla. Usala para posicionar una celda manualmente o para aplicar estilos. Para lo segundo, podés usar la función para reemplazar las propiedades de una celda en particular o usarla en reglas show para aplicar ciertos estilos a varias celdas a la vez.

Quizás el caso de uso más importante de `table.cell` sea hacer que una celda abarque varias columnas y/o filas con los campos `colspan` y `rowspan`.

```
#show table.cell.where(y: 0): strong
#set table(
  stroke: (x, y) => if y == 0 {
    (bottom: 0.7pt + black)
  },
  align: (x, y) => (
    if x > 0 { center }
    else { left }
  )
)

#table(
  columns: 3,
  table.header(
    [Substance],
    [Subcritical °C],
    [Supercritical °C],
  ),
  [Hydrochloric Acid],
  [12.0], [92.1],
  [Sodium Myreth Sulfate],
  [16.6], [104],
  [Potassium Hydroxide],
  table.cell(colspan: 2)[24.7],
)
```

| Substance | Subcritical °C | Supercritical °C |
|---|---|---|
| Hydrochloric Acid | 12.0 | 92.1 |
| Sodium Myreth Sulfate | 16.6 | 104 |
| Potassium Hydroxide | 24.7 | |

Por ejemplo, podés reemplazar el relleno, la alineación o el inset de una sola celda:

```
// You can also import those.
#import table: cell, header

#table(
  columns: 2,
  align: center,
  header(
    [*Trip progress*],
    [*Itinerary*],
```

| Trip progress | Itinerary |
|---|---|
|  | Get in, folks! |
|  | Eat curbside hotdog |
|  |  |

```
  ),
  cell(
    align: right,
    fill: fuchsia.lighten(80%),
    [🚗],
  ),
  [Get in, folks!],
  [🚗], [Eat curbside hotdog],
  cell(align: left)[🌴🚗],
  cell(
    inset: 0.06em,
    text(1.62em)[🏝️🌅🌊],
  ),
)
```

También podés aplicar una regla show sobre `table.cell` para dar estilo a todas las celdas a la vez. Combinada con selectores, esto te permite aplicar estilos según la posición de una celda:

```
#show table.cell.where(x: 0): strong

#table(
  columns: 3,
  gutter: 3pt,
  [Name], [Age], [Strength],
  [Hannes], [36], [Grace],
  [Irma], [50], [Resourcefulness],
  [Vikram], [49], [Perseverance],
)
```

| **Name** | Age | Strength |
|---|---|---|
| **Hannes** | 36 | Grace |
| **Irma** | 50 | Resourcefulness |
| **Vikram** | 49 | Perseverance |

```
table.cell(
  content,
  x: auto int,
  y: auto int,
  colspan: int,
  rowspan: int,
  inset: auto relative dictionary,
  align: auto alignment,
  fill: none auto color gradient tiling,
  stroke: none length color gradient stroke tiling dictionary,
  breakable: auto bool,
) → content
```

/4.1.1 **body** `content` *Obligatorio* *Posicional*

El cuerpo de la celda.

/4.1.2 **x** `auto` or `int` *Configurable con set* Por defecto: `auto`

La columna de la celda (con índice desde cero). Funciona igual que el campo x de `grid.cell`[3.10.10/6.1].

/4.1.3 **y** `auto` or `int` *Configurable con set* Por defecto: `auto`

La fila de la celda (con índice desde cero). Funciona igual que el campo y de `grid.cell`[3.10.10/6.1].

/4.1.4 **colspan** `int` *Configurable con set* Por defecto: 1

La cantidad de columnas que abarca esta celda.

/4.1.5 **rowspan** `int` *Configurable con set* Por defecto: 1

La cantidad de filas que abarca esta celda.

/4.1.6 **inset** `auto` or `relative` or `dictionary` *Configurable con set* Por defecto: `auto`

El reemplazo de inset[3.6.19/3.6] de la celda.

/4.1.7 **align** `auto` or `alignment` *Configurable con set* Por defecto: `auto`

El reemplazo de alineación[3.6.19/3.7] de la celda.

/4.1.8 **fill** `none` or `auto` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `auto`

El reemplazo de relleno[3.6.19/3.8] de la celda.

/4.1.9 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `(:)`

El reemplazo de trazo[3.6.19/3.9] de la celda.

/4.1.10 **breakable** `auto` or `bool` *Configurable con set* Por defecto: `auto`

Si las filas que abarca esta celda se pueden ubicar en páginas distintas. Cuando es igual a `auto`, una celda que abarca solo filas de tamaño fijo no se puede cortar, mientras que una celda que abarca al menos una fila de tamaño `auto` sí se puede cortar.

## /4.2 hline Elemento Desde: 0.11.0

Una línea horizontal en la tabla.

Reemplaza cualquier trazo por celda, incluido el trazo especificado mediante el campo `stroke` de la tabla. Puede cruzar el espacio entre celdas creado mediante la opción `column-gutter`[3.6.19/3.4] de la tabla.

Usá esta función en lugar del campo `stroke` de la tabla si querés ubicar manualmente una línea horizontal en una posición específica de una sola tabla. Considerá usar en cambio el campo `stroke` de la tabla[3.6.19/3.9] o el campo `stroke` de `table.cell`[3.6.19/4.1.9] si la línea que querés ubicar es parte del diseño de todas tus tablas.

```
#set table.hline(stroke: .6pt)

#table(
  stroke: none,
  columns: (auto, 1fr),
  [09:00], [Badge pick up],
  [09:45], [Opening Keynote],
  [10:30], [Talk: Typst's Future],
  [11:15], [Session: Good PRs],
  table.hline(start: 1),
  [Noon], [_Lunch break_],
  table.hline(start: 1),
  [14:00], [Talk: Tracked Layout],
  [15:00], [Talk: Automations],
  [16:00], [Workshop: Tables],
  table.hline(),
  [19:00], [Day 1 Attendee Mixer],
)
```

| | |
|---|---|
| 09:00 | Badge pick up |
| 09:45 | Opening Keynote |
| 10:30 | Talk: Typst's Future |
| 11:15 | Session: Good PRs |
| Noon | *Lunch break* |
| 14:00 | Talk: Tracked Layout |
| 15:00 | Talk: Automations |
| 16:00 | Workshop: Tables |
| 19:00 | Day 1 Attendee Mixer |

```
table.hline(
  y: auto int,
  start: int,
  end: none int,
  stroke: none length color gradient stroke tiling dictionary,
  position: alignment,
) → content
```

/4.2.1 **y** `auto` or `int` *Configurable con set* Por defecto: `auto`

La fila encima de la cual se ubica la línea horizontal (con índice desde cero). Funciona igual que el campo y de `grid.hline`[3.10.10/6.2.1].

/4.2.2 **start** `int` *Configurable con set* Por defecto: `0`

La columna en la que empieza la línea horizontal (con índice desde cero, inclusivo).

/4.2.3 **end** `none` or `int` *Configurable con set* Por defecto: none

La columna antes de la cual termina la línea horizontal (con índice desde cero, exclusivo).

/4.2.4 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `stroke()`

El trazo de la línea.

Especificar none elimina cualquier línea ubicada previamente a lo largo del rango de esta línea, incluidas las hlines o el trazo por celda que estén debajo.

/4.2.5 **position** `alignment` *Configurable con set* Por defecto: top

La posición en la que se ubica la línea, dada su fila (y): `top` para dibujarla por encima o `bottom` para dibujarla por debajo.

Esta configuración solo es relevante cuando el gutter de filas está habilitado (y no se debe usar de otro modo; conviene simplemente aumentar en uno el campo y), ya que en ese caso la posición debajo de una fila pasa a ser distinta de la posición encima de la fila siguiente, debido al espacio entre ambas.

## /4.3 `vline` Elemento Desde: 0.11.0

Una línea vertical en la tabla. Mirá la documentación de `grid.vline`[3.10.10/6.3] para más información sobre cómo usar los campos de este elemento.

Reemplaza cualquier trazo por celda, incluido el trazo especificado mediante el campo `stroke` de la tabla. Puede cruzar el espacio entre celdas creado mediante la opción `row-gutter`[3.6.19/3.5] de la tabla.

De manera similar a `table.hline`[3.6.19/4.2], usá esta función si querés ubicar manualmente una línea vertical en una posición específica de una sola tabla y usá en cambio el campo `stroke` de la tabla[3.6.19/3.9] o el campo `stroke` de `table.cell`[3.6.19/4.1.9] si la línea que querés ubicar es parte del diseño de todas tus tablas.

```
table.vline(
  x: auto int,
  start: int,
  end: none int,
  stroke: none length color gradient stroke tiling dictionary,
  position: alignment,
) → content
```

/4.3.1 **x** `auto` or `int` *Configurable con set* Por defecto: `auto`

La columna antes de la cual se ubica la línea vertical (con índice desde cero). Funciona igual que el campo x de `grid.vline`[3.10.10/6.3].

/4.3.2 **start** `int` *Configurable con set* Por defecto: `0`

La fila en la que empieza la línea vertical (con índice desde cero, inclusivo).

/4.3.3 **end** `none` or `int` *Configurable con set* Por defecto: none

La fila encima de la cual termina la línea vertical (con índice desde cero, exclusivo).

/4.3.4 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `stroke()`

El trazo de la línea.

Especificar none elimina cualquier línea ubicada previamente a lo largo del rango de esta línea, incluidas las vlines o el trazo por celda que estén debajo.

/4.3.5 **position** `alignment` *Configurable con set* Por defecto: start

La posición en la que se ubica la línea, dada su columna (x): `start` para dibujarla antes o end para dibujarla después.

También se aceptan los valores `left` y `right`, pero no se recomiendan, ya que hacen que tu tabla sea inconsistente entre documentos de izquierda a derecha y de derecha a izquierda.

Esta configuración solo es relevante cuando el gutter de columnas está habilitado (y no se debe usar de otro modo; conviene simplemente aumentar en uno el campo `x`), ya que en ese caso la posición después de una columna pasa a ser distinta de la posición antes de la columna siguiente, debido al espacio entre ambas.

### /4.4 header Elemento Desde: 0.11.0

Un encabezado de tabla repetible.

Deberías envolver las filas de encabezado de tus tablas en esta función, incluso si no pensás que tu tabla se corte entre páginas, porque Typst usa esta función para adjuntar metadatos de accesibilidad a las tablas y asegurar el acceso universal[4.4.1] a tu documento.

Podés usar el parámetro `repeat` para controlar si el encabezado de tu tabla se repetirá en todas las páginas.

Actualmente, esta función no es adecuada para crear una columna de encabezado o celdas de encabezado individuales. Usá celdas normales o, si estás exportando un PDF, también podés usar la función `pdf.header-cell`[3.14.4/7] para marcar una celda como celda de encabezado. Del mismo modo, podés usar `pdf.data-cell`[3.14.4/6] para marcar celdas de esta función como celdas de datos. Tené en cuenta que estas funciones no son definitivas y, por lo tanto, solo están disponibles cuando habilitás la característica `a11y-extras` (mirá la documentación del formato PDF[3.14.4] para más detalles).

```
#set page(height: 11.5em)
#set table(
  fill: (x, y) =>
    if x == 0 or y == 0 {
      gray.lighten(40%)
    },
  align: right,
)

#show table.cell.where(x: 0): strong
#show table.cell.where(y: 0): strong

#table(
  columns: 4,
  table.header(
    [], [Blue chip],
    [Fresh IPO], [Penny st'k],
  ),
  table.cell(
    rowspan: 6,
    align: horizon,
    rotate(-90deg, reflow: true)[
      *USD / day*
    ],
  ),
  [0.20], [104], [5],
  [3.17], [108], [4],
  [1.59], [84],  [1],
  [0.26], [98],  [15],
  [0.01], [195], [4],
  [7.34], [57],  [2],
)
```

| | Blue chip | Fresh IPO | Penny st'k |
|---|---|---|---|
| USD / day | 0.20 | 104 | 5 |
| | 3.17 | 108 | 4 |
| | 1.59 | 84 | 1 |
| | 0.26 | 98 | 15 |

| | Blue chip | Fresh IPO | Penny st'k |
|---|---|---|---|
| | 0.01 | 195 | 4 |
| | 7.34 | 57 | 2 |

```
table.header(
  repeat: bool,
  level: int,
```

```
  ..content,
) → content
```

/4.4.1 **repeat** bool *Configurable con set* Por defecto: true

Si este encabezado se debe repetir en todas las páginas.

/4.4.2 **level** int *Configurable con set* Por defecto: 1

El nivel del encabezado. No debe ser cero.

Esto permite repetir varios encabezados a la vez. Los encabezados con distintos niveles pueden repetirse juntos, siempre que tengan niveles ascendentes.

En particular, cuando un encabezado de nivel más bajo empieza a repetirse, todos los encabezados de nivel más alto o igual dejan de repetirse (son "reemplazados" por el nuevo encabezado).

/4.4.3 **children** content *Obligatorio Posicional Variádico*

Las celdas y líneas dentro del encabezado.

## /4.5 footer Elemento Desde: 0.11.0

Un pie de tabla repetible.

Igual que el elemento table.header[3.6.19/4.4], el pie puede repetirse en cada página de la tabla. Esto es útil para mejorar la legibilidad agregando las etiquetas de columna tanto en el encabezado como en el pie de una tabla grande, totales u otra información que deba ser visible en cada página.

No se pueden ubicar otras celdas de la tabla después del pie.

```
table.footer(
  repeat: bool,
  ..content,
) → content
```

/4.5.1 **repeat** bool *Configurable con set* Por defecto: true

Si este pie se debe repetir en todas las páginas.

/4.5.2 **children** content *Obligatorio Posicional Variádico*

Las celdas y líneas dentro del pie.

### 3.6.20 `terms` Elemento Desde: siempre

Una lista de términos y sus descripciones.

Muestra verticalmente una secuencia de términos y sus descripciones. Cuando las descripciones ocupan varias líneas, usan sangría francesa para comunicar la jerarquía visual.

#### /1 Ejemplo

```
/ Ligature: A merged glyph.
/ Kerning: A spacing adjustment
  between two adjacent letters.
```

**Ligature** A merged glyph.
**Kerning** A spacing adjustment between two adjacent letters.

#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: empezar una línea con una barra, seguida de un término, dos puntos y una descripción crea un elemento de lista de términos.

#### /3 Parámetros

```
terms(
  tight: bool,
  separator: content,
  indent: length,
  hanging-indent: length,
  spacing: auto length,
  ..content,
) → content
```

##### /3.1 `tight` `bool` *Configurable con set* Por defecto: `true`

Define el espaciado[3.6.20/3.5] por defecto de la lista de términos. Si es `false`, los elementos se separan con el espaciado de párrafo[3.6.14/4.2]. Si es `true`, en cambio usan el interlineado de párrafo[3.6.14/4.1]. Esto hace que la lista sea más compacta, lo que puede verse mejor si los elementos son cortos.

En modo marcado, el valor de este parámetro se determina según si los elementos están separados por una línea en blanco. Si los elementos se siguen directamente, se establece en `true`; si los elementos están separados por una línea en blanco, se establece en `false`. La compacidad definida en el marcado no se puede reemplazar con reglas set.

```
/ Fact: If a term list has a lot
  of text, and maybe other inline
  content, it should not be tight
  anymore.

/ Tip: To make it wide, simply
  insert a blank line between the
  items.
```

**Fact** If a term list has a lot of text, and maybe other inline content, it should not be tight anymore.

**Tip** To make it wide, simply insert a blank line between the items.

##### /3.2 `separator` `content` *Configurable con set* Por defecto: `h(amount: 0.6em, weak: true)`

El separador entre el elemento y la descripción.

Si querés separarlos simplemente con una cierta cantidad de espacio, usá `h(2cm, weak: true)` como separador y reemplazá `2cm` por la cantidad de espacio que desees.

```
#set terms(separator: [: ])

/ Colon: A nice separator symbol.
```

**Colon**: A nice separator symbol.

/3.3 **indent** `length` *Configurable con set* Por defecto: `0pt`

La sangría de cada elemento.

/3.4 **hanging-indent** `length` *Configurable con set* Por defecto: `2em`

La sangría francesa de la descripción.

Se suma a la `indent` de todo el elemento.

```
#set terms(hanging-indent: 0pt)
/ Term: This term list does not
  make use of hanging indents.
```

**Term** This term list does not make use of hanging indents.

/3.5 **spacing** `auto` or `length` *Configurable con set* Por defecto: `auto`

El espaciado entre los elementos de la lista de términos.

Si se establece en `auto`, usa el `leading`[3.6.14/4.1] del párrafo para las listas de términos compactas y el `spacing`[3.6.14/4.2] del párrafo para las listas de términos amplias (no compactas).

/3.6 **children** `content` *Obligatorio* *Posicional* *Variádico*

Los hijos de la lista de términos.

Al usar la sintaxis de lista de términos, los elementos adyacentes se reúnen automáticamente en listas de términos, incluso a través de construcciones como los bucles for.

```
#for (year, product) in (
  "1978": "TeX",
  "1984": "LaTeX",
  "2019": "Typst",
) [/ #product: Born in #year.]
```

**TeX** Born in 1978.
**LaTeX** Born in 1984.
**Typst** Born in 2019.

## /4 Funciones

/4.1 **item** Elemento Desde: 0.4.0

Un elemento de una lista de términos.

```
terms.item(
  content,
  content,
) → content
```

/4.1.1 **term** `content` *Obligatorio* *Posicional*

El término que describe el elemento de la lista.

/4.1.2 **description** `content` *Obligatorio* *Posicional*

La descripción del término.

### 3.6.21 `title` Elemento Desde: 0.14.0

Un título de documento.

Debe usarse para mostrar el título principal de todo el documento y debería aparecer una sola vez por documento. En cambio, los títulos[3.6.9] de nivel 1 están pensados para usarse en las secciones de nivel superior del documento.

Tené en cuenta que el material preliminar adicional (como una lista de autores) que debe aparecer junto con el título no pertenece a su cuerpo.

En la exportación a HTML, se muestra como un elemento `h1`, mientras que los títulos de nivel 1 se muestran como elementos `h2`.

#### /1 Ejemplo

```
#set document(
  title: [Interstellar Mail Delivery]
)

#title()

= Introduction
In recent years, ...
```




#### /2 Parámetros

```
title(auto content) → content
```

##### /2.1 **body** `auto` or `content` *Posicional* *Configurable con set* Por defecto: `auto`

El contenido del título.

Cuando se omite (o es `auto`), por defecto será `document.title`[3.6.5/3.3]. En ese caso, se debe haber establecido previamente un título de documento con `set document(title: [..])`.

```
#set document(title: [Course ABC, Homework 1])
#title[Homework 1]

...
```



# 3.7 Texto

Estilos de texto.

La función text[3.7.12] es de particular interés.

## Definiciones


3.7.1 Highlight Highlights text with a background color. . . . 236
3.7.2 Line Break Inserts a line break. . . . 238
3.7.3 Lorem Creates blind text. . . . 239
3.7.4 Lowercase Converts a string or content to lowercase. . . . 240
3.7.5 Overline Adds a line over text. . . . 241
3.7.6 Raw Text / Code Raw text with optional syntax highlighting. . . . 243
3.7.7 Small Capitals Displays text in small capitals. . . . 255
3.7.8 Smartquote A language-aware quote that reacts to its context. . . . 257
3.7.9 Strikethrough Strikes through text. . . . 259
3.7.10 Subscript Renders text in subscript. . . . 261
3.7.11 Superscript Renders text in superscript. . . . 262
3.7.12 Text Customizes the look and layout of text in a variety of ways. . . . 264
3.7.13 Underline Underlines text. . . . 278
3.7.14 Uppercase Converts a string or content to uppercase. . . . 280

## 3.7.1 highlight Elemento Desde: 0.8.0

Resalta texto con un color de fondo.

### /1 Ejemplo

```
This is #highlight[important].
```

This is important.

### /2 Parámetros

```
highlight(
  fill: none color gradient tiling,
  stroke: none length color gradient stroke tiling dictionary,
  top-edge: length str,
  bottom-edge: length str,
  extent: length,
  radius: relative dictionary,
  content,
) → content
```

#### /2.1 fill none or color or gradient or tiling *Configurable con set* Por defecto: `rgb("#fffd11a1")`

El color con el que se resalta el texto.

```
This is #highlight(
  fill: blue
)[highlighted with blue].
```

This is highlighted with blue.

#### /2.2 stroke none or length or color or gradient or stroke or tiling or dictionary

*Configurable con set* Por defecto: `(:)`

El color del borde del resaltado. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

```
This is a #highlight(
  stroke: fuchsia
)[stroked highlighting].
```

This is a stroked highlighting.

#### /2.3 top-edge length or str *Configurable con set* Por defecto: `"ascender"`

El extremo superior del rectángulo de fondo.

```
#set highlight(top-edge: "ascender")
#highlight[a] #highlight[aib]

#set highlight(top-edge: "x-height")
#highlight[a] #highlight[aib]
```

a aib

a aib

| Variante | Detalles |
|---|---|
| `"ascender"` | El ascendente de la fuente, que normalmente supera la altura de todos los glifos. |
| `"cap-height"` | La altura aproximada de las letras mayúsculas. |
| `"x-height"` | La altura aproximada de las letras minúsculas sin ascendentes. |
| `"baseline"` | La línea base sobre la que se apoyan las letras. |
| `"bounds"` | El borde superior del cuadro delimitador del glifo. |

/2.4 **bottom-edge** `length` or `str` *Configurable con set* Por defecto: `"descender"`

El extremo inferior del rectángulo de fondo.

```
#set highlight(bottom-edge: "descender")
#highlight[a] #highlight[ap]

#set highlight(bottom-edge: "baseline")
#highlight[a] #highlight[ap]
```

a ap

a ap

| Variante | Detalles |
|---|---|
| `"baseline"` | La línea base sobre la que se apoyan las letras. |
| `"descender"` | El descendente de la fuente, que normalmente supera la profundidad de todos los glifos. |
| `"bounds"` | El borde inferior del cuadro delimitador del glifo. |

/2.5 **extent** `length` *Configurable con set* Por defecto: `0pt`

La cantidad en que se extiende el fondo hacia los lados más allá del contenido (o hacia adentro, si es negativa).

```
A long #highlight(extent: 4pt)[background].
```

A long background.

/2.6 **radius** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se redondean las esquinas del resaltado. Mirá la documentación del rectángulo[3.11.10/2.5] para más detalles.

```
Listen #highlight(
  radius: 5pt, extent: 2pt
)[carefully], it will be on the test.
```

Listen carefully, it will be on the test.

/2.7 **body** `content` *Obligatorio* *Posicional*

El contenido que se debe resaltar.

## 3.7.2 `linebreak` Elemento Desde: siempre

Inserta un salto de línea.

Avanza el párrafo a la línea siguiente. Un único salto de línea al final de un párrafo se ignora, pero más de uno crea líneas vacías adicionales.

### /1 Ejemplo

```
*Date:* 26.12.2022 \
*Topic:* Infrastructure Test \
*Severity:* High \
```

**Date:** 26.12.2022
**Topic:** Infrastructure Test
**Severity:** High

### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: para insertar un salto de línea, simplemente escribí una barra invertida seguida de un espacio en blanco. Esto siempre crea un salto sin justificar.

### /3 Parámetros

```
linebreak(justify: bool) → content
```

#### /3.1 `justify` `bool` *Configurable con set* Por defecto: `false`

Si se justifica la línea antes del salto.

Esto es útil si encontraste una mejor oportunidad de salto de línea en tu texto justificado que la que encontró Typst.

```
#set par(justify: true)
#let jb = linebreak(justify: true)

I have manually tuned the #jb
line breaks in this paragraph #jb
for an _interesting_ result. #jb
```

I have manually tuned the line breaks in this paragraph for an *interesting* result.

### 3.7.3 `lorem` Desde: siempre

Crea texto de relleno.

Esta función produce un texto de relleno *Lorem Ipsum* de apariencia latina con la cantidad de palabras indicada. La secuencia de palabras que genera la función siempre es la misma, pero elegida al azar. Como es habitual en los textos de relleno, no tiene ningún sentido. Usalo como marcador de posición para probar diseños.

#### /1 Ejemplo

```
= Blind Text
#lorem(30)

= More Blind Text
#lorem(15)
```

**Blind Text**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

**More Blind Text**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

#### /2 Parámetros

```
lorem(int) → str
```

/2.1 **`words`** `int` *Obligatorio* *Posicional*

La longitud del texto de relleno en palabras.

### 3.7.4 `lower` Desde: siempre

Convierte una cadena de texto o contenido a minúsculas.

#### /1 Ejemplo

```
#lower("ABC") \
#lower[*My Text*] \
#lower[already low]
```

abc
**my text**
already low

#### /2 Parámetros

```
lower( str  content ) → str  content
```

**/2.1 `text`** `str` or `content` *Obligatorio* *Posicional*

El texto que se convierte a minúsculas.

## 3.7.5 `overline` Elemento Desde: siempre

Agrega una línea sobre el texto.

### /1 Ejemplo

```
#overline[A line over text.]
```

A line over text.

### /2 Parámetros

```
overline(
  stroke: auto length color gradient stroke tiling dictionary,
  offset: auto length,
  extent: length,
  evade: bool,
  background: bool,
  content,
) → content
```

#### /2.1 `stroke` `auto` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary`

*Configurable con set* Por defecto: `auto`

Cómo trazar[3.11.12] la línea.

Si se establece en `auto`, toma el color del texto y un grosor definido en la fuente actual.

```
#set text(fill: olive)
#overline(
  stroke: green.darken(20%),
  offset: -12pt,
  [The Forest Theme],
)
```

The Forest Theme

#### /2.2 `offset` `auto` or `length` *Configurable con set*

Por defecto: `auto`

La posición de la línea relativa a la línea base. Se lee de las tablas de la fuente si es `auto`.

```
#overline(offset: -1.2em)[
  The Tale Of A Faraway Line II
]
```

The Tale Of A Faraway Line II

#### /2.3 `extent` `length` *Configurable con set*

Por defecto: `0pt`

La cantidad en que se extiende la línea más allá del contenido (o hacia adentro, si es negativa).

```
#set overline(extent: 4pt)
#set underline(extent: 4pt)
#overline(underline[Typography Today])
```

Typography Today

#### /2.4 `evade` `bool` *Configurable con set*

Por defecto: `true`

Si la línea omite las secciones en las que chocaría con los glifos.

```
#overline(
  evade: false,
  offset: -7.5pt,
  stroke: 1pt,
  extent: 3pt,
  [Temple],
)
```

Temple

/2.5 **background** `bool` *Configurable con set* Por defecto: `false`

Si la línea se ubica detrás del contenido sobre el que va.

```
#set overline(stroke: aqua + 5pt)
#overline(background: true)[Fully visible.] \
#overline(background: false)[Partially hidden.]
```

Fully visible.
Partially hidden.

/2.6 **body** `content` *Obligatorio* *Posicional*

El contenido sobre el que se agrega una línea.

### 3.7.6 `raw` Elemento Desde: siempre

Texto sin procesar (raw) con resaltado de sintaxis opcional.

Muestra el texto literalmente y en una fuente monoespaciada. Normalmente se usa para incrustar código informático en un documento.

El texto que se le da a este elemento ignorará la sintaxis de marcado, como `*strong*` o `_emphasis_`, y se mostrará literalmente. Si querés mostrar contenido con una fuente monoespaciada pero permitiendo la sintaxis de marcado, en lugar de usar `raw`[3.7.6], podés establecer explícitamente la fuente del texto en una fuente monoespaciada con el parámetro `text.font`[3.7.12/2.1].

Los elementos raw se producen principalmente con su sintaxis dedicada[3.7.6/3], encerrando el texto entre uno o tres o más caracteres de acento grave (`` ` ``) a ambos lados. Al usar tres o más acentos graves, el texto que sigue inmediatamente a los acentos graves iniciales se tratará como una etiqueta de lenguaje[3.7.6/4.3] usada para el resaltado de sintaxis, y el texto raw empieza después del primer espacio en blanco.

#### /1 Ejemplo

````
Adding `rbx` to `rcx` gives
the desired result.

What is ```rust fn main()``` in Rust
would be ```c int main()``` in C.

```rust
fn main() {
    println!("Hello World!");
}
```

This has ``` `backticks` ``` in it
(but the spaces are trimmed). And
``` here``` the leading space is
also trimmed.
````

Adding `rbx` to `rcx` gives the desired result.

What is `fn main()` in Rust would be `int main()` in C.

```rust
fn main() {
    println!("Hello World!");
}
```

This has `` `backticks` `` in it (but the spaces are trimmed). And `here` the leading space is also trimmed.

También podés construir un elemento `raw`[3.7.6] programáticamente a partir de una cadena de texto (y proporcionar la etiqueta de lenguaje mediante el parámetro opcional `lang`[3.7.6/4.3]).

```
#raw("fn " + "main() {}", lang: "rust")
```

```rust
fn main() {}
```

Si no hay resaltado de sintaxis disponible por defecto para la etiqueta de lenguaje que especificaste (o si querés reemplazar la definición incorporada), podés proporcionar un archivo de especificación de sintaxis personalizado en el parámetro `syntaxes`[3.7.6/4.5].

#### /2 Estilos

Por defecto, el elemento `raw` usa la fuente `DejaVu Sans Mono` (incluida con Typst), con un tamaño de fuente menor de `0.8em` (es decir, el 80% del tamaño de fuente global). Esto se debe a que las fuentes monoespaciadas tienden a verse más grandes que las que no lo son.

Podés personalizar estas propiedades con reglas show-set:

```
// Switch to Cascadia Code for both
// inline and block raw.
#show raw: set text(font: "Cascadia Code")

// Reset raw blocks to the same size as normal text,
// but keep inline raw at the reduced size.
```
```

```
#show raw.where(block: true): set text(1em / 0.8)

Now using the `Cascadia Code` font for raw text.
Here's some Python code. It looks larger now:

```py
def python():
  return 5 + 5
```
```

Now using the `Cascadia Code` font for raw text. Here's some Python code. It looks larger now:

```
def python():
  return 5 + 5
```

Además, podés personalizar los colores del resaltado de sintaxis estableciendo un tema personalizado con el parámetro `theme`[3.7.6/4.6].

Para personalizar por completo el aspecto de un bloque raw, puede ser útil una regla show sobre `raw.line`[3.7.6/5.1], por ejemplo para agregar números de línea.

Tené en cuenta que, en el texto raw, las características de composición como la separación en sílabas[3.7.12/2.20], el overhang[3.7.12/2.13], el espaciado CJK-latino[3.7.12/2.11] y (para los bloques raw) la justificación[3.6.14/4.3] estarán desactivadas por defecto.

## /3 Sintaxis

Esta función tiene una sintaxis dedicada que produce un elemento raw tanto en modo marcado como en modo código. Podés encerrar el texto entre uno o tres o más caracteres de acento grave (`) a ambos lados para convertirlo en raw. La cantidad de acentos graves debe ser la misma a ambos lados y el texto encerrado no puede contener un grupo de esa cantidad de acentos graves seguidos. Escribir solo dos acentos graves (``) produce texto raw vacío.

Entre las diferencias notables con Markdown se cuentan que los acentos graves simples pueden encerrar texto que abarca varias líneas sin quitar la sangría, y que la sintaxis de tres o más acentos graves sigue interpretando las etiquetas de lenguaje cuando se usa en línea.

El texto raw encerrado entre acentos graves *simples* no tiene forma de especificar una etiqueta de lenguaje y siempre se trata como en línea para usar dentro de un párrafo, es decir, el parámetro `block`[3.7.6/4.2] es `false`.

La sintaxis raw con *tres o más* acentos graves tiene las siguientes propiedades:

- **Después de los acentos graves iniciales, el bloque raw solo termina con una secuencia de la misma cantidad de acentos graves**

  Para incluir texto que contiene una secuencia de acentos graves, los acentos graves inicial y final deben tener al menos un acento grave más que la secuencia.

- **Si el texto raw contiene un salto de línea, será de bloque; de lo contrario, será en línea**

  Esto establece el parámetro `block`[3.7.6/4.2] en `true` o `false` según corresponda.

- **El texto que sigue inmediatamente a los acentos graves iniciales, hasta el primer espacio en blanco, se trata como una *etiqueta de lenguaje* usada para el resaltado de sintaxis**

  Está previsto cambiar las reglas específicas sobre qué texto se puede tratar como etiqueta de lenguaje, y están explicadas en detalle más abajo.[3.7.6/3.2]

- **Las líneas inicial y final tienen un comportamiento de recorte especial**

En la línea inicial, si todos los caracteres que siguen a los acentos graves iniciales o a la etiqueta de lenguaje son espacios en blanco, se recortará la línea entera. Sin embargo, si hay caracteres que no son espacios en blanco en esa línea, solo se recortará un único espacio que siga inmediatamente a los acentos graves iniciales o a la etiqueta de lenguaje, si lo hay.

Si la línea final es íntegramente espacios en blanco hasta los acentos graves de cierre, se recortará. De lo contrario, si el último carácter que no es un espacio en blanco de la línea final es un acento grave, se recortará un carácter de espacio del final de la línea, si lo hay.

- **La sangría común al comienzo de las líneas se recorta**

  Typst quitará los espacios en blanco iniciales al comienzo de las líneas del texto raw que son compartidos por todas las líneas, es decir, la sangría común. Aunque esto excluye el texto de la línea con los acentos graves iniciales.

  Typst primero encuentra la línea con la menor cantidad de caracteres de espacio en blanco iniciales que contiene algún carácter que no es un espacio en blanco, incluida la línea con los acentos graves de cierre. Luego Typst recorta de cada línea una cantidad de caracteres igual a la cantidad de caracteres de espacio en blanco iniciales de esa línea. Las líneas que son solo espacios en blanco quitarán la misma cantidad de caracteres hasta quedar vacías, pero conservarán cualquier espacio en blanco final adicional.

  Tené en cuenta que, por simplicidad, esta comprobación trata las tabulaciones y los espacios como caracteres equivalentes y que opera sobre cantidades de puntos de código Unicode[A], es decir, caracteres, no sobre longitudes en bytes.

Estas propiedades de la sintaxis de tres o más acentos graves permiten algunos casos de uso que pueden no ser evidentes:

- Para escribir texto que contiene una secuencia de acentos graves, encerralo con uno o más acentos graves que la secuencia: ```` ``` enclosed```backticks```` ````
- Para escribir texto que empieza o termina con un acento grave, agregá un espacio dentro de los acentos graves de apertura y de cierre: ``` `backticks` ```
- Para escribir texto en línea resaltado con una etiqueta de lenguaje, agregá un espacio entre la etiqueta de lenguaje y el texto ```` ```rust fn main() {}``` ````
- Para escribir texto en línea sin ninguna etiqueta de lenguaje, agregá un espacio después de los acentos graves iniciales: ```` ``` text``` ```` o usá la sintaxis de acento grave simple: `` `text` ``

## /3.1 Incrustar cadenas de texto con la sintaxis raw

Un caso de uso común de la sintaxis raw es incrustar datos como cadenas de texto con formato, accediendo al campo `.text` del contenido raw para obtener la cadena subyacente. Esto también se puede combinar con el constructor de `bytes`[3.5.7] para convertir la cadena en bytes.

````
An inline YAML dictionary via `.text`

#yaml(bytes(
  ```yaml
  Magic:
    limited-by: Mana
  Pokémon:
    limited-by: Energy
  Yu-Gi-Oh:
    limited-by: false
  ```.text
  //  ^^^^ used as a string
))
````

An inline YAML dictionary via `.text`

```
(
  Magic: (limited-by: "Mana"),
  Pokémon: (limited-by: "Energy"),
  Yu-Gi-Oh: (limited-by: false),
)
```

[A] https://www.unicode.org/glossary/#code_point

### /3.2 Cambios en las etiquetas de lenguaje

Al usar la sintaxis raw con tres o más acentos graves, el texto que sigue inmediatamente a los acentos graves iniciales (hasta el primer espacio en blanco) se trata como una etiqueta de lenguaje[3.7.6/4.3]. Sin embargo, en la versión actual de Typst, solo se trata como etiqueta de lenguaje el texto que sería un identificador válido de Typst. El primer carácter que no sea válido para un identificador se interpretará como el inicio del texto raw.

Por ejemplo, en la versión actual de Typst, si un bloque raw empieza con `C++`, el identificador `C` será la etiqueta de lenguaje y el texto raw empezará con `++`. Si un bloque raw empieza con `++C`, no tendrá etiqueta de lenguaje y el texto raw empezará con `++C`.

Para usar etiquetas de lenguaje que no son válidas como identificadores en la versión actual de Typst, tenés que usar el parámetro `lang`[3.7.6/4.3], ya sea llamando al constructor con una cadena de texto: `#raw("text", lang: "...")`, o escribiendo una regla set: `#set raw(lang: "...")`.

En la próxima versión de Typst, *todo el texto* hasta el primer espacio en blanco o acento grave se tratará como la etiqueta de lenguaje, lo que permite un conjunto de caracteres más amplio para las etiquetas de lenguaje. Las etiquetas que incluyan espacios o acentos graves seguirán necesitando establecerse manualmente mediante el parámetro `lang`[3.7.6/4.3].

Typst te avisará, emitiendo una advertencia, si tus bloques raw se van a interpretar de manera distinta en la próxima versión de Typst.

## /4 Parámetros

```
raw(
  str,
  block: bool,
  lang: none str,
  align: alignment,
  syntaxes: str path bytes array,
  theme: none auto str path bytes,
  tab-size: int,
) → content
```

### /4.1 **text** `str` *Obligatorio* *Posicional*

El texto raw.

También podés usar los bloques raw de forma creativa para crear sintaxis personalizadas para tus automatizaciones.

Example · Implementar un DSL con raw y reglas show

```
// Parse numbers in raw blocks with the
// `mydsl` tag and sum them up.
#show raw.where(lang: "mydsl"): it => {
  let sum = 0
  for part in it.text.split("+") {
    sum += int(part.trim())
  }
  sum
}

```mydsl
1 + 2 + 3 + 4 + 5
```
```

```
15
```

### /4.2 **block** `bool` *Configurable con set* Por defecto: `false`

Si el texto raw se muestra como un bloque separado.

En modo marcado, usar la notación de un acento grave lo establece en `false`. Usar la notación de tres acentos graves lo establece en `true` si el contenido encerrado contiene al menos un salto de línea.

```
// Display inline code in a small box
// that retains the correct baseline.
#show raw.where(block: false): box.with(
  fill: luma(240),
  inset: (x: 3pt, y: 0pt),
  outset: (y: 3pt),
  radius: 2pt,
)

// Display block code in a larger block
// with more padding.
#show raw.where(block: true): block.with(
  fill: luma(240),
  inset: 10pt,
  radius: 4pt,
)

With `rg`, you can search through your files quickly.
This example searches the current directory recursively
for the text `Hello World`:

```bash
rg "Hello World"
```
```

With `rg`, you can search through your files quickly. This example searches the current directory recursively for the text `Hello World`:

```
rg "Hello World"
```

/4.3 **lang** `none` or `str` *Configurable con set* Por defecto: none

El lenguaje con el que se interpreta el texto raw para el resaltado de sintaxis.

En la exportación a HTML[3.14.3], esto establece el atributo `data-lang` del elemento `html.code`[3.14.3/6.2.19] generado.

Además de las etiquetas de lenguaje típicas conocidas de Markdown, admite las etiquetas `"typ"`, `"typc"` y `"typm"` para marcado de Typst[3.1.2], código de Typst[3.1.4] y matemática de Typst[3.1.3], respectivamente.

*Sintaxis disponibles*

| Nombre | Etiquetas |
|---|---|
| Typst | `typ`, `typst` |
| Typst (code) | `typc` |
| Typst (math) | `typm` |
| ActionScript | `as` |
| Ada | `adb`, `ads`, `gpr` |
| Apache Conf | `envvars`, `htaccess`, `HTACCESS`, `htgroups`, `HTGROUPS`, `htpasswd`, `HTPASSWD`, `.htaccess`, `.HTACCESS`, `.htgroups`, `.HTGROUPS`, `.htpasswd`, `.HTPASSWD` |

| Nombre | Etiquetas |
|---|---|
| AppleScript | `applescript`, `script editor` |
| AsciiDoc (Asciidoctor) | `adoc`, `ad`, `asciidoc` |
| ASP | `asa` |
| Assembly (x86_64) | `yasm`, `nasm`, `asm`, `inc`, `mac` |
| Authorized Keys | `authorized_keys`, `pub`, `authorized_keys2` |
| AWK | `awk` |
| Batch File | `bat`, `cmd` |
| BibTeX | `bib` |
| Bourne Again Shell (bash) | `sh`, `bash`, `zsh`, `ash`, `.bash_aliases`, `.bash_completions`, `.bash_functions`, `.bash_login`, `.bash_logout`, `.bash_profile`, `.bash_variables`, `.bashrc`, `.profile`, `.textmate_init`, `.zlogin`, `.zlogout`, `.zprofile`, `.zshenv`, `.zshrc`, `PKGBUILD`, `ebuild`, `eclass` |
| C | `c`, `h` |
| C# | `cs`, `csx` |
| C++ | `cpp`, `cc`, `cp`, `cxx`, `c++`, `C`, `h`, `hh`, `hpp`, `hxx`, `h++`, `inl`, `ipp` |
| Cabal | `cabal` |
| CFML | `cfml`, `cfm`, `cfc` |
| Clojure | `clj`, `cljc`, `cljs`, `edn` |
| CMake | `CMakeLists.txt`, `cmake` |
| CMake C Header | `h.in` |
| CMake C++ Header | `hh.in`, `hpp.in`, `hxx.in`, `h++.in` |
| CMakeCache | `CMakeCache.txt` |
| CoffeeScript | `coffee`, `Cakefile`, `coffee.erb`, `cson` |
| Comma Separated Values | `csv`, `tsv` |
| CpuInfo | `cpuinfo` |
| Crontab | `tab`, `crontab`, `cron.d` |
| Crystal | `cr` |
| CSS | `css`, `css.erb`, `css.liquid` |
| D | `d`, `di` |
| Dart | `dart` |
| Diff | `diff`, `patch` |
| Dockerfile | `Dockerfile`, `dockerfile`, `.Dockerfile` |
| DotENV | `.env`, `.env.dist`, `.env.local`, `.env.sample`, `.env.example`, `.env.template`, `.env.test`, `.env.test.local`, `.env.testing`, `.env.dev`, `.env.development`, `.env.development.local`, `.env.prod`, `.env.production`, `.env.production.local`, `.env.dusk.local`, `.env.staging`, `.env.default`, `.env.defaults`, `.envrc`, `.flaskenv`, `env`, `env.example`, `env.sample`, `env.template` |
| Elixir | `ex`, `exs` |
| Elm | `elm` |
| Email | `eml`, `msg`, `mbx`, `mboxz` |
| Erlang | `erl`, `hrl`, `Emakefile`, `emakefile`, `escript` |
| F# | `fs`, `fsi`, `fsx` |
| Fish | `fish` |
| Fortran (Fixed Form) | `f`, `F`, `f77`, `F77`, `for`, `FOR`, `fpp`, `FPP` |
| Fortran (Modern) | `f90`, `F90`, `f95`, `F95`, `f03`, `F03`, `f08`, `F08` |
| Fortran Namelist | `namelist` |

| Nombre | Etiquetas |
|---|---|
| fstab | fstab, crypttab, mtab |
| Git Attributes | attributes, gitattributes, .gitattributes |
| Git Commit | COMMIT_EDITMSG, MERGE_MSG, TAG_EDITMSG |
| Git Config | gitconfig, .gitconfig, .gitmodules |
| Git Ignore | exclude, gitignore, .gitignore |
| Git Link | .git |
| Git Log | gitlog |
| Git Mailmap | .mailmap, mailmap |
| Git Rebase Todo | git-rebase-todo |
| GLSL | vs, fs, gs, vsh, fsh, gsh, vshader, fshader, gshader, vert, frag, geom, tesc, tese, comp, glsl, mesh, task, rgen, rint, rahit, rchit, rmiss, rcall |
| gnuplot | gp, gpl, gnuplot, gnu, plot, plt |
| Go | go |
| GraphQL | graphql, graphqls, gql |
| Graphviz (DOT) | dot, DOT, gv |
| Groff/troff | groff, troff, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| Groovy | groovy, gvy, gradle, Jenkinsfile |
| group | group |
| Haskell | hs |
| Highlight non-printables | show-nonprintable |
| hosts | hosts |
| HTML | html, htm, shtml, xhtml |
| HTML (ASP) | asp |
| HTML (EEx) | html.eex, html.leex |
| HTML (Erlang) | yaws |
| HTML (Jinja2) | htm.j2, html.j2, xhtml.j2, xml.j2 |
| HTML (Rails) | rails, rhtml, erb, html.erb |
| HTML (Tcl) | adp |
| HTML (Twig) | twig, html.twig |
| HTTP Request and Response | http |
| INI | ini, INI, inf, INF, reg, REG, lng, cfg, CFG, desktop, url, URL, .editorconfig, .coveragerc, .pylintrc, .gitlint, .hgrc, hgrc |
| Java | java, bsh |
| Java Properties | properties |
| Java Server Page (JSP) | jsp |
| JavaScript | js, htc |
| JavaScript (Rails) | js.erb |
| Jinja2 | j2, jinja2, jinja |
| JQ | jq |
| JSON | json, sublime-settings, sublime-menu, sublime-keymap, sublime-mousemap, sublime-theme, sublime-build, sublime-project, sublime-completions, sublime-commands, sublime-macro, sublime-color-scheme, ipynb, Pipfile.lock |
| JSON (Terraform) | tfstate |
| jsonnet | jsonnet, libsonnet, libjsonnet |
| Julia | jl |

| Nombre | Etiquetas |
|---|---|
| Known Hosts | known_hosts, known_hosts.old |
| Kotlin | kt, kts |
| LaTeX | tex, ltx |
| Lean | lean |
| Less | less, css.less |
| Lisp | lisp, cl, clisp, l, mud, el, scm, ss, lsp, fasl, sld |
| Literate Haskell | lhs |
| LLVM | ll |
| log | log |
| Lua | lua |
| Makefile | make, GNUmakefile, makefile, Makefile, makefile.am, Makefile.am, makefile.in, Makefile.in, OCamlMakefile, mak, mk |
| Manpage | man |
| Markdown | md, mdown, markdown, markdn |
| MATLAB | matlab |
| MediaWiki | mediawiki, wikipedia, wiki |
| MemInfo | meminfo |
| NAnt Build File | build |
| nginx | conf.erb, conf, nginx.conf, mime.types, fastcgi_params, scgi_params, uwsgi_params |
| Nim | nim, nims, nimble |
| Ninja | ninja |
| Nix | nix |
| NSIS | nsi, nsh, bnsi, bnsh, nsdinc |
| Objective-C | m, h |
| Objective-C++ | mm, M, h |
| OCaml | ml, mli |
| OCamllex | mll |
| OCamlyacc | mly |
| orgmode | org |
| Pascal | pas, p, dpr |
| passwd | passwd |
| Perl | pl, pc, pm, pmc, pod, t |
| PHP | php, php3, php4, php5, php7, phps, phpt, phtml |
| Plain Text | txt |
| Protocol Buffer | proto, protobuf, protodevel |
| Protocol Buffer (TEXT) | pb.txt, proto.text, textpb, pbtxt, prototxt, textproto |
| Puppet | pp, epp |
| PureScript | purs |
| Python | py, py3, pyw, pyi, pyx, pyx.in, pxd, pxd.in, pxi, pxi.in, rpy, cpy, SConstruct, Sconstruct, sconstruct, SConscript, gyp, gypi, Snakefile, vpy, wscript, bazel, bzl |
| QML | qml, qmlproject |
| R | R, r, Rprofile |
| Racket | rkt |
| Rd (R Documentation) | rd |

| Nombre | Etiquetas |
|---|---|
| Rego | `rego` |
| Regular Expression | `re` |
| Requirements.txt | `requirements.txt`, `requirements.in`, `pip` |
| resolv | `resolv.conf` |
| reStructuredText | `rst`, `rest` |
| Robot Framework | `robot`, `resource` |
| Ruby | `rb`, `Appfile`, `Appraisals`, `Berksfile`, `Brewfile`, `capfile`, `cgi`, `Cheffile`, `config.ru`, `Deliverfile`, `Fastfile`, `fcgi`, `Gemfile`, `gemspec`, `Guardfile`, `irbrc`, `jbuilder`, `Podfile`, `podspec`, `prawn`, `rabl`, `rake`, `Rakefile`, `Rantfile`, `rbx`, `rjs`, `ruby.rail`, `Scanfile`, `simplecov`, `Snapfile`, `thor`, `Thorfile`, `Vagrantfile` |
| Ruby Haml | `haml`, `sass` |
| Ruby on Rails | `rxml`, `builder` |
| Ruby Slim | `slim`, `skim` |
| Rust | `rs` |
| Scala | `scala`, `sbt`, `sc` |
| SCSS | `scss` |
| SML | `sml`, `cm`, `sig` |
| Solidity | `sol` |
| SQL | `sql`, `ddl`, `dml` |
| SQL (Rails) | `erbsql`, `sql.erb` |
| SSH Config | `ssh_config` |
| SSHD Config | `sshd_config` |
| Strace | `strace` |
| Stylus | `styl`, `stylus` |
| Svelte | `svlt`, `svelte` |
| Swift | `swift` |
| syslog | `syslog` |
| SystemVerilog | `sv`, `v`, `svh`, `vh` |
| Tcl | `tcl` |
| Terraform | `tf`, `tfvars`, `hcl` |
| TeX | `sty`, `cls` |
| Textile | `textile` |
| Todo.txt | `todo.txt`, `done.txt` |
| TOML | `toml`, `tml`, `Cargo.lock`, `Gopkg.lock`, `Pipfile`, `pdm.lock`, `poetry.lock`, `uv.lock` |
| TypeScript | `ts`, `mts`, `cts` |
| TypeScriptReact | `tsx` |
| varlink | `varlink` |
| Verilog | `v`, `V` |
| VimL | `vim`, `vimrc`, `gvimrc`, `.vimrc`, `.gvimrc`, `_vimrc`, `_gvimrc` |
| Vue Component | `vue` |
| Vyper | `vy` |
| WGSL | `wgsl` |
| XML | `xml`, `xsd`, `xslt`, `tld`, `dtml`, `rng`, `rss`, `opml`, `svg`, `xaml` |
| YAML | `yaml`, `yml`, `sublime-syntax` |

| Nombre | Etiquetas |
|---|---|
| Zig | zig |

````
```typ
This is *Typst!*
```

This is ```typ also *Typst*```, but inline!
````

This is ***Typst!***

This is also ***Typst***, but inline!

### /4.4 align `alignment` *Configurable con set* Por defecto: `start`

La alineación horizontal que debe tener cada línea de un bloque raw. Esta opción se ignora si no es un bloque raw (si se especificó `block: false` o se usaron acentos graves simples en modo marcado).

Por defecto, se establece en `start`, lo que significa que el texto raw se alinea hacia el inicio de la dirección del texto dentro del bloque, sin importar la alineación del contexto actual (lo que te permite, por ejemplo, centrar el bloque raw en sí sin centrar el texto que contiene).

````
#set raw(align: center)

```typc
let f(x) = x
code = "centered"
```
````

```
  let f(x) = x
code = "centered"
```

### /4.5 syntaxes `str` or `path` or `bytes` or `array` *Configurable con set* Por defecto: ()

Definiciones de sintaxis adicionales que se cargan. Las definiciones de sintaxis deben estar en el formato de archivo `sublime-syntax`[A].

Podés pasar cualquiera de los siguientes valores:

- Una cadena de ruta o `path`[3.5.22] desde donde cargar un archivo de sintaxis.
- Bytes sin procesar a partir de los cuales se debe decodificar la sintaxis.
- Un array en el que cada elemento es una de las opciones anteriores.

Para una lista de las sintaxis incorporadas, mirá la documentación del parámetro `lang`[3.7.6/4.3].

````
#set raw(syntaxes: "SExpressions.sublime-syntax")

```sexp
(defun factorial (x)
  (if (zerop x)
    ; with a comment
    1
    (* x (factorial (- x 1)))))
```
````

[A] https://www.sublimetext.com/docs/syntax.html

```
(defun factorial (x)
  (if (zerop x)
    ; with a comment
    1
    (* x (factorial (- x 1)))))
```

### /4.6 theme `none` or `auto` or `str` or `path` or `bytes` *Configurable con set* Por defecto: `auto`

El tema que se usa para el resaltado de sintaxis. Los temas deben estar en el formato de archivo `tmTheme`[A].

Podés pasar cualquiera de los siguientes valores:

- `none`: Desactiva el resaltado de sintaxis.
- `auto`: Resalta con el tema por defecto de Typst.
- Una cadena de ruta o `path`[3.5.22] desde donde cargar un archivo de tema.
- Bytes sin procesar a partir de los cuales se debe decodificar el tema.

Aplicar un tema solo afecta el color del texto resaltado específicamente. No tiene en cuenta las propiedades de primer plano y de fondo del tema, de modo que conservás el control sobre el color del texto raw. Podés aplicar vos mismo el color de primer plano con la función `text`[3.7.12] y el fondo con un bloque relleno[3.10.4/2.4]. También podrías usar la función `xml`[3.13.7] para extraer estas propiedades del tema.

```
#set raw(theme: "halcyon.tmTheme")
#show raw: it => block(
  fill: rgb("#1d2433"),
  inset: 8pt,
  radius: 5pt,
  text(fill: rgb("#a2aabc"), it)
)

```typ
= Chapter 1
#let hi = "Hello World"
```
```

```
= Chapter 1
#let hi = "Hello World"
```

### /4.7 tab-size `int` *Configurable con set* Por defecto: `2`

El tamaño de una tabulación en espacios. Una tabulación se reemplaza por la cantidad de espacios necesaria para alinearse con el siguiente múltiplo del tamaño.

```
#set raw(tab-size: 8)
```tsv
Year  Month Day
2000  2 3
2001  2 1
2002  3 10
```
```

```
Year    Month   Day
2000    2       3
2001    2       1
2002    3       10
```

## /5 Funciones

### /5.1 line Elemento Desde: 0.9.0

Una línea resaltada de texto raw.

Es un elemento auxiliar que sintetizan los elementos `raw`[3.7.6].

Te permite acceder a varias propiedades de la línea, como el número de línea, el texto raw sin resaltar, el texto resaltado y si es la primera o la última línea del bloque raw.

[A] https://www.sublimetext.com/docs/color_schemes_tmtheme.html

```
raw.line(
  int,
  int,
  str,
  content,
) → content
```

/5.1.1 **number** `int` *Obligatorio* *Posicional*

El número de línea de la línea raw dentro del bloque raw, empezando en 1.

/5.1.2 **count** `int` *Obligatorio* *Posicional*

La cantidad total de líneas del bloque raw.

/5.1.3 **text** `str` *Obligatorio* *Posicional*

La línea de texto raw.

/5.1.4 **body** `content` *Obligatorio* *Posicional*

El texto raw resaltado.

## 3.7.7 `smallcaps` Elemento Desde: siempre

Muestra texto en versalitas.

### /1 Ejemplo

```
Hello \
#smallcaps[Hello]
```

Hello
HELLO

### /2 Fuentes con versalitas

Por defecto, esto usa las características OpenType `smcp` y `c2sc` de la fuente. No todas las fuentes admiten estas características. A veces, las versalitas forman parte de una fuente dedicada. Este es, por ejemplo, el caso de la familia de fuentes *Latin Modern*. En esos casos, podés usar una regla show-set para personalizar el aspecto del texto en versalitas:

```
#show smallcaps: set text(font: "Latin Modern Roman Caps")
```

En el futuro, esta función admitirá sintetizar versalitas a partir de letras normales, pero todavía no está implementado.

### /3 Títulos en versalitas

Podés usar una regla show[3.2.2] para aplicar formato de versalitas a todos tus títulos. En el siguiente ejemplo, además alineamos al centro nuestros títulos y desactivamos la fuente en negrita estándar.

```
#set par(justify: true)
#set heading(numbering: "I.")

#show heading: smallcaps
#show heading: set align(center)
#show heading: set text(
  weight: "regular"
)

= Introduction
#lorem(40)
```

I. INTRODUCTION

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere.

### /4 Parámetros

```
smallcaps(
  all: bool,
  content,
) → content
```

#### /4.1 `all` `bool` *Configurable con set* Por defecto: `false`

Si también se convierten las letras mayúsculas en versalitas.

A menos que una regla show lo reemplace, esto habilita la característica OpenType `c2sc`.

```
#smallcaps(all: true)[UNICEF] is an
agency of #smallcaps(all: true)[UN].
```

UNICEF is an agency of UN.

/4.2 **body** `content` *Obligatorio* *Posicional*

El contenido que se muestra en versalitas.

## 3.7.8 `smartquote` Elemento Desde: siempre

Una comilla sensible al idioma que reacciona a su contexto.

Se convierte automáticamente en una comilla de apertura o de cierre apropiada según el idioma del texto[3.7.12/2.16] activo.

### /1 Ejemplo

```
"This is in quotes."

#set text(lang: "de")
"Das ist in Anführungszeichen."

#set text(lang: "fr")
"C'est entre guillemets."
```

“This is in quotes.”

„Das ist in Anführungszeichen.“

« C’est entre guillemets. »

### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: los caracteres de comilla normales (' y ") . Typst vuelve inteligentes tus comillas automáticamente.

### /3 Parámetros

```
smartquote(
  double: bool,
  enabled: bool,
  alternative: bool,
  quotes: auto str array dictionary,
) → content
```

#### /3.1 `double` `bool` *Configurable con set* Por defecto: `true`

Si debe ser una comilla doble.

#### /3.2 `enabled` `bool` *Configurable con set* Por defecto: `true`

Si las comillas inteligentes están habilitadas.

Para desactivar la inteligencia en una sola comilla, también podés escaparla con una barra invertida.

```
#set smartquote(enabled: false)

These are "dumb" quotes.
```

These are "dumb" quotes.

#### /3.3 `alternative` `bool` *Configurable con set* Por defecto: `false`

Si se usan comillas alternativas.

No hace nada para los idiomas que no tienen comillas alternativas o si se establecieron comillas explícitas.

```
#set text(lang: "de")
#set smartquote(alternative: true)

"Das ist in anderen Anführungszeichen."
```

»Das ist in anderen Anführungszeichen.«

#### /3.4 `quotes` `auto` or `str` or `array` or `dictionary` *Configurable con set* Por defecto: `auto`

Las comillas que se usan.

– Cuando se establece en `auto`, se usarán las comillas simples apropiadas para el idioma del texto[3.7.12/2.16]. Este es el valor por defecto.

- Se pueden pasar comillas personalizadas como una cadena, un array o un diccionario de cualquiera de ellos
  - cadena de texto[3.5.28]: una cadena de dos caracteres que contiene las comillas dobles de apertura y de cierre (acá, los caracteres se refieren a grupos de grafemas de Unicode)
  - array[3.5.2]: un array que contiene las comillas dobles de apertura y de cierre
  - diccionario[3.5.12]: un diccionario que contiene las comillas dobles y simples, cada una especificada como `auto`, cadena o array

```
#set text(lang: "de")
'Das sind normale Anführungszeichen.'

#set smartquote(quotes: "()")
"Das sind eigene Anführungszeichen."

#set smartquote(quotes: (single: ("[[", "]]"),  double: auto))
'Das sind eigene Anführungszeichen.'
```

‚Das sind normale Anführungszeichen.‘

(Das sind eigene Anführungszeichen.)

[[Das sind eigene Anführungszeichen.]]

## 3.7.9 `strike` Elemento Desde: siempre

Tacha texto.

### /1 Ejemplo

```
This is #strike[not] relevant.
```

This is ~~not~~ relevant.

### /2 Parámetros

```
strike(
  stroke: auto | length | color | gradient | stroke | tiling | dictionary,
  offset: auto | length,
  extent: length,
  background: bool,
  content,
) → content
```

#### /2.1 `stroke` `auto` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary`

*Configurable con set* Por defecto: `auto`

Cómo trazar[3.11.12] la línea.

Si se establece en `auto`, toma el color del texto y un grosor definido en la fuente actual.

*Nota:* Por favor, no uses esto para censurar de verdad, ya que igualmente se puede copiar y pegar el texto.

```
This is #strike(stroke: 1.5pt + red)[very stricken through]. \
This is #strike(stroke: 10pt)[redacted].
```

This is ~~very stricken through~~.
This is [redacted].

#### /2.2 `offset` `auto` or `length` *Configurable con set* Por defecto: `auto`

La posición de la línea relativa a la línea base. Se lee de las tablas de la fuente si es `auto`.

Es útil si no estás conforme con el desplazamiento que ofrece tu fuente.

```
#set text(font: "Inria Serif")
This is #strike(offset: auto)[low-ish]. \
This is #strike(offset: -3.5pt)[on-top].
```

This is ~~low-ish~~.
This is ~~on-top~~.

#### /2.3 `extent` `length` *Configurable con set* Por defecto: `0pt`

La cantidad en que se extiende la línea más allá del contenido (o hacia adentro, si es negativa).

```
This #strike(extent: -2pt)[skips] parts of the word.
This #strike(extent: 2pt)[extends] beyond the word.
```

This ~~skips~~ parts of the word. This ~~extends~~ beyond the word.

/2.4 **background** `bool` *Configurable con set* Por defecto: `false`

Si la línea se ubica detrás del contenido.

```
#set strike(stroke: red + 2pt)
#strike(background: true)[This is behind.] \
#strike(background: false)[This is in front.]
```

~~This is behind.~~
~~This is in front.~~

/2.5 **body** `content` *Obligatorio* *Posicional*

El contenido que se tacha.

### 3.7.10 sub Elemento Desde: siempre

Renderiza texto como subíndice.

El texto se renderiza más pequeño y su línea base se baja.

#### /1 Ejemplo

```
Revenue#sub[yearly]
```

Revenue$_{yearly}$

#### /2 Parámetros

```
sub(
  typographic: bool ,
  baseline: auto length ,
  size: auto length ,
  content ,
) → content
```

##### /2.1 typographic bool *Configurable con set* Por defecto: true

Si se usan los glifos de subíndice de la fuente, si están disponibles.

Idealmente, los glifos de subíndice los proporciona la fuente (mediante la característica OpenType `subs`). De lo contrario, Typst puede sintetizar subíndices bajando y reduciendo los glifos normales.

Cuando se establece en `false`, se usarán glifos sintetizados sin importar si la fuente ofrece glifos de subíndice dedicados. Cuando es `true`, igualmente se pueden usar glifos sintetizados en caso de que la fuente no proporcione los glifos de subíndice necesarios.

```
N#sub(typographic: true)[1]
N#sub(typographic: false)[1]
```

$N_1$ $N_1$

##### /2.2 baseline auto or length *Configurable con set* Por defecto: auto

El desplazamiento hacia abajo de la línea base para los subíndices sintetizados.

Esto solo se aplica a los subíndices sintetizados. En otras palabras, no tiene efecto si `typographic` es `true` y la fuente proporciona los glifos de subíndice necesarios.

Si se establece en `auto`, la línea base se desplaza según las métricas que proporciona la fuente, con un valor de respaldo de `0.2em` en caso de que la fuente no defina las métricas necesarias.

Al usar varias fuentes, puede ser necesario establecer explícitamente `baseline` y `size`[3.7.10/2.3]. Mirá `super`[3.7.11/2.2] para ver un ejemplo.

##### /2.3 size auto or length *Configurable con set* Por defecto: auto

El tamaño de fuente de los subíndices sintetizados.

Esto solo se aplica a los subíndices sintetizados. En otras palabras, no tiene efecto si `typographic` es `true` y la fuente proporciona los glifos de subíndice necesarios.

Si se establece en `auto`, el tamaño se escala según las métricas que proporciona la fuente, con un valor de respaldo de `0.6em` en caso de que la fuente no defina las métricas necesarias.

##### /2.4 body content *Obligatorio* *Posicional*

El texto que se muestra como subíndice.

## 3.7.11 `super` Elemento Desde: siempre

Renderiza texto como superíndice.

El texto se renderiza más pequeño y su línea base se sube.

### /1 Ejemplo

```
1#super[st] try!
```

1st try!

### /2 Parámetros

```
super(
  typographic: bool,
  baseline: auto length,
  size: auto length,
  content,
) → content
```

#### /2.1 `typographic` `bool` *Configurable con set* Por defecto: `true`

Si se usan los glifos de superíndice de la fuente, si están disponibles.

Idealmente, los glifos de superíndice los proporciona la fuente (mediante la característica OpenType `sups`). De lo contrario, Typst puede sintetizar superíndices subiendo y reduciendo los glifos normales.

Cuando se establece en `false`, se usarán glifos sintetizados sin importar si la fuente ofrece glifos de superíndice dedicados. Cuando es `true`, igualmente se pueden usar glifos sintetizados en caso de que la fuente no proporcione los glifos de superíndice necesarios.

```
N#super(typographic: true)[1]
N#super(typographic: false)[1]
```

N1 N1

#### /2.2 `baseline` `auto` or `length` *Configurable con set* Por defecto: `auto`

El desplazamiento hacia abajo de la línea base para los superíndices sintetizados.

Esto solo se aplica a los superíndices sintetizados. En otras palabras, no tiene efecto si `typographic` es `true` y la fuente proporciona los glifos de superíndice necesarios.

Si se establece en `auto`, la línea base se desplaza según las métricas que proporciona la fuente, con un valor de respaldo de `-0.5em` en caso de que la fuente no defina las métricas necesarias.

Tené en cuenta que, como el desplazamiento de la línea base se aplica hacia abajo, vas a tener que proporcionar un valor negativo para que el contenido aparezca elevado por encima de la línea base normal.

A veces es necesario establecer explícitamente `baseline` y `size`[3.7.11/2.3]. En el siguiente ejemplo, el texto en superíndice usa varias fuentes con métricas incompatibles. Para evitar desalineaciones, reemplazamos las métricas de todas las fuentes.

```
#let tz(timezone) = text(
  font: "Roboto",
  smallcaps(timezone, all: true),
)

/ Per-font metrics:
  14:00#super[#tz[UTC], not #tz[CET]]
```

**Per-font metrics** 14:00UTC, not CET

**Unified metrics** 14:00UTC, not CET

```
#set super(
  baseline: -0.4em,
  size: 0.6em,
  typographic: false,
)
/ Unified metrics:
  14:00#super[#tz[UTC], not #tz[CET]]
```

/2.3 **size** `auto` or `length` *Configurable con set* Por defecto: `auto`

El tamaño de fuente de los superíndices sintetizados.

Esto solo se aplica a los superíndices sintetizados. En otras palabras, no tiene efecto si `typographic` es `true` y la fuente proporciona los glifos de superíndice necesarios.

Si se establece en `auto`, el tamaño se escala según las métricas que proporciona la fuente, con un valor de respaldo de `0.6em` en caso de que la fuente no defina las métricas necesarias.

/2.4 **body** `content` *Obligatorio* *Posicional*

El texto que se muestra como superíndice.

### 3.7.12 `text` Elemento Desde: siempre

Personaliza el aspecto y el diseño del texto de diversas maneras.

Esta función se usa con frecuencia, tanto con reglas set como directamente. Si bien la regla set suele ser la opción más simple, llamar directamente a la función `text` puede ser útil al pasar texto como argumento a otra función.

#### /1 Ejemplo

```
#set text(18pt)
With a set rule.

#emph(text(blue)[
  With a function call.
])
```

With a set rule.

*With a function call.*

#### /2 Parámetros

```
text(
  font: str array dictionary,
  fallback: bool,
  style: str,
  weight: int str,
  stretch: ratio,
  size: length,
  fill: color gradient tiling,
  stroke: none length color gradient stroke tiling dictionary,
  tracking: length,
  spacing: relative,
  cjk-latin-spacing: none auto,
  baseline: length,
  overhang: bool,
  top-edge: length str,
  bottom-edge: length str,
  lang: str,
  region: none str,
  script: auto str,
  dir: auto direction,
  hyphenate: auto bool,
  costs: dictionary,
  kerning: bool,
  alternates: bool int,
  stylistic-set: none int array,
  ligatures: bool,
  discretionary-ligatures: bool,
  historical-ligatures: bool,
  number-type: auto str,
  number-width: auto str,
  slashed-zero: bool,
  fractions: bool,
  features: array dictionary,
  variations: dictionary,
  content,
```

```
  str,
) → content
```

/2.1 **font** `str` or `array` or `dictionary` *Configurable con set* Por defecto: `"libertinus serif"`

Un descriptor de familia de fuentes o una lista de prioridad de descriptores de familias de fuentes.

Un descriptor de familia de fuentes puede ser una cadena de texto simple que representa el nombre de la familia o un diccionario con las siguientes claves:

- `name` (obligatoria): El nombre de la familia de fuentes.
- `covers` (opcional): Define los puntos de código Unicode para los que se debe usar la familia. Puede ser:
  - Un conjunto de cobertura predefinido:
    - `"latin-in-cjk"` cubre todos los puntos de código excepto los que existen en fuentes latinas, pero que preferentemente deberían tomarse de fuentes CJK.
  - Una expresión regular[3.5.24] que define exactamente qué puntos de código se deben cubrir. Acepta solo el subconjunto de expresiones regulares que consisten en exactamente un punto, una letra o una clase de caracteres.

Al procesar texto, Typst prueba todas las familias de fuentes especificadas en orden hasta encontrar una fuente que tenga los glifos necesarios. En el siguiente ejemplo, se prefiere la fuente `Inria Serif`, pero como no contiene glifos árabes, el texto árabe usa en cambio `Noto Sans Arabic`.

Typst busca unificar distintas fuentes de la misma familia bajo un único nombre de familia. Con ese fin, recorta automáticamente los sufijos de estilo comunes, como "Bold" o "Condensed", de los nombres de familias de fuentes. En lugar de seleccionarlos mediante el nombre, accedé a ellos con los mecanismos incorporados de Typst (como los parámetros `weight`[3.7.12/2.4] y `stretch`[3.7.12/2.5]). De manera similar, al usar una fuente variable con Typst, se deben omitir los sufijos "Variable", "Var" y "VF", ya que Typst los recorta para unificar las fuentes estáticas y variables en una única familia.

Entre fuentes de la misma familia, Typst elige la que más se aproxima al `style`[3.7.12/2.3], `weight`[3.7.12/2.4] y `stretch`[3.7.12/2.5] del texto configurados. Si tanto una fuente estática como una variable admiten una configuración específica, se prefiere la fuente variable.

La colección de fuentes disponibles difiere según la plataforma:

- En la app web, podés ver la lista de fuentes disponibles haciendo clic en el botón "Ag". Podés proporcionar fuentes adicionales subiendo archivos `.ttf` u `.otf` a tu proyecto. Se descubrirán automáticamente. La prioridad es: fuentes del proyecto > fuentes del servidor.
- De forma local, Typst usa las fuentes del sistema instaladas o las fuentes incorporadas en la CLI, que son `Libertinus Serif`, `New Computer Modern`, `New Computer Modern Math` y `DejaVu Sans Mono`. Además, podés usar el argumento `--font-path` o la variable de entorno `TYPST_FONT_PATHS` para agregar directorios que se deben escanear en busca de fuentes. La prioridad es: `--font-path` > fuentes del sistema > fuentes incorporadas. Ejecutá `typst fonts` para ver las fuentes que Typst descubrió en tu sistema. Tené en cuenta que podés pasar el parámetro `--ignore-system-fonts` a la CLI para asegurarte de que Typst no busque fuentes del sistema.

```
#set text(font: "PT Sans")
This is sans-serif.

#set text(font: (
  "Inria Serif",
  "Noto Sans Arabic",
))

This is Latin. \
هذا عربي.

// Change font only for numbers.
#set text(font: (
  (name: "PT Sans", covers: regex("[0-9]")),
```

```
  "Libertinus Serif"
))

The number 123.

// Mix Latin and CJK fonts.
#set text(font: (
  (name: "Inria Serif", covers: "latin-in-cjk"),
  "Noto Serif CJK SC"
))
分别设置“中文”和English字体
```

This is sans-serif.

This is Latin.
هذا عربي.

The number 123.

分别设置“中文”和 English 字体

### /2.2 **fallback** `bool` *Configurable con set* Por defecto: `true`

Si se permite el respaldo de fuente como último recurso cuando la lista de fuentes primarias no contiene ninguna coincidencia. Esto permite que Typst busque entre todas las fuentes disponibles la más similar que tenga los glifos necesarios.

*Nota:* Actualmente no hay advertencias cuando el respaldo está desactivado y no se encuentran glifos. En cambio, tu texto aparece en forma de “tofus”: pequeñas cajas que indican la falta de un glifo apropiado. En el futuro, vas a poder indicarle a Typst que emita advertencias para que sepas que algo anda mal.

```
#set text(font: "Inria Serif")
هذا عربي

#set text(fallback: false)
هذا عربي
```

هذا عربي

□□□□ □□□

### /2.3 **style** `str` *Configurable con set* Por defecto: `"normal"`

El estilo de fuente deseado.

Cuando se solicita un estilo cursivo y solo hay uno oblicuo disponible, se usa este. De manera similar, a la inversa, un estilo cursivo puede reemplazar a uno oblicuo. Cuando no hay disponible ni un estilo cursivo ni uno oblicuo, Typst selecciona el estilo normal. Como la mayoría de las fuentes solo están disponibles en estilo cursivo u oblicuo, rara vez se observa la diferencia entre el estilo cursivo y el oblicuo.

Cuando se usa con una fuente variable adecuada, Typst configurará automáticamente la variación de fuente[3.7.12/2.33] `ital` (para un estilo cursivo) o `slnt` (para un estilo oblicuo) según esta propiedad.

*Nota:* Si querés enfatizar tu texto, deberías hacerlo con la función emph[3.6.6]. Esto facilita adaptar el estilo más adelante si cambiás de idea sobre cómo señalar el énfasis.

```
#text(font: "Libertinus Serif", style: "italic")[Italic]
#text(font: "DejaVu Sans", style: "oblique")[Oblique]
```

*Italic* *Oblique*

| Variante | Detalles |
|---|---|
| `"normal"` | El estilo predeterminado, normalmente vertical. |
| `"italic"` | Un estilo cursivo con letras de trazo personalizado. |
| `"oblique"` | Simplemente una versión inclinada del estilo normal. |

### /2.4 `weight` `int` or `str` *Configurable con set* Por defecto: `"regular"`

El grosor deseado de los glifos de la fuente. Acepta un entero entre `100` y `900` o uno de los nombres de peso predefinidos. Cuando el peso deseado no está disponible, Typst selecciona de la familia la fuente más cercana en peso.

Cuando se usa con una fuente variable adecuada, Typst configurará automáticamente la variación de fuente[3.7.12/2.33] `wght` según esta propiedad.

*Nota:* Si querés enfatizar fuertemente tu texto, deberías hacerlo con la función strong[3.6.18]. Esto facilita adaptar el estilo más adelante si cambiás de idea sobre cómo señalar el énfasis fuerte.

```
#set text(font: "IBM Plex Sans")

#text(weight: "light")[Light] \
#text(weight: "regular")[Regular] \
#text(weight: "medium")[Medium] \
#text(weight: 500)[Medium] \
#text(weight: "bold")[Bold]
```

Light
Regular
**Medium**
**Medium**
**Bold**

| Variante | Detalles |
|---|---|
| `"thin"` | Grosor fino (100). |
| `"extralight"` | Grosor extraligero (200). |
| `"light"` | Grosor ligero (300). |
| `"regular"` | Grosor normal (400). |
| `"medium"` | Grosor medio (500). |
| `"semibold"` | Grosor seminegrita (600). |
| `"bold"` | Grosor negrita (700). |
| `"extrabold"` | Grosor extranegrita (800). |
| `"black"` | Grosor negro (900). |

### /2.5 `stretch` `ratio` *Configurable con set* Por defecto: `100%`

El ancho deseado de los glifos. Acepta una proporción entre `50%` y `200%`. Cuando el ancho deseado no está disponible, Typst selecciona de la familia la fuente más cercana en estiramiento. Esto solo estirará el texto si hay una versión condensada o expandida de la fuente disponible.

Cuando se usa con una fuente variable adecuada, Typst configurará automáticamente la variación de fuente[3.7.12/2.33] `wdth` según esta propiedad.

Si querés ajustar la cantidad de espacio entre caracteres en lugar de estirar los propios glifos, usá en cambio la propiedad `tracking`[3.7.12/2.9].

```
#set text(font: "IBM Plex Sans")
#text(stretch: 75%)[Condensed] \
#text(stretch: 100%)[Normal]
```

Condensed
Normal

### /2.6 `size` `length` *Configurable con set* Por defecto: `11pt`

El tamaño de los glifos. Este valor forma la base de la unidad `em`: `1em` equivale al tamaño de la fuente.

También podés dar el propio tamaño de fuente en unidades `em`. En ese caso, es relativo al tamaño de fuente anterior.

Cuando se usa con una fuente variable adecuada, Typst configurará automáticamente la variación de fuente[3.7.12/2.33] `opsz` (tamaño óptico) según esta propiedad, optimizando la legibilidad para el tamaño específico.

```
#set text(size: 20pt)
very #text(1.5em)[big] text
```

very big text

/2.7 **fill** `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `luma(0%)`

El relleno de los glifos.

```
#set text(fill: red)
This text is red.
```

This text is red.

/2.8 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary`
*Configurable con set* Por defecto: none

Cómo trazar el texto.

```
#text(stroke: 0.5pt + red)[Stroked]
```

Stroked

/2.9 **tracking** `length` *Configurable con set* Por defecto: `0pt`

La cantidad de espacio que se debe agregar entre caracteres.

```
#set text(tracking: 1.5pt)
Distant text.
```

Distant text.

/2.10 **spacing** `relative` *Configurable con set* Por defecto: `100% + 0pt`

La cantidad de espacio entre palabras.

Se puede dar como una longitud absoluta, pero también relativa al ancho del carácter de espacio de la fuente.

Si querés ajustar la cantidad de espacio entre caracteres en lugar de entre palabras, usá en cambio la propiedad `tracking`[3.7.12/2.9].

```
#set text(spacing: 200%)
Text with distant words.
```

Text with distant words.

/2.11 **cjk-latin-spacing** `none` or `auto` *Configurable con set* Por defecto: auto

Si se inserta automáticamente espaciado entre caracteres CJK y latinos.

```
#set text(cjk-latin-spacing: auto)
第4章介绍了基本的API。

#set text(cjk-latin-spacing: none)
第4章介绍了基本的API。
```

第 4 章介绍了基本的 API.

第4章介绍了基本的API.

/2.12 **baseline** `length` *Configurable con set* Por defecto: `0pt`

Una cantidad por la que se desplaza la línea base del texto.

```
A #text(baseline: 3pt)[lowered]
word.
```

A lowered word.

/2.13 **overhang** `bool` *Configurable con set* Por defecto: `true`

Si ciertos glifos pueden sobresalir hacia el margen en el texto justificado. Esto puede hacer que la justificación sea visualmente más agradable.

```
#set page(width: 220pt)

#set par(justify: true)
This justified text has a hyphen in
the paragraph's second line. Hanging
the hyphen slightly into the margin
results in a clearer paragraph edge.

#set text(overhang: false)
This justified text has a hyphen in
the paragraph's second line. Hanging
the hyphen slightly into the margin
results in a clearer paragraph edge.
```

This justified text has a hyphen in the paragraph's second line. Hanging the hyphen slightly into the margin results in a clearer paragraph edge.

This justified text has a hyphen in the paragraph's second line. Hanging the hyphen slightly into the margin results in a clearer paragraph edge.

/2.14 **top-edge** `length` or `str` *Configurable con set* Por defecto: `"cap-height"`

El extremo superior del marco conceptual alrededor del texto que se usa para el diseño y el posicionamiento. Esto afecta el tamaño de los contenedores que contienen texto.

```
#set rect(inset: 0pt)
#set text(size: 20pt)

#set text(top-edge: "ascender")
#rect(fill: aqua)[Typst]

#set text(top-edge: "cap-height")
#rect(fill: aqua)[Typst]
```

Typst

Typst

| Variante | Detalles |
|---|---|
| `"ascender"` | El ascendente de la fuente, que normalmente supera la altura de todos los glifos. |
| `"cap-height"` | La altura aproximada de las letras mayúsculas. |
| `"x-height"` | La altura aproximada de las letras minúsculas sin ascendentes. |
| `"baseline"` | La línea base sobre la que se apoyan las letras. |
| `"bounds"` | El borde superior del cuadro delimitador del glifo. |

/2.15 **bottom-edge** `length` or `str` *Configurable con set* Por defecto: `"baseline"`

El extremo inferior del marco conceptual alrededor del texto que se usa para el diseño y el posicionamiento. Esto afecta el tamaño de los contenedores que contienen texto.

```
#set rect(inset: 0pt)
#set text(size: 20pt)

#set text(bottom-edge: "baseline")
#rect(fill: aqua)[Typst]

#set text(bottom-edge: "descender")
#rect(fill: aqua)[Typst]
```

Typst

Typst

| Variante | Detalles |
|---|---|
| `"baseline"` | La línea base sobre la que se apoyan las letras. |
| `"descender"` | El descendente de la fuente, que normalmente supera la profundidad de todos los glifos. |
| `"bounds"` | El borde inferior del cuadro delimitador del glifo. |

/2.16 **lang** `str` *Configurable con set* Por defecto: `"en"`

Un código de idioma ISO 639-1/2/3.[A]

Establecer el idioma correcto afecta varias partes de Typst:

- El proceso de tratamiento del texto puede tomar decisiones más informadas.
- La separación en sílabas usará los patrones correctos para el idioma.
- Las comillas inteligentes[3.7.8] se convierten en las comillas correctas para el idioma.
- Y todas las demás cosas que dependen del idioma.

Elegir el idioma correcto es importante para la accesibilidad. Por ejemplo, los lectores de pantalla lo usarán para elegir una voz que coincida con el idioma del texto. Si tu documento está en un idioma distinto del inglés (el valor por defecto), deberías establecer el idioma del texto al comienzo de tu documento, antes de cualquier otro contenido. Podés, por ejemplo, ponerlo justo después de la regla `#set document(/* ... */)` que establece el título de tu documento[3.6.5/3.3].

Si tu documento contiene pasajes en un idioma distinto del idioma principal, deberías cambiar localmente el idioma del texto solo para esas partes, ya sea con una regla set acotada a un bloque[3.3.2] o con una llamada directa a la función text, como `#text(lang: "de")[...]`.

Si hay varios códigos disponibles para tu idioma, deberías preferir el código de dos letras (ISO 639-1) en lugar de los códigos de tres letras (ISO 639-2/3). Cuando tengas que usar un código de tres letras y tu idioma difiera entre ISO 639-2 e ISO 639-3, usá ISO 639-2 para PDF 1.7 (el valor por defecto de Typst para la exportación a PDF) y anteriores, e ISO 639-3 para PDF 2.0 y la exportación a HTML.

El código de idioma no distingue mayúsculas de minúsculas y se pasará a minúsculas al acceder a él a través del contexto[3.4].

Example · Establecer el idioma del texto en alemán

```
#set text(lang: "de")
#outline()

= Einleitung
In diesem Dokument, ...
```

**Inhaltsverzeichnis**

Einleitung ..... 1

**Einleitung**

In diesem Dokument, ...

/2.17 **region** `none` or `str` *Configurable con set* Por defecto: `none`

Un código de región ISO 3166-1 alfa-2.[B]

Esto permite que el proceso de tratamiento del texto tome decisiones más informadas.

[A] https://en.wikipedia.org/wiki/ISO_639
[B] https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

El código de región no distingue mayúsculas de minúsculas y se pasará a mayúsculas al acceder a él a través del contexto[3.4].

### /2.18 `script` `auto` or `str` *Configurable con set* Por defecto: `auto`

La escritura OpenType.

La combinación de `lang` y `script` determina cómo se implementan las características de fuente, como la sustitución de glifos. Con frecuencia, el valor es un identificador de escritura ISO 15924 modificado (todo en minúsculas), y la escritura `math` se usa para las características apropiadas para símbolos matemáticos.

Cuando se establece en `auto`, el valor por defecto y recomendado, se elige una escritura apropiada para cada bloque de caracteres que comparten una propiedad de escritura Unicode común.

```
#set text(
  font: "IBM Plex Sans",
  size: 20pt,
)

#let scedilla = [Ş]
#scedilla // S with a cedilla

#set text(lang: "ro", script: "latn")
#scedilla // S with a subscript comma

#set text(lang: "ro", script: "grek")
#scedilla // S with a cedilla
```

Ş

Ș

Ş

### /2.19 `dir` `auto` or `direction` *Configurable con set* Por defecto: `auto`

La dirección dominante del texto y los objetos en línea. Los valores posibles son:

- `auto`: Infiere automáticamente la dirección a partir de la propiedad `lang`.
- `ltr`: Dispone el texto de izquierda a derecha.
- `rtl`: Dispone el texto de derecha a izquierda.

Al escribir en escrituras de derecha a izquierda, como el árabe o el hebreo, deberías establecer el idioma del texto[3.7.12/2.16] o la dirección. Si bien las series individuales de texto se disponen automáticamente en la dirección correcta, establecer la dirección dominante le da al algoritmo de reordenamiento bidireccional la información necesaria para ubicar correctamente la puntuación y los objetos en línea. Además, establecer la dirección afecta los valores de alineación `start` y `end`, que equivalen a `left` y `right` en texto `ltr` y al revés en texto `rtl`.

Si establecés esto en `rtl` y encontrás errores o una salida que de algún modo se ve mal, por favor contactanos a través del foro[A], del servidor de Discord[B] o de nuestro formulario de contacto[C].

```
#set text(dir: rtl)
هذا عربي.
```

هذا عربي.

### /2.20 `hyphenate` `auto` or `bool` *Configurable con set* Por defecto: `auto`

Si se separa el texto en sílabas para mejorar los saltos de línea. Cuando es `auto`, el texto se separará en sílabas si y solo si la justificación está habilitada.

Establecer el idioma del texto[3.7.12/2.16] asegura que se usen los patrones de separación en sílabas correctos.

[A] https://forum.typst.app/
[B] https://discord.gg/2uDybryKPe
[C] https://typst.app/contact

```
#set page(width: 200pt)

#set par(justify: true)
This text illustrates how
enabling hyphenation can
improve justification.

#set text(hyphenate: false)
This text illustrates how
enabling hyphenation can
improve justification.
```

This text illustrates how enabling hyphenation can improve justification.

This text illustrates how enabling hyphenation can improve justification.

/2.21 **costs** `dictionary` *Configurable con set*

Por defecto: (
  hyphenation: 100%,
  runt: 100%,
  widow: 100%,
  orphan: 100%,
)

El "costo" de varias decisiones al disponer el texto. Un costo más alto significa que el motor de diseño tomará esa decisión con menos frecuencia. Los costos se especifican como una proporción del costo por defecto, así que `50%` hará que el diseño del texto sea el doble de propenso a tomar una decisión dada, mientras que `200%` lo hará la mitad de propenso.

Actualmente se pueden personalizar los siguientes costos:
- `hyphenation`: dividir una palabra entre varias líneas
- `runt`: terminar un párrafo con una línea de una sola palabra
- `widow`: dejar una sola línea de un párrafo en la página siguiente
- `orphan`: dejar una sola línea de un párrafo en la página anterior

La separación en sílabas generalmente se evita poniendo toda la palabra en la línea siguiente, por lo que un costo de separación más alto puede dar como resultado un espaciado de justificación incómodo. Nota: Los costos de separación en sílabas solo se aplicarán cuando los `linebreaks`[3.6.14/4.5] estén establecidos en "optimized". (Por ejemplo, por defecto está implícito en `justify`[3.6.14/4.3].)

Las líneas huérfanas de una palabra (runts) se evitan colocando más o menos palabras en las líneas anteriores, por lo que un costo de runt más alto puede dar como resultado un espaciado de justificación más incómodo.

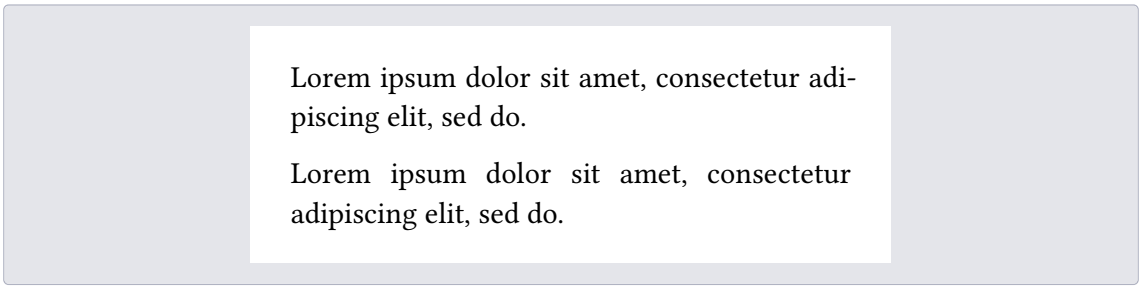
El diseño del texto evita las viudas y las huérfanas por defecto, porque las guías de estilo generalmente las desaconsejan. Sin embargo, en algunos contextos se permiten, porque el método de prevención, que mueve una línea a la página siguiente, puede producir una cantidad despareja de líneas entre las páginas. Los costos `widow` y `orphan` permiten desactivar estas modificaciones. (Actualmente, `0%` permite viudas/huérfanas; cualquier otro valor, incluido el valor por defecto de `100%`, las evita. Está previsto para el futuro una especificación de costos más matizada para estas modificaciones.)

```
#set text(hyphenate: true, size: 11.4pt)
#set par(justify: true)

#lorem(10)

// Set hyphenation to ten times the normal cost.
#set text(costs: (hyphenation: 1000%))

#lorem(10)
```



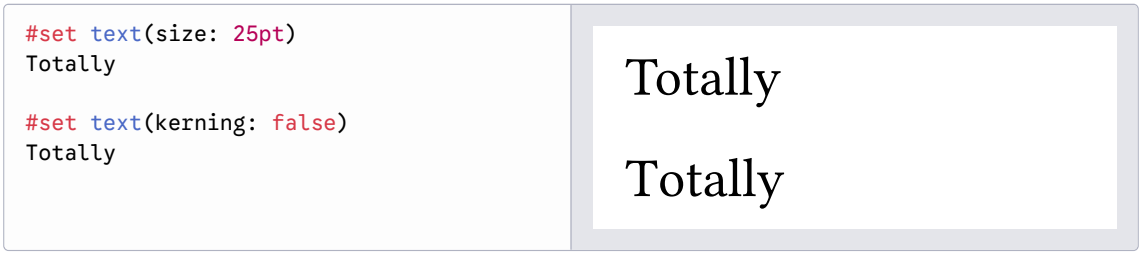

/2.22 **kerning** `bool` *Configurable con set* Por defecto: `true`

Si se aplica el kerning (interletraje).

Cuando está habilitado, ciertos pares de letras se acercan o se alejan entre sí para lograr un resultado visualmente más agradable. El siguiente ejemplo demuestra cómo disminuir el espacio entre la "T" y la "o" da un aspecto más natural. Establecer esto en `false` desactiva el kerning al apagar la característica de fuente OpenType `kern`.

```
#set text(size: 25pt)
Totally

#set text(kerning: false)
Totally
```




/2.23 **alternates** `bool` or `int` *Configurable con set* Por defecto: `0`

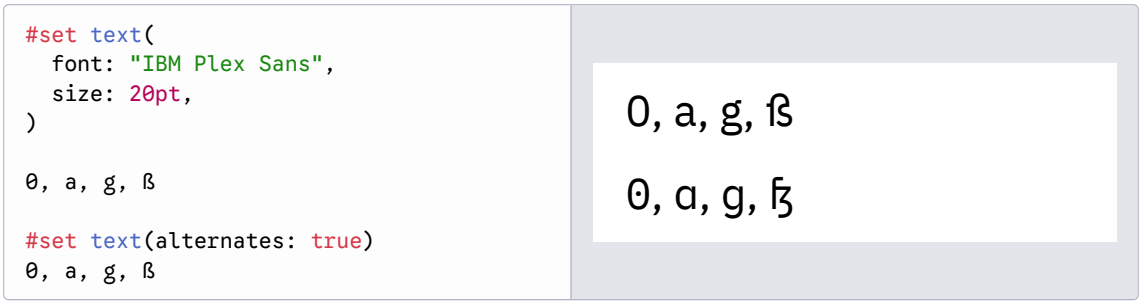
Si se aplican alternativas estilísticas.

A veces las fuentes contienen glifos alternativos para el mismo punto de código. Establecer esto en `true` cambia a ellos habilitando la característica de fuente OpenType `salt`. Se puede usar un entero para elegir entre varias alternativas.

```
#set text(
  font: "IBM Plex Sans",
  size: 20pt,
)

0, a, g, ß

#set text(alternates: true)
0, a, g, ß
```




/2.24 **stylistic-set** `none` or `int` or `array` *Configurable con set* Por defecto: ()

Qué conjuntos estilísticos se aplican. Los diseñadores de fuentes pueden categorizar las formas de glifos alternativos en conjuntos estilísticos. Como este valor es muy específico de cada fuente, necesitás consultar tu fuente para saber qué conjuntos están disponibles.

Se puede establecer en un entero o en un array de enteros, todos los cuales deben estar entre `1` y `20`, y habilita las características OpenType correspondientes, de `ss01` a `ss20`. Establecerlo en `none` desactivará todos los conjuntos estilísticos.

```
#set text(font: "IBM Plex Serif")
ß vs #text(stylistic-set: 5)[ß] \
10 years ago vs #text(stylistic-set: (1, 2, 3))[10 years ago]
```




/2.25 **ligatures** `bool` *Configurable con set* Por defecto: `true`

Si están activas las ligaduras estándar.

Ciertas combinaciones de letras, como "fi", a menudo se muestran como un único glifo fusionado llamado *ligadura*. Establecer esto en `false` desactiva estas ligaduras al apagar las características de fuente OpenType `liga` y `clig`.

```
#set text(size: 20pt)
A fine ligature.

#set text(ligatures: false)
A fine ligature.
```

A fine ligature.

A fine ligature.

Tené en cuenta que algunas fuentes de programación usan otras características de fuente OpenType para implementar "ligaduras", incluida la característica de alternativas contextuales (`calt`), que también está habilitada por defecto. Usá el parámetro general `features`[3.7.12/2.32] para controlar esas características.

/2.26 **discretionary-ligatures** `bool` *Configurable con set* Por defecto: `false`

Si están activas las ligaduras que se deben usar con moderación. Establecer esto en `true` habilita la característica de fuente OpenType `dlig`.

/2.27 **historical-ligatures** `bool` *Configurable con set* Por defecto: `false`

Si están activas las ligaduras históricas. Establecer esto en `true` habilita la característica de fuente OpenType `hlig`.

/2.28 **number-type** `auto` or `str` *Configurable con set* Por defecto: `auto`

Qué tipo de números / cifras seleccionar. Cuando se establece en `auto`, se usan los números por defecto de la fuente.

```
#set text(font: "Noto Sans", 20pt)
#set text(number-type: "lining")
Lining: 0123456789.

#set text(number-type: "old-style")
Old style: 0123456789.
```

Lining: 0123456789.

Old style: 0123456789.

| Variante | Detalles |
|---|---|
| `"lining"` | Números que combinan bien con texto en mayúsculas (la característica de fuente OpenType `lnum`). |
| `"old-style"` | Números que se integran bien en un flujo de texto en mayúsculas y minúsculas (la característica de fuente OpenType `onum`). |

/2.29 **number-width** `auto` or `str` *Configurable con set* Por defecto: `auto`

El ancho de los números / cifras. Cuando se establece en `auto`, se usan los números por defecto de la fuente.

```
#set text(font: "Noto Sans", 20pt)
#set text(number-width: "proportional")
A 12 B 34. \
A 56 B 78.

#set text(number-width: "tabular")
A 12 B 34. \
A 56 B 78.
```

A 12 B 34.
A 56 B 78.

A 12 B 34.
A 56 B 78.

| Variante | Detalles |
|---|---|
| `"proportional"` | Números con anchos propios de cada glifo (la característica de fuente OpenType `pnum`). |
| `"tabular"` | Números de ancho uniforme (la característica de fuente OpenType `tnum`). |

/2.30 **`slashed-zero`** `bool` *Configurable con set* Por defecto: `false`

Si el glifo del cero lleva una barra. Establecer esto en `true` habilita la característica de fuente OpenType `zero`.

```
0, #text(slashed-zero: true)[0]
```

0, 0

/2.31 **`fractions`** `bool` *Configurable con set* Por defecto: `false`

Si se convierten los números en fracciones. Establecer esto en `true` habilita la característica de fuente OpenType `frac`.

No es aconsejable habilitar esta propiedad globalmente, ya que arruinará todas las apariciones de números después de una barra (por ejemplo, en las URL). En cambio, habilitala localmente cuando quieras una fracción.

```
1/2 \
#text(fractions: true)[1/2]
```

1/2
1⁄2

/2.32 **`features`** `array` or `dictionary` *Configurable con set* Por defecto: `(:)`

Características OpenType sin procesar que se aplican.

- Si se da un array de cadenas de texto, establece en `1` las características identificadas por las cadenas.
- Si se da un diccionario que asigna números, establece las características identificadas por las claves en los valores. Esto permite interactuar con características no booleanas como `swsh`.

Example · Dar un array de cadenas de texto

```
// Enable the `frac` feature manually.
#set text(features: ("frac",))
1/2
```

1⁄2

Example · Dar un diccionario que asigna números

```
#set text(font: "Cascadia Code")
=>
// Disable the contextual alternates (`calt`) feature.
#set text(features: (calt: 0))
=>
```

⇒ =>

/2.33 **variables** `dictionary` *Configurable con set* Por defecto: (:)

Variaciones de fuente OpenType sin procesar que se aplican.

Mientras que las fuentes estáticas clásicas requieren un archivo de fuente separado para cada combinación de estilos, las fuentes variables tienen *ejes de variación* a partir de los cuales se pueden instanciar muchos estilos distintos. Los ejes de variación se identifican con cadenas de cuatro letras que distinguen mayúsculas de minúsculas.

Hay algunos ejes de variación muy conocidos, para los que Typst establecerá automáticamente valores adecuados según el `weight`[3.7.12/2.4], el `stretch`[3.7.12/2.5], el `style`[3.7.12/2.3] y el `size`[3.7.12/2.6] del texto. Esto incluye:

- `wght`: Peso (por ejemplo, 400 para regular, 700 para negrita)
- `wdth`: Ancho (porcentaje, por ejemplo, 100 para normal)
- `slnt`: Inclinación (en grados, negativa para inclinado hacia la derecha)
- `ital`: Cursiva (0 para vertical, 1 para cursiva)
- `opsz`: Tamaño óptico (en puntos)

Las fuentes también pueden definir ejes de variación personalizados para lograr efectos visuales arbitrarios. Por ejemplo, el aspecto de una fuente podría volverse más caprichoso cuanto mayor sea el valor de un eje en particular.

Con el parámetro `variations`, podés establecer directamente valores para los ejes que admite la fuente activa. Solo tiene efecto cuando se usa con una fuente variable adecuada que admita los ejes especificados. Podés usar el parámetro tanto para reemplazar los valores establecidos automáticamente para los ejes muy conocidos como para establecer valores para ejes personalizados.

El valor debe ser un diccionario que asigne etiquetas de ejes (cadenas de texto[3.5.28] de cuatro caracteres) a sus valores (números de punto flotante[3.5.15]).

Example · Establecer valores para ejes personalizados

```
#set text(font: "Fraunces", size: 60pt)

#text(variations: (SOFT: 0))[Soft? No.] \
#text(variations: (SOFT: 100))[Soft? Yes.]
```

Soft? No.

Soft? Yes.

Example · Reemplazar valores establecidos automáticamente

```
#set text(
  font: "Roboto Flex",
  weight: 900,
  stretch: 150%,
)

Wide and Heavy

#text(variations: (wght: 400))[
  Forced back to normal,
  but still wide.
]
```

**Wide and Heavy**

Forced back to normal, but still wide.

/2.34 **body** `content` *Obligatorio* *Posicional*

Contenido en el que todo el texto recibe estilo según los demás argumentos.

Este parámetro solo está disponible en el constructor de `text` (y no en los elementos `text` existentes), porque el constructor de `text` no necesariamente construye un único elemento `text`. En cambio, aplica estilos a todos los elementos de texto dentro del cuerpo que se le pasa.

```
#text(red)[
  The `body` can be _arbitrary_
  #underline[content].
]
```

The body can be *arbitrary* content.

/2.35 **text** `str` *Obligatorio* *Posicional*

El texto contenido en un elemento `text`.

Este parámetro no está disponible en el constructor. En cambio, podés acceder a él en un elemento `text` existente. Esto se debe a que el constructor de `text` acepta contenido arbitrario como cuerpo (ver `text.body`[3.7.12/2.34]).

```
#show regex("i.+n"): it => {
  "i" + str(it.text.len() - 2) + "n"
}
internationalization
```

i18n

## 3.7.13 `underline` Elemento Desde: siempre

Subraya texto.

### /1 Ejemplo

```
This is #underline[important].
```

This is important.

### /2 Parámetros

```
underline(
  stroke: auto length color gradient stroke tiling dictionary,
  offset: auto length,
  extent: length,
  evade: bool,
  background: bool,
  content,
) → content
```

#### /2.1 `stroke` auto or length or color or gradient or stroke or tiling or dictionary

*Configurable con set* Por defecto: `auto`

Cómo trazar[3.11.12] la línea.

Si se establece en `auto`, toma el color del texto y un grosor definido en la fuente actual.

```
Take #underline(
  stroke: 1.5pt + red,
  offset: 2pt,
  [care],
)
```

Take care

#### /2.2 `offset` auto or length *Configurable con set* Por defecto: `auto`

La posición de la línea relativa a la línea base, que se lee de las tablas de la fuente si es `auto`.

```
#underline(offset: 5pt)[
  The Tale Of A Faraway Line I
]
```

The Tale Of A Faraway Line I

#### /2.3 `extent` length *Configurable con set* Por defecto: `0pt`

La cantidad en que se extiende la línea más allá del contenido (o hacia adentro, si es negativa).

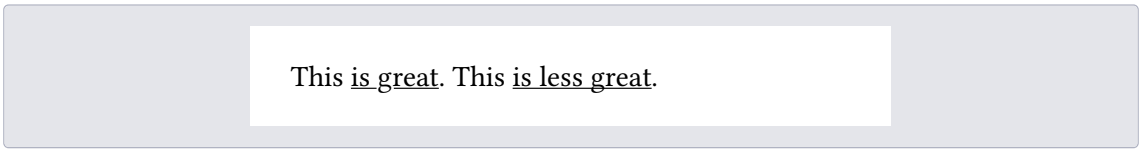
```
#align(center,
  underline(extent: 2pt)[Chapter 1]
)
```

Chapter 1

#### /2.4 `evade` bool *Configurable con set* Por defecto: `true`

Si la línea omite las secciones en las que chocaría con los glifos.

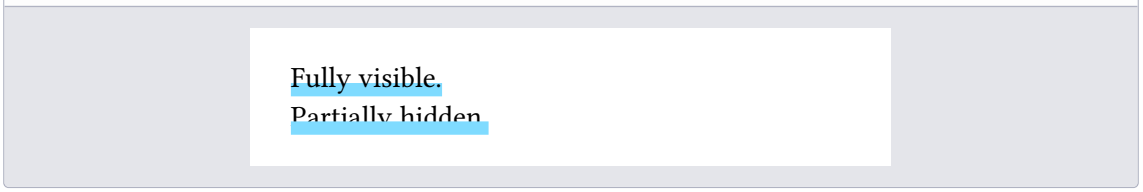
```
This #underline(evade: true)[is great].
This #underline(evade: false)[is less great].
```




/2.5 **background** `bool` *Configurable con set* Por defecto: `false`

Si la línea se ubica detrás del contenido que subraya.

```
#set underline(stroke: aqua + 5pt, evade: false)
#underline(background: true)[Fully visible.] \
#underline(background: false)[Partially hidden.]
```




/2.6 **body** `content` *Obligatorio* *Posicional*

El contenido que se subraya.

### 3.7.14 `upper` Desde: siempre

Convierte una cadena de texto o contenido a mayúsculas.

#### /1 Ejemplo

```
#upper("abc") \
#upper[*my text*] \
#upper[ALREADY HIGH]
```

ABC
**MY TEXT**
ALREADY HIGH

#### /2 Parámetros

```
upper( str content ) → str content
```

**/2.1 `text`** `str` or `content` *Obligatorio* *Posicional*

El texto que se convierte a mayúsculas.

## 3.8 Matemática

Typst tiene una sintaxis[3.1.3] especial y funciones de biblioteca para componer fórmulas matemáticas. Las fórmulas matemáticas se pueden mostrar en línea con el texto o como bloques separados. Se compondrán en su propio bloque si empiezan y terminan con al menos un espacio (por ejemplo, `$ x^2 $`).

### 3.8.1 Variables

En matemática, las letras sueltas siempre se muestran tal cual. Las secuencias de varias letras, en cambio, se interpretan como variables y funciones. Para mostrar varias letras literalmente, podés ponerlas entre comillas y, para acceder a variables de una sola letra, podés usar la sintaxis con numeral[3.3.1].

```
$ A = pi r^2 $
$ "area" = pi dot "radius"^2 $
$ cal(A) :=
    { x in RR | x "is natural" } $
#let x = 5
$ #x < 17 $
```

$$A = \pi r^2$$

$$\text{area} = \pi \cdot \text{radius}^2$$

$$\mathcal{A} := \{x \in \mathbb{R} \mid x \text{ is natural}\}$$

$$5 < 17$$

### 3.8.2 Símbolos

El modo matemático pone a disposición una amplia selección de símbolos[3.9.2], como `pi`, `dot` o `RR`. Muchos símbolos matemáticos están disponibles en distintas variantes. Podés elegir entre las distintas variantes aplicando modificadores[3.5.29] al símbolo. Además, Typst reconoce varias secuencias abreviadas, como `=>`, que aproximan un símbolo. Cuando existe una abreviatura así, la documentación del símbolo la enumera.

```
$ x < y => x gt.eq.not y $
```

$$x < y \Rightarrow x \gneq y$$

### 3.8.3 Saltos de línea

Las fórmulas también pueden contener saltos de línea. Cada línea puede contener uno o varios *puntos de alineación* (`&`) que luego se alinean.

```
$ sum_(k=0)^n k
    &= 1 + ... + n \
    &= (n(n+1)) / 2 $
```

$$\sum_{k=0}^{n} k = 1 + ... + n$$
$$= \frac{n(n+1)}{2}$$

### 3.8.4 Llamadas a funciones

El modo matemático admite llamadas a funciones especiales sin el prefijo de numeral. En estas "llamadas matemáticas", la lista de argumentos funciona un poco distinto que en el código:

- Dentro de ellas, Typst sigue en "modo matemático". Por lo tanto, podés escribir matemática directamente en ellas, pero necesitás usar la sintaxis con numeral para pasar expresiones de código (excepto las cadenas de texto, que están disponibles en la sintaxis matemática).
- Admiten argumentos posicionales y con nombre, así como la expansión de argumentos, pero no admiten bloques de contenido finales.

– Ofrecen sintaxis adicional para listas de argumentos bidimensionales. El punto y coma (`;`) reúne en un argumento de tipo array los argumentos anteriores separados por comas.

```
$ frac(a^2, 2) $
$ vec(1, 2, delim: "[") $
$ mat(1, 2; 3, 4) $
$ mat(..#range(1, 5).chunks(2)) $
$ lim_x =
    op("lim", limits: #true)_x $
```

$$\frac{a^2}{2}$$

$$\begin{bmatrix}1\\2\end{bmatrix}$$

$$\begin{pmatrix}1 & 2\\3 & 4\end{pmatrix}$$

$$\begin{pmatrix}1 & 2\\3 & 4\end{pmatrix}$$

$$\lim_x = \lim_x$$

Para escribir una coma o un punto y coma literales en una llamada matemática, escapalos con una barra invertida. Los dos puntos, en cambio, solo se reconocen de manera especial si están inmediatamente precedidos por un identificador, así que, para mostrarlos literalmente en esos casos, podés simplemente insertar un espacio antes.

Las llamadas a funciones precedidas por un numeral son llamadas normales a funciones de código y no se ven afectadas por estas reglas.

### 3.8.5 Alineación

Cuando las ecuaciones incluyen varios *puntos de alineación* (`&`), esto crea bloques de columnas alineadas alternadamente a la derecha y a la izquierda. En el siguiente ejemplo, la expresión `(3x + y) / 7` está alineada a la derecha y `= 9` está alineado a la izquierda. La palabra "given" también está alineada a la izquierda porque `&&` crea dos puntos de alineación seguidos, alternando dos veces la alineación. `& &` y `&&` se comportan exactamente igual. Mientras tanto, "multiply by 7" está alineado a la derecha porque solo lo precede un `&`. Cada punto de alineación simplemente alterna entre alineado a la derecha y alineado a la izquierda.

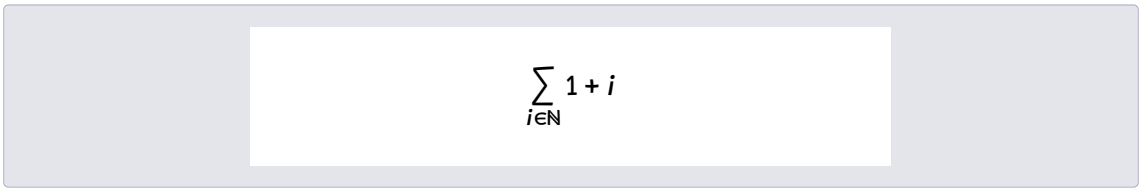
```
$ (3x + y) / 7 &= 9 && "given" \
  3x + y &= 63 & "multiply by 7" \
  3x &= 63 - y && "subtract y" \
  x &= 21 - y/3 & "divide by 3" $
```

$$\begin{aligned}\frac{3x+y}{7} &= 9 && \text{given}\\ 3x+y &= 63 && \text{multiply by 7}\\ 3x &= 63-y && \text{subtract y}\\ x &= 21-\frac{y}{3} && \text{divide by 3}\end{aligned}$$

### 3.8.6 Fuentes matemáticas

Podés establecer la fuente matemática con una regla show-set[3.2.2], como se demuestra a continuación. Tené en cuenta que solo las fuentes matemáticas especiales de OpenType son adecuadas para la composición matemática.

```
#show math.equation: set text(font: "Pennstander Math")
$ sum_(i in NN) 1 + i $
```



$$\sum_{i \in \mathbb{N}} 1 + i$$

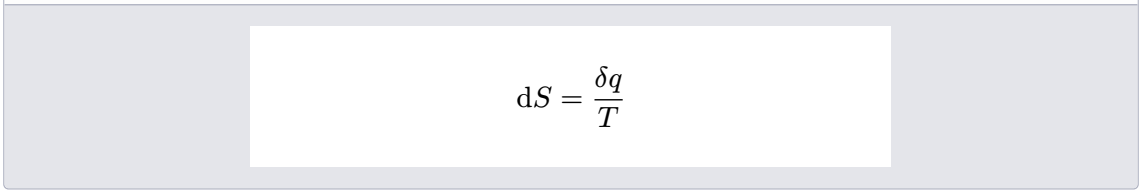
## 3.8.7 Módulo math

Todas las funciones matemáticas forman parte del módulo[3.3.8] `math`, que está disponible por defecto en las ecuaciones. Fuera de las ecuaciones, se puede acceder a ellas con el prefijo `math.`.

## 3.8.8 Accesibilidad

Para que la matemática sea accesible, tenés que proporcionar descripciones alternativas de las ecuaciones en lenguaje natural con el parámetro `alt` de `math.equation`[3.8.16/3.5]. Para más información, mirá la sección de representaciones textuales de la guía de accesibilidad[4.4.7].

```
#math.equation(
  alt: "d S equals delta q divided by T",
  block: true,
  $ dif S = (delta q) / T $,
)
```



$$\mathrm{d}S = \frac{\delta q}{T}$$

En el futuro, Typst hará accesibles automáticamente las ecuaciones sin descripciones alternativas en la exportación a HTML y a PDF 2.0.

# Definiciones


3.8.9 Accent Attaches an accent to a base. 284
3.8.10 Adjuntos Subíndices, superíndices y límites. 286
3.8.11 Binomial A binomial expression. 288
3.8.12 Cancel Displays a diagonal line over a part of an equation. 289
3.8.13 Cases A case distinction. 291
3.8.14 Class Forced use of a certain math class. 292
3.8.15 Debajo/Encima Delimitadores encima o debajo de partes de una ecuación. 293
3.8.16 Equation A mathematical equation. 297
3.8.17 Espacios Espacios matemáticos predefinidos de varios anchos 300
3.8.18 Estilos Formas de letras alternativas dentro de las fórmulas. 301
3.8.19 Fraction A mathematical fraction. 302
3.8.20 Izquierda/Derecha Emparejamiento de delimitadores. 304
3.8.21 Matrix A matrix. 308
3.8.22 Primes Grouped primes. 311
3.8.23 Raíces Raíces cuadradas y no cuadradas. 312
3.8.24 Stretch Stretches a glyph. 313
3.8.25 Tamaños Estilos de tamaño forzado para las expresiones dentro de las fórmulas. 315
3.8.26 Text Operator A text operator in an equation. 317
3.8.27 Variantes Tipografías alternativas dentro de las fórmulas. 318
3.8.28 Vector A column vector. 321

### 3.8.9 `accent` Elemento Desde: siempre

Agrega un acento a una base.

En modo matemático, los acentos comunes también están disponibles como símbolos[3.5.29] con nombre que se pueden llamar directamente (como funciones[3.5.16]) para adjuntarlos a algún contenido.

#### /1 Ejemplo

```
$grave(a) = accent(a, `)$ \
$arrow(a) = accent(a, arrow)$ \
$tilde(a) = accent(a, \u{0303})$
```

$$\grave{a} = \grave{a}$$
$$\vec{a} = \vec{a}$$
$$\tilde{a} = \tilde{a}$$

#### /2 Parámetros

```
math.accent(
  content,
  str content,
  size: relative,
  dotless: bool,
) → content
```

##### /2.1 base `content` *Obligatorio* *Posicional*

La base a la que se aplica el acento. Puede estar formada por varias letras.

```
$arrow(A B C)$
```

$$\overrightarrow{ABC}$$

##### /2.2 accent `str` or `content` *Obligatorio* *Posicional*

El acento que se aplica a la base.

Los acentos admitidos incluyen:

| Acento | Nombre | Punto de código |
|---|---|---|
| Grave | `grave` | ` |
| Agudo | `acute` | ´ |
| Circunflejo | `hat` | ^ |
| Tilde | `tilde` | ~ |
| Macron | `macron` | ¯ |
| Guion | `dash` | ‾ |
| Breve | `breve` | ˘ |
| Punto | `dot` | . |
| Punto doble, diéresis | `dot.double`, `diaer` | ¨ |
| Punto triple | `dot.triple` | \u{20db} |
| Punto cuádruple | `dot.quad` | \u{20dc} |
| Círculo | `circle` | ∘ |
| Agudo doble | `acute.double` | ˝ |
| Caron | `caron` | ˇ |
| Flecha a la derecha | `arrow`, `->` | → |

| Acento | Nombre | Punto de código |
|---|---|---|
| Flecha a la izquierda | `arrow.l`, `<-` | ← |
| Flecha izquierda/derecha | `arrow.l.r` | ↔ |
| Arpón a la derecha | `harpoon` | ⇀ |
| Arpón a la izquierda | `harpoon.lt` | ↼ |

### /2.3 `size` `relative` *Configurable con set*

Por defecto: `100% + 0pt`

El tamaño del acento, relativo al ancho de la base.

Example · Uso básico

```
$dash(A, size: #150%)$
```

$\overline{A}$

Tené en cuenta que el acento resultante puede no tener exactamente el tamaño deseado. Por ejemplo, una flecha puede ser un glifo corto predefinido o un glifo largo armado con piezas (punta de flecha + línea) que proporciona la fuente. Los tamaños de las dos posibilidades pueden no cubrir todo el rango. En consecuencia, no se pueden construir flechas de ciertos tamaños intermedios.

Example · Tamaño de la flecha que crece de forma discontinua

```
#for i in range(6) {
  $ arrow(#box(
    width: 0.4em + 0.3em * i,
    fill: aqua,
    height: 0.4em,
  )) $
}
```



### /2.4 `dotless` `bool` *Configurable con set*

Por defecto: `true`

Si se quita el punto de las i y j minúsculas al agregar un acento superior.

Esto habilita la característica OpenType `dtls`.

```
$hat(dotless: #false, i)$
```

$\hat{i}$

## 3.8.10 Adjuntos

Subíndices, superíndices y límites.

Los adjuntos se pueden mostrar como subíndices/superíndices o como límites. Typst decide automáticamente cuál es más adecuado según la base, pero también podés controlarlo manualmente con las funciones `scripts` y `limits`.

Si querés que la base se estire para ajustarse a adjuntos superiores e inferiores largos (por ejemplo, una flecha con texto encima), usá la función `stretch`[3.8.24].

### /1 Example

```
$ sum_(i=0)^n a_i = 2^(1+i) $
```

$$\sum_{i=0}^{n} a_i = 2^{1+i}$$

### /2 Syntax

Esta función también tiene una sintaxis dedicada para los adjuntos después de la base: usá el guion bajo (_) para indicar un subíndice, es decir, un adjunto inferior, y el acento circunflejo (^) para indicar un superíndice, es decir, un adjunto superior.

### /3 Funciones

#### /3.1 `attach` Elemento Desde: siempre

Una base con adjuntos opcionales.

```
$ attach(
  Pi, t: alpha, b: beta,
  tl: 1, tr: 2+3, bl: 4+5, br: 6,
) $
```

$$\prod_{4+5}^{\alpha}{}_{1}^{2+3}$$

Si querés agregar acentos (sombreros, tildes, flechas, etcétera) en lugar de subíndices/superíndices o adjuntos en las esquinas, usá en cambio la función `accent`[3.8.9].

```
math.attach(
  content,
  t: none content,
  b: none content,
  tl: none content,
  bl: none content,
  tr: none content,
  br: none content,
) → content
```

##### /3.1.1 **base** content *Obligatorio* *Posicional*

La base a la que se adjuntan las cosas.

##### /3.1.2 **t** none or content *Configurable con set* Por defecto: none

El adjunto superior, ubicado de forma inteligente arriba a la derecha o encima de la base.

Podés envolver la base en `limits()` o `scripts()` para reemplazar el posicionamiento inteligente.

##### /3.1.3 **b** none or content *Configurable con set* Por defecto: none

El adjunto inferior, ubicado de forma inteligente abajo a la derecha o debajo de la base.

Podés envolver la base en `limits()` o `scripts()` para reemplazar el posicionamiento inteligente.

/3.1.4 **tl** `none` or `content` *Configurable con set* Por defecto: none

El adjunto superior izquierdo (antes de la base).

/3.1.5 **bl** `none` or `content` *Configurable con set* Por defecto: none

El adjunto inferior izquierdo (antes de la base).

/3.1.6 **tr** `none` or `content` *Configurable con set* Por defecto: none

El adjunto superior derecho (después de la base).

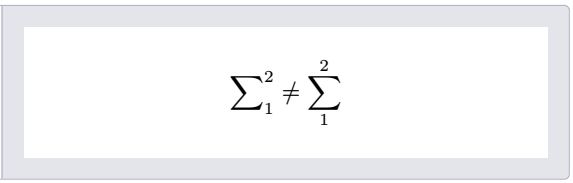
/3.1.7 **br** `none` or `content` *Configurable con set* Por defecto: none

El adjunto inferior derecho (después de la base).

## /3.2 scripts Elemento Desde: siempre

Fuerza a una base a mostrar los adjuntos como subíndices/superíndices.

```
$ scripts(sum)_1^2 != sum_1^2 $
```



```
math.scripts(content) → content
```

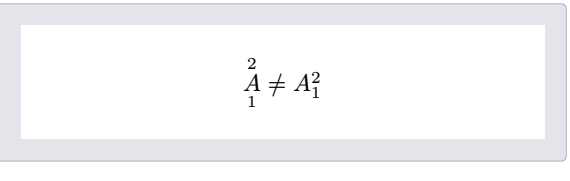
/3.2.1 **body** `content` *Obligatorio* *Posicional*

La base a la que se adjuntan los subíndices/superíndices.

## /3.3 limits Elemento Desde: siempre

Fuerza a una base a mostrar los adjuntos como límites.

```
$ limits(A)_1^2 != A_1^2 $
```



```
math.limits(
  content,
  inline: bool,
) → content
```

/3.3.1 **body** `content` *Obligatorio* *Posicional*

La base a la que se adjuntan los límites.

/3.3.2 **inline** `bool` *Configurable con set* Por defecto: true

Si también se fuerzan los límites en las ecuaciones en línea.

Al aplicar límites globalmente (por ejemplo, mediante una regla show), normalmente conviene desactivar esto.

### 3.8.11 `binom` Elemento Desde: siempre

Una expresión binomial.

#### /1 Ejemplo

```
$ binom(n, k) $
$ binom(n, k_1, k_2, k_3, ..., k_m) $
```

$$\binom{n}{k}$$

$$\binom{n}{k_1, k_2, k_3, ..., k_m}$$

#### /2 Parámetros

```
math.binom(
  content,
  ..content,
) → content
```

/2.1 **upper** `content` *Obligatorio* *Posicional*

El índice superior del binomial.

/2.2 **lower** `content` *Obligatorio* *Posicional* *Variádico*

El índice inferior del binomial.

## 3.8.12 `cancel` Elemento Desde: 0.3.0

Muestra una línea diagonal sobre una parte de una ecuación.

Se usa comúnmente para mostrar la eliminación de un término.

### /1 Ejemplo

```
Here, we can simplify:
$ (a dot b dot cancel(x)) /
    cancel(x) $
```

Here, we can simplify:

$$\frac{a \cdot b \cdot \cancel{x}}{\cancel{x}}$$

### /2 Parámetros

```
math.cancel(
  content,
  length: relative,
  inverted: bool,
  cross: bool,
  angle: auto angle function,
  stroke: length color gradient stroke tiling dictionary,
  background: bool,
) → content
```

#### /2.1 **body** `content` *Obligatorio* *Posicional*

El contenido sobre el que se ubica la línea.

#### /2.2 **`length`** `relative` *Configurable con set* Por defecto: `100% + 0.3em`

La longitud de la línea, relativa a la longitud de la diagonal que abarca todo el elemento que se "cancela". Un valor de `100%` haría que la línea abarque exactamente la diagonal del elemento.

```
$ a + cancel(x, length: #200%)
    - cancel(x, length: #200%) $
```

$$a + \cancel{x} - \cancel{x}$$

#### /2.3 **`inverted`** `bool` *Configurable con set* Por defecto: `false`

Si la línea de cancelación se invierte (se voltea a lo largo del eje y). Con el ángulo por defecto, invertida significa que la línea de cancelación apunta hacia arriba a la izquierda en lugar de hacia arriba a la derecha.

```
$ (a cancel((b + c), inverted: #true)) /
    cancel(b + c, inverted: #true) $
```

$$\frac{a\cancel{(b + c)}}{\cancel{b + c}}$$

#### /2.4 **`cross`** `bool` *Configurable con set* Por defecto: `false`

Si se dibujan dos líneas de cancelación opuestas que forman una cruz sobre el elemento. Reemplaza a `inverted`.

```
$ cancel(Pi, cross: #true) $
```

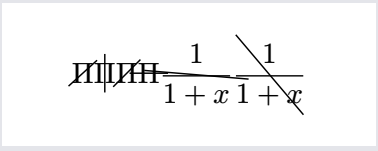


### /2.5 angle `auto` or `angle` or `function` *Configurable con set* Por defecto: `auto`

Cuánto se rota la línea de cancelación.

- Si se da un ángulo, la línea se rota ese ángulo en sentido horario respecto del eje y.
- Si es `auto`, la línea toma el ángulo por defecto; es decir, a lo largo de la diagonal ascendente del cuadro de contenido.
- Si se da una función `angle => angle`, la línea se rota, respecto del eje y, el ángulo que devuelve esa función. La función recibe como entrada el ángulo por defecto.

```
$ cancel(Pi)
  cancel(Pi, angle: #0deg)
  cancel(Pi, angle: #45deg)
  cancel(Pi, angle: #90deg)
  cancel(1/(1+x), angle: #(a => a + 45deg))
  cancel(1/(1+x), angle: #(a => a + 90deg)) $
```



### /2.6 stroke `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `0.05em`

Cómo trazar[3.11.12] la línea de cancelación.

```
$ cancel(
  sum x,
  stroke: #(
    paint: red,
    thickness: 1.5pt,
    dash: "dashed",
  ),
) $
```



### /2.7 background `bool` *Configurable con set* Por defecto: `false`

Si la línea se ubica detrás del contenido.

```
#set math.cancel(stroke: red + 2pt)
$ cancel("behind", background: #true) \
  cancel("in front", background: #false) $
```



### 3.8.13 `cases` Elemento Desde: siempre

Una distinción de casos.

El contenido de las distintas ramas se puede alinear con el símbolo &.

#### /1 Ejemplo

```
$ f(x, y) := cases(
  1 "if" (x dot y)/2 <= 0,
  2 "if" x "is even",
  3 "if" x in NN,
  4 "else",
) $
```

$$f(x, y) := \begin{cases} 1 \text{ if } \frac{x \cdot y}{2} \leq 0 \\ 2 \text{ if } x \text{ is even} \\ 3 \text{ if } x \in \mathbb{N} \\ 4 \text{ else} \end{cases}$$

#### /2 Parámetros

```
math.cases(
  delim: none str symbol array,
  reverse: bool,
  gap: relative,
  ..content,
) → content
```

##### /2.1 `delim` `none` or `str` or `symbol` or `array` *Configurable con set* Por defecto: `("{", "}")`

El delimitador que se usa.

Puede ser un único carácter que especifica el delimitador izquierdo, en cuyo caso se infiere el delimitador derecho. De lo contrario, puede ser un array que contenga un delimitador izquierdo y uno derecho.

```
#set math.cases(delim: "[")
$ x = cases(1, 2) $
```

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

##### /2.2 `reverse` `bool` *Configurable con set* Por defecto: `false`

Si se invierte la dirección de los casos.

```
#set math.cases(reverse: true)
$ cases(1, 2) = x $
```

$$\left.\begin{matrix} 1 \\ 2 \end{matrix}\right\} = x$$

##### /2.3 `gap` `relative` *Configurable con set* Por defecto: `0% + 0.2em`

El espacio entre las ramas.

```
#set math.cases(gap: 1em)
$ x = cases(1, 2) $
```

$$x = \begin{cases} 1 \\ 2 \end{cases}$$

##### /2.4 `children` `content` *Obligatorio* *Posicional* *Variádico*

Las ramas de la distinción de casos.

### 3.8.14 `class` Elemento Desde: siempre

Uso forzado de una cierta clase matemática.

Esto es útil para tratar ciertos símbolos como si fueran de otra clase, por ejemplo, para que un símbolo se comporte como una relación. La clase de un símbolo define cómo se dispone, incluido el espaciado a su alrededor y cómo se adjuntan por defecto sus subíndices/superíndices. Tené en cuenta que esto último siempre se puede reemplazar con `limits`[3.8.10/3.3] y `scripts`[3.8.10/3.2].

#### /1 Ejemplo

```
#let loves = math.class(
  "relation",
  sym.suit.heart,
)

$x loves y and y loves 5$
```

$x \heartsuit y \land y \heartsuit 5$

#### /2 Parámetros

```
math.class(
  str,
  content,
) → content
```

##### /2.1 `class` `str` *Obligatorio* *Posicional*

La clase que se aplica al contenido.

| Variante | Detalles |
|---|---|
| `"normal"` | La clase predeterminada para las cosas que no son especiales. |
| `"punctuation"` | Puntuación, p. ej., una coma. |
| `"opening"` | Un delimitador de apertura, p. ej., `(`. |
| `"closing"` | Un delimitador de cierre, p. ej., `)`. |
| `"fence"` | Un delimitador que es igual de ambos lados, p. ej., `\|`. |
| `"large"` | Un operador grande como `sum`.<br>Si el cuerpo es un único glifo, esta clase lo centra verticalmente sobre el eje matemático (donde se ubica la línea de la fracción) y lo estira verticalmente en el estilo `display`. |
| `"relation"` | Una relación como `=` o `prec`. |
| `"unary"` | Un operador unario como `not`. |
| `"binary"` | Un operador binario como `times`. |
| `"vary"` | Un operador que puede ser unario o binario, como `+`. |
| `"diacritic"` | Un carácter de acento. |

##### /2.2 `body` `content` *Obligatorio* *Posicional*

El contenido al que se aplica la clase.

## 3.8.15 Debajo/Encima

Delimitadores encima o debajo de partes de una ecuación.

Además, las llaves y los corchetes te permiten agregar una anotación opcional debajo o encima de ellos.

Estas funciones están pensadas específicamente para agregar delimitadores. Si querés ubicar dos partes arbitrarias de una ecuación una encima o debajo de la otra, sin delimitadores, usá en cambio la función attach[3.8.10/3.1].

### /1 Funciones

#### /1.1 underline Elemento Desde: siempre

Una línea horizontal bajo el contenido.

```
$ underline(1 + 2 + ... + 5) $
```

$$\underline{1 + 2 + ... + 5}$$

```
math.underline(content) → content
```

##### /1.1.1 body content *Obligatorio* *Posicional*

El contenido encima de la línea.

#### /1.2 overline Elemento Desde: siempre

Una línea horizontal sobre el contenido.

```
$ overline(1 + 2 + ... + 5) $
```

$$\overline{1 + 2 + ... + 5}$$

```
math.overline(content) → content
```

##### /1.2.1 body content *Obligatorio* *Posicional*

El contenido debajo de la línea.

#### /1.3 underbrace Elemento Desde: siempre

Una llave horizontal bajo el contenido, con una anotación opcional debajo.

```
$ underbrace(0 + 1 + dots.c + n, n + 1 "numbers") $
```

$$\underbrace{0 + 1 + \cdots + n}_{n+1 \text{ numbers}}$$

```
math.underbrace(
  content,
  none content,
) → content
```

##### /1.3.1 body content *Obligatorio* *Posicional*

El contenido encima de la llave.

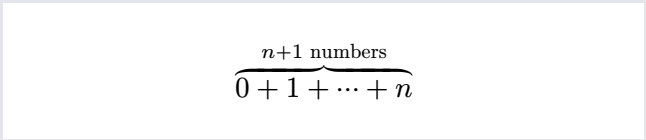
/1.3.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional debajo de la llave.

/1.4 **overbrace** Elemento Desde: siempre

Una llave horizontal sobre el contenido, con una anotación opcional encima.

```
$ overbrace(0 + 1 + dots.c + n, n + 1 "numbers") $
```




```
math.overbrace(
  content,
  none content,
) → content
```

/1.4.1 **body** `content` *Obligatorio* *Posicional*

El contenido debajo de la llave.

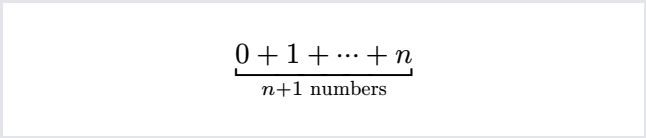
/1.4.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional encima de la llave.

/1.5 **underbracket** Elemento Desde: siempre

Un corchete horizontal bajo el contenido, con una anotación opcional debajo.

```
$ underbracket(0 + 1 + dots.c + n, n + 1 "numbers") $
```




```
math.underbracket(
  content,
  none content,
) → content
```

/1.5.1 **body** `content` *Obligatorio* *Posicional*

El contenido encima del corchete.

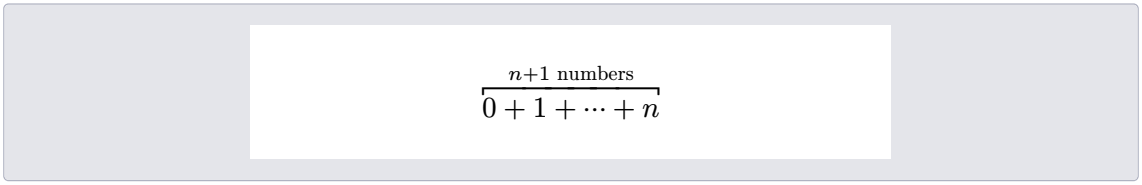
/1.5.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional debajo del corchete.

/1.6 **overbracket** Elemento Desde: siempre

Un corchete horizontal sobre el contenido, con una anotación opcional encima.

```
$ overbracket(0 + 1 + dots.c + n, n + 1 "numbers") $
```




```
math.overbracket(
  content,
  none content,
) → content
```

/1.6.1 **body** `content` *Obligatorio* *Posicional*

El contenido debajo del corchete.

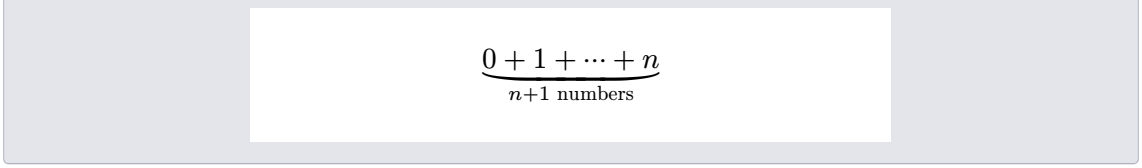
/1.6.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional encima del corchete.

/1.7 **underparen** Elemento Desde: siempre

Un paréntesis horizontal bajo el contenido, con una anotación opcional debajo.

```
$ underparen(0 + 1 + dots.c + n, n + 1 "numbers") $
```




```
math.underparen(
  content,
  none content,
) → content
```

/1.7.1 **body** `content` *Obligatorio* *Posicional*

El contenido encima del paréntesis.

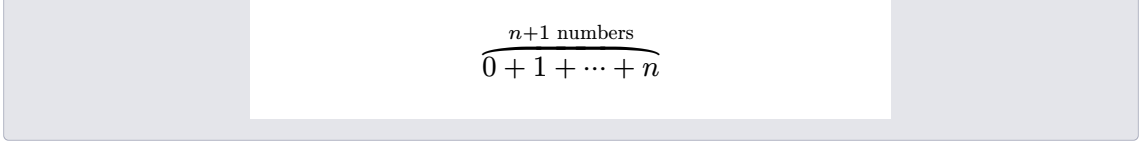
/1.7.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional debajo del paréntesis.

/1.8 **overparen** Elemento Desde: siempre

Un paréntesis horizontal sobre el contenido, con una anotación opcional encima.

```
$ overparen(0 + 1 + dots.c + n, n + 1 "numbers") $
```




```
math.overparen(
  content,
  none content,
) → content
```

/1.8.1 **body** `content` *Obligatorio* *Posicional*

El contenido debajo del paréntesis.

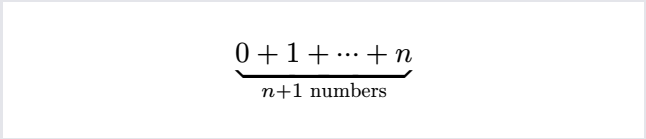
/1.8.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional encima del paréntesis.

/1.9 **undershell** Elemento Desde: siempre

Un corchete de carey (tortoise shell) horizontal bajo el contenido, con una anotación opcional debajo.

```
$ undershell(0 + 1 + dots.c + n, n + 1 "numbers") $
```



```
math.undershell(
  content,
  none content,
) → content
```

/1.9.1 **body** `content` *Obligatorio* *Posicional*

El contenido encima del corchete de carey.

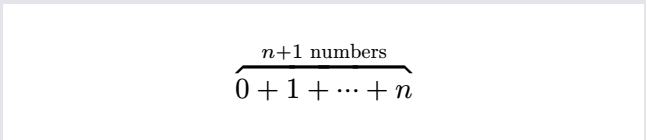
/1.9.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional debajo del corchete de carey.

/1.10 **overshell** Elemento Desde: siempre

Un corchete de carey (tortoise shell) horizontal sobre el contenido, con una anotación opcional encima.

```
$ overshell(0 + 1 + dots.c + n, n + 1 "numbers") $
```



```
math.overshell(
  content,
  none content,
) → content
```

/1.10.1 **body** `content` *Obligatorio* *Posicional*

El contenido debajo del corchete de carey.

/1.10.2 **annotation** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido opcional encima del corchete de carey.

## 3.8.16 `equation` Elemento Desde: siempre

Una ecuación matemática.

Se puede mostrar en línea con el texto o como un bloque separado. Una ecuación pasa a ser de bloque por la presencia de un espacio en blanco después del signo de pesos inicial y un espacio en blanco antes del signo de pesos final.

### /1 Ejemplo

```
#set text(font: "New Computer Modern")

Let $a$, $b$, and $c$ be the side
lengths of a right-angled triangle.
Then, we know that:
$ a^2 + b^2 = c^2 $

Prove by induction:
$ sum_(k=1)^n k = (n(n+1)) / 2 $
```

Let $a$, $b$, and $c$ be the side lengths of a right-angled triangle. Then, we know that:

$$a^2 + b^2 = c^2$$

Prove by induction:

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Por defecto, las ecuaciones de bloque no se cortan entre páginas. Esto se puede cambiar con `show math.equation: set block(breakable: true)`.

### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: escribí marcado matemático entre signos de pesos para crear una ecuación. Empezar y terminar la ecuación con un espacio en blanco la eleva a un bloque separado centrado horizontalmente. Para más detalles sobre la sintaxis matemática, mirá la página principal de matemática[3.8].

### /3 Parámetros

```
math.equation(
  block: bool,
  numbering: none str function,
  number-align: alignment,
  supplement: none auto content function,
  alt: none str,
  content,
) → content
```

#### /3.1 `block` `bool` *Configurable con set* Por defecto: `false`

Si la ecuación se muestra como un bloque separado.

#### /3.2 `numbering` `none` or `str` or `function` *Configurable con set* Por defecto: `none`

Cómo numerar las ecuaciones de bloque. Acepta un patrón o función de numeración[3.6.12] que toma un único número.

```
#set math.equation(numbering: "(1)")

We define:
$ phi.alt := (1 + sqrt(5)) / 2 $ <ratio>

With @ratio, we get:
$ F_n = floor(1 / sqrt(5) phi.alt^n) $
```

We define:

$$\phi := \frac{1 + \sqrt{5}}{2} \tag{1}$$

With Ecuación 1, we get:

$$F_n = \left\lfloor \frac{1}{\sqrt{5}} \phi^n \right\rfloor \tag{2}$$

### /3.3 number-align `alignment` *Configurable con set* Por defecto: end + horizon

La alineación de la numeración de la ecuación.

Por defecto, la alineación es `end + horizon`. Para el componente horizontal, podés usar `right`, `left` o `start` y `end` de la dirección del texto; para el componente vertical, podés usar `top`, `horizon` o `bottom`.

```
#set math.equation(numbering: "(1)", number-align: bottom)

We can calculate:
$ E &= sqrt(m_0^2 + p^2) \
    &approx 125 "GeV" $
```

We can calculate:

$$\begin{aligned} E &= \sqrt{m_0^2 + p^2} \\ &\approx 125 \text{ GeV} \end{aligned} \tag{1}$$

### /3.4 supplement `none` or `auto` or `content` or `function` *Configurable con set* Por defecto: auto

Un suplemento para la ecuación.

En las referencias a ecuaciones, se agrega antes del número referenciado.

Si se especifica una función, se le pasa la ecuación referenciada y debe devolver contenido.

```
#set math.equation(numbering: "(1)", supplement: [Eq.])

We define:
$ phi.alt := (1 + sqrt(5)) / 2 $ <ratio>

With @ratio, we get:
$ F_n = floor(1 / sqrt(5) phi.alt^n) $
```

We define:

$$\phi := \frac{1 + \sqrt{5}}{2} \tag{1}$$

With Eq. 1, we get:

$$F_n = \left\lfloor \frac{1}{\sqrt{5}} \phi^n \right\rfloor \tag{2}$$

/3.5 **alt** `none` or `str` *Configurable con set* Por defecto: none

Una descripción alternativa de la ecuación matemática.

Debería describir la ecuación completa en lenguaje natural y se pondrá a disposición de las tecnologías de asistencia. Podés aprender más en la sección de representaciones textuales de la guía de accesibilidad[4.4.7].

```
#math.equation(
  alt: "integral from 1 to infinity of a x squared plus b with respect to x",
  block: true,
  $ integral_1^oo a x^2 + b dif x $,
)
```

$$\int_1^\infty ax^2 + b \,\mathrm{d}x$$

/3.6 **body** `content` *Obligatorio* *Posicional*

El contenido de la ecuación.

## 3.8.17 Espacios

Espacios matemáticos predefinidos de varios anchos.

### /1 Constantes

#### /1.1 thin `content` Desde: siempre

Un espacio matemático de ancho un sexto de em.

```
$
  x y \
  x thin y
$
```

$$xy$$
$$x\,y$$

#### /1.2 med `content` Desde: siempre

Un espacio matemático de ancho dos novenos de em.

```
$ (lambda x. x) med y $
```

$$(\lambda x.x)\:y$$

#### /1.3 thick `content` Desde: siempre

Un espacio matemático de ancho cinco dieciochoavos de em.

```
$ x thick y $
```

$$x\;y$$

#### /1.4 quad `content` Desde: siempre

Un espacio matemático de un em de ancho.

```
$
  f(x) = cases(
    1/x quad &"if" x != 0,
    0 quad &"otherwise",
  )
$
```

$$f(x) = \begin{cases} \frac{1}{x} & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

#### /1.5 wide `content` Desde: siempre

Un espacio matemático de dos em de ancho.

```
$
  f(x) = cases(
    1/x wide &"if" x != 0,
    0 wide &"otherwise",
  )
$
```

$$f(x) = \begin{cases} \frac{1}{x} & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

## 3.8.18 Estilos

Formas de letras alternativas dentro de las fórmulas.

Estas funciones son distintas de la función `text`[3.7.12] porque las fuentes matemáticas contienen varias variantes de cada letra.

### /1 Funciones

#### /1.1 `upright` Desde: siempre

Estilo de fuente vertical (no cursiva) en matemática.

```
$ upright(A) != A $
```

$$\mathrm{A} \neq A$$

```
math.upright(content) → content
```

##### /1.1.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

#### /1.2 `italic` Desde: siempre

Estilo de fuente cursiva en matemática.

Para las letras romanas y las letras griegas minúsculas, esto ya es el valor por defecto.

```
math.italic(content) → content
```

##### /1.2.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

#### /1.3 `bold` Desde: siempre

Estilo de fuente en negrita en matemática.

```
$ bold(A) := B^+ $
```

$$\boldsymbol{A} := B^+$$

```
math.bold(content) → content
```

##### /1.3.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

### 3.8.19 `frac` Elemento Desde: siempre

Una fracción matemática.

#### /1 Ejemplo

```
$ 1/2 < (x+1)/2 $
$ ((x+1)) / 2 = frac(a, b) $
```

$$\frac{1}{2} < \frac{x+1}{2}$$

$$\frac{(x+1)}{2} = \frac{a}{b}$$

#### /2 Sintaxis

Esta función también tiene una sintaxis dedicada: usá una barra para convertir las expresiones vecinas en una fracción. Se pueden agrupar varios átomos en una única expresión con paréntesis redondos de agrupación. Esos paréntesis se quitan de la salida, pero podés anidar varios para forzar que se muestren.

#### /3 Parámetros

```
math.frac(
  content,
  content,
  style: str,
) → content
```

##### /3.1 **num** `content` *Obligatorio* *Posicional*

El numerador de la fracción.

##### /3.2 **denom** `content` *Obligatorio* *Posicional*

El denominador de la fracción.

##### /3.3 **style** `str` *Configurable con set* Por defecto: `"vertical"`

Cómo debe disponerse la fracción.

Example · Estilos

```
$ frac(x, y, style: "vertical") $
$ frac(x, y, style: "skewed") $
$ frac(x, y, style: "horizontal") $
```

$$\frac{x}{y}$$

$$x/y$$

$$x/y$$

Example · Establecer el valor por defecto

```
#set math.frac(style: "skewed")
$ a / b $
```

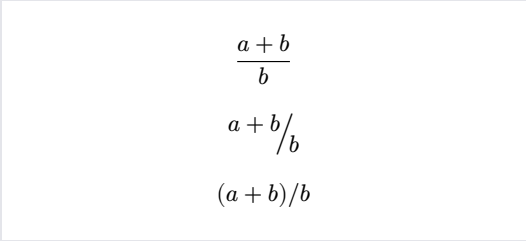
$$a/b$$

Example · Manejo de los paréntesis de agrupación

```
// Grouping parentheses are removed.
#set math.frac(style: "vertical")
$ (a + b) / b $

// Grouping parentheses are removed.
#set math.frac(style: "skewed")
$ (a + b) / b $

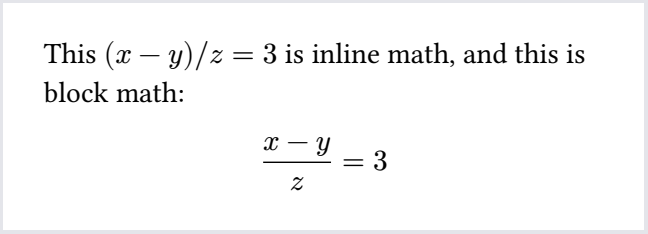
// Grouping parentheses are retained.
#set math.frac(style: "horizontal")
$ (a + b) / b $
```



Example · Distintos estilos en ecuaciones en línea y en bloque

```
// This changes the style for inline equations only.
#show math.equation.where(block: false): set math.frac(style: "horizontal")

This $(x-y)/z = 3$ is inline math, and this is block math:
$ (x-y)/z = 3 $
```

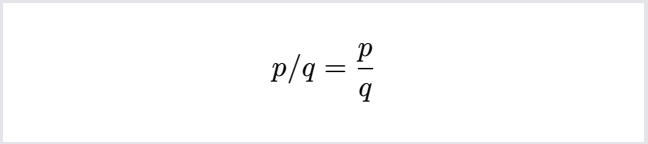


Example · Usar la convención de tipo LaTeX

```
// Change the default style.
#set math.frac(style: "horizontal")
// Define a shorthand with the original style.
#let frac = math.frac.with(style: "vertical")

$ p/q = frac(p, q) $

// The shadowed definition can still be accessed.
#assert.eq($p/q$, $std.math.frac(p, q)$)
```



| Variante | Detalles |
|---|---|
| `"vertical"` | Numerador y denominador apilados con una barra. |
| `"skewed"` | Numerador y denominador separados por una barra. |
| `"horizontal"` | Numerador y denominador en línea, y los paréntesis no se absorben. |

## 3.8.20 Izquierda/Derecha

Emparejamiento de delimitadores.

La función `lr` te permite emparejar dos delimitadores y escalarlos con el contenido que encierran. Si bien esto también ocurre automáticamente con los delimitadores que se emparejan sintácticamente, `lr` te permite emparejar dos delimitadores arbitrarios y controlar exactamente su tamaño. Además de la función `lr`, Typst ofrece algunas funciones más que crean pares de delimitadores para valores absolutos, redondeados hacia arriba y hacia abajo, así como para normas.

Para evitar que Typst empareje un delimitador y, por lo tanto, lo escale automáticamente, escapalo con una barra invertida. Para desactivar por completo el escalado automático, usá `set math.lr(size: 1em)`.

### /1 Example

```
$ [a, b/2] $
$ lr(]sum_(x=1)^n], size: #50%) x $
$ abs((x + y) / 2) $
$ \{ (x / y) \} $
#set math.lr(size: 1em)
$ { (a / b), a, b in (0; 1/2] } $
```

$$\left[a, \frac{b}{2}\right]$$

$$\left]\sum_{x=1}^{n}\right] x$$

$$\left|\frac{x+y}{2}\right|$$

$$\{\left(\frac{x}{y}\right)\}$$

$$\{(\frac{a}{b}), a, b \in (0; \frac{1}{2}]\}$$

### /2 Funciones

#### /2.1 `lr` Elemento Desde: siempre

Escala delimitadores.

Si bien los delimitadores que forman pareja se escalan por defecto, esto se puede usar para escalar delimitadores sin pareja y para controlar con más precisión el escalado de los delimitadores.

```
math.lr(
  size: relative,
  content,
) → content
```

##### /2.1.1 `size` `relative` *Configurable con set* Por defecto: `100% + 0pt`

El tamaño de los paréntesis, relativo a la altura del contenido que envuelven.

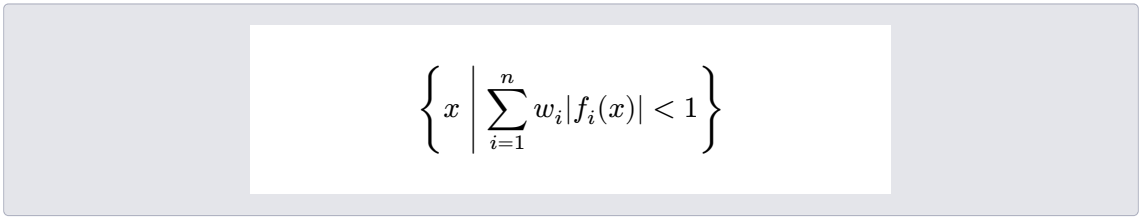
##### /2.1.2 `body` `content` *Obligatorio* *Posicional*

El contenido delimitado, incluidos los delimitadores.

#### /2.2 `mid` Elemento Desde: 0.10.0

Escala delimitadores verticalmente hasta el grupo `lr()` circundante más cercano.

```
$ { x mid(|) sum_(i=1)^n w_i abs(f_i (x)) < 1 } $
```



```
math.mid(content) → content
```

/2.2.1 **body** `content` *Obligatorio* *Posicional*

El contenido que se escala.

### /2.3 `abs` Desde: siempre

Toma el valor absoluto de una expresión.

```
$ abs(x/2) $
```



```
math.abs(
  size: relative,
  content,
) → content
```

/2.3.1 **size** `relative`

El tamaño de los paréntesis, relativo a la altura del contenido que envuelven.

Por defecto: el valor actual de `lr.size`[3.8.20/2.1.1].

/2.3.2 **body** `content` *Obligatorio* *Posicional*

La expresión de la que se toma el valor absoluto.

### /2.4 `norm` Desde: siempre

Toma la norma de una expresión.

```
$ norm(x/2) $
```



```
math.norm(
  size: relative,
  content,
) → content
```

/2.4.1 **size** `relative`

El tamaño de los paréntesis, relativo a la altura del contenido que envuelven.

Por defecto: el valor actual de `lr.size`[3.8.20/2.1.1].

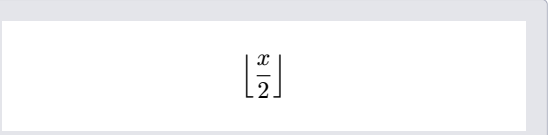
/2.4.2 **body** `content` *Obligatorio* *Posicional*

La expresión de la que se toma la norma.

### /2.5 `floor` Desde: siempre

Redondea una expresión hacia abajo.

```
$ floor(x/2) $
```



```
math.floor(
  size: relative,
  content,
) → content
```

#### /2.5.1 `size` `relative`

El tamaño de los paréntesis, relativo a la altura del contenido que envuelven.

Por defecto: el valor actual de `lr.size`[3.8.20/2.1.1].

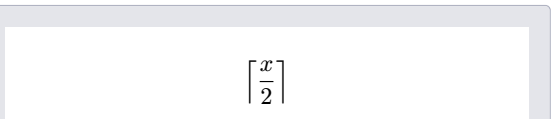
#### /2.5.2 **body** `content` *Obligatorio* *Posicional*

La expresión que se redondea hacia abajo.

### /2.6 `ceil` Desde: siempre

Redondea una expresión hacia arriba.

```
$ ceil(x/2) $
```



```
math.ceil(
  size: relative,
  content,
) → content
```

#### /2.6.1 `size` `relative`

El tamaño de los paréntesis, relativo a la altura del contenido que envuelven.

Por defecto: el valor actual de `lr.size`[3.8.20/2.1.1].

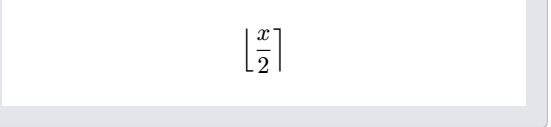
#### /2.6.2 **body** `content` *Obligatorio* *Posicional*

La expresión que se redondea hacia arriba.

### /2.7 `round` Desde: siempre

Redondea una expresión.

```
$ round(x/2) $
```



```
math.round(
  size: relative,
  content,
) → content
```

/2.7.1 **`size`** `relative`

El tamaño de los paréntesis, relativo a la altura del contenido que envuelven.

Por defecto: el valor actual de `lr.size`[3.8.20/2.1.1].

/2.7.2 **`body`** `content` *Obligatorio* *Posicional*

La expresión que se redondea.

### 3.8.21 `mat` Elemento Desde: siempre

Una matriz.

Los elementos de una fila deben separarse con comas, mientras que las filas en sí deben separarse con puntos y coma. La sintaxis con punto y coma combina en un array los argumentos anteriores separados por comas. También podés usar esta sintaxis especial de las llamadas a funciones matemáticas para definir funciones personalizadas que toman datos 2D.

El contenido de las celdas se puede alinear con el parámetro `align`[3.8.21/2.2], o bien el contenido de las celdas que están en la misma fila se puede alinear con el símbolo &.

#### /1 Ejemplo

```
$ mat(
  1, 2, ..., 10;
  2, 2, ..., 10;
  dots.v, dots.v, dots.down, dots.v;
  10, 10, ..., 10;
) $
```

$$\begin{pmatrix} 1 & 2 & \dots & 10 \\ 2 & 2 & \dots & 10 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 10 & \dots & 10 \end{pmatrix}$$

#### /2 Parámetros

```
math.mat(
  delim: none str symbol array,
  align: alignment,
  augment: none int dictionary,
  gap: relative,
  row-gap: relative,
  column-gap: relative,
  .. array,
) → content
```

##### /2.1 `delim` `none` or `str` or `symbol` or `array` *Configurable con set* Por defecto: `("(", ")")`

El delimitador que se usa.

Puede ser un único carácter que especifica el delimitador izquierdo, en cuyo caso se infiere el delimitador derecho. De lo contrario, puede ser un array que contenga un delimitador izquierdo y uno derecho.

```
#set math.mat(delim: "[")
$ mat(1, 2; 3, 4) $
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

##### /2.2 `align` `alignment` *Configurable con set* Por defecto: `center`

La alineación horizontal que debe tener cada celda.

```
#set math.mat(align: right)
$ mat(-1, 1, 1; 1, -1, 1; 1, 1, -1) $
```

$$\begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$$

##### /2.3 `augment` `none` or `int` or `dictionary` *Configurable con set* Por defecto: none

Dibuja líneas de aumento en una matriz.

- none: No se dibuja ninguna línea.
- Un único número: Se dibuja una línea de aumento vertical después del número de columna especificado. Los números negativos cuentan desde el final.
- Un diccionario: con un diccionario, se pueden dibujar varias líneas de aumento tanto horizontales como verticales. Además, se puede establecer el estilo de las líneas. El diccionario puede contener las siguientes claves:
  - `hline`: Los desplazamientos en los que se deben dibujar las líneas horizontales. Por ejemplo, un desplazamiento de 2 dibujaría una línea horizontal después de la segunda fila de la matriz. Acepta un entero para una sola línea o un array de enteros para varias líneas. Igual que para un único número, los números negativos cuentan desde el final.
  - `vline`: Los desplazamientos en los que se deben dibujar las líneas verticales. Por ejemplo, un desplazamiento de 2 dibujaría una línea vertical después de la segunda columna de la matriz. Acepta un entero para una sola línea o un array de enteros para varias líneas. Igual que para un único número, los números negativos cuentan desde el final.
  - `stroke`: Cómo trazar[3.11.12] la línea. Si se establece en `auto`, toma un grosor de 0,05 em y remates de línea cuadrados.

Example · Uso básico

```
$ mat(1, 0, 1; 0, 1, 2; augment: #2) $
// Equivalent to:
$ mat(1, 0, 1; 0, 1, 2; augment: #(-1)) $
```

$$\left(\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 2 \end{array}\right)$$

$$\left(\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 2 \end{array}\right)$$

Example · Personalizar la línea de aumento

```
$ mat(0, 0, 0; 1, 1, 1; augment: #(hline: 1, stroke: 2pt + green)) $
```

$$\left(\begin{array}{ccc} 0 & 0 & 0 \\ \hline 1 & 1 & 1 \end{array}\right)$$

### /2.4 gap `relative`

Por defecto: `0% + 0pt`

El espacio entre filas y columnas.

Es un atajo para establecer `row-gap` y `column-gap` en el mismo valor.

```
#set math.mat(gap: 1em)
$ mat(1, 2; 3, 4) $
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

### /2.5 row-gap `relative` *Configurable con set*

Por defecto: `0% + 0.2em`

El espacio entre filas.

```
#set math.mat(row-gap: 1em)
$ mat(1, 2; 3, 4) $
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

### /2.6 column-gap `relative` *Configurable con set*

Por defecto: `0% + 0.5em`

El espacio entre columnas.

```
#set math.mat(column-gap: 1em)
$ mat(1, 2; 3, 4) $
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

### /2.7 rows `array` *Obligatorio* *Posicional* *Variádico*

Un array de arrays con las filas de la matriz.

```
#let data = ((1, 2, 3), (4, 5, 6))
#let matrix = math.mat(..data)
$ v := matrix $
```

$$v := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

### 3.8.22 `primes` Elemento Desde: 0.11.0

Primas agrupadas.

```
$ a'''_b = a^'''_b $
```

$$a'''_b = a'''_b$$

#### /1 Sintaxis

Esta función tiene una sintaxis dedicada: usá apóstrofes en lugar de primas. Se adjuntarán automáticamente al elemento anterior, moviendo los superíndices al siguiente nivel.

#### /2 Parámetros

```
math.primes(int) → content
```

**/2.1 count** `int` *Obligatorio* *Posicional*

La cantidad de primas agrupadas.

## 3.8.23 Raíces

Raíces cuadradas y no cuadradas.

### /1 Example

```
$ sqrt(3 - 2 sqrt(2)) = sqrt(2) - 1 $
$ root(3, x) $
```

$$\sqrt{3 - 2\sqrt{2}} = \sqrt{2} - 1$$

$$\sqrt[3]{x}$$

### /2 Funciones

#### /2.1 **root** Elemento Desde: siempre

Una raíz general.

```
$ root(3, x) $
```

$$\sqrt[3]{x}$$

```
math.root(
  none content,
  content,
) → content
```

##### /2.1.1 **index** none or content *Posicional* *Configurable con set* Por defecto: none

Qué raíz del radicando se toma.

##### /2.1.2 **radicand** content *Obligatorio* *Posicional*

La expresión de la que se toma la raíz.

#### /2.2 **sqrt** Desde: siempre

Una raíz cuadrada.

```
$ sqrt(3 - 2 sqrt(2)) = sqrt(2) - 1 $
```

$$\sqrt{3 - 2\sqrt{2}} = \sqrt{2} - 1$$

```
math.sqrt(content) → content
```

##### /2.2.1 **radicand** content *Obligatorio* *Posicional*

La expresión de la que se toma la raíz cuadrada.

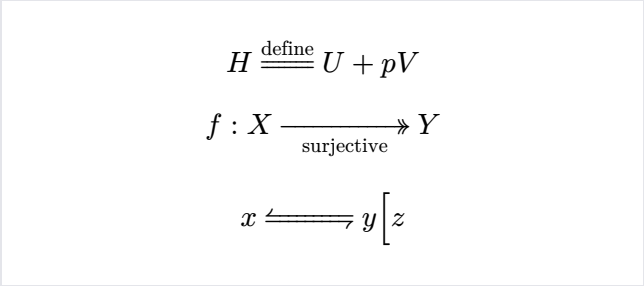
## 3.8.24 `stretch` Elemento Desde: 0.12.0

Estira un glifo.

Esta función también se puede usar para estirar automáticamente la base de un adjunto, de modo que se ajuste a los adjuntos superior e inferior.

Tené en cuenta que solo algunos glifos se pueden estirar y cuáles depende de la fuente matemática que se use. Sin embargo, la mayoría de las fuentes matemáticas son iguales en este aspecto.

```
$ H stretch(=)^"define" U + p V $
$ f : X stretch(->>, size: #150%)_"surjective" Y $
$ x stretch(harpoons.ltrb, size: #3em) y
    stretch(\[, size: #150%) z $
```




### /1 Parámetros

```
math.stretch(
  content,
  size: relative,
) → content
```

#### /1.1 **body** `content` *Obligatorio* *Posicional*

El glifo que se estira.

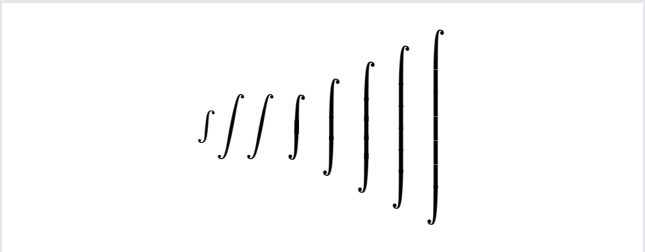
#### /1.2 **size** `relative` *Configurable con set* Por defecto: `100% + 0pt`

El tamaño al que se estira, relativo al tamaño máximo del glifo y sus adjuntos.

Tené en cuenta que el glifo resultante puede no tener exactamente el tamaño deseado. Un glifo estirado puede ser un glifo predefinido o un glifo armado con piezas que proporciona la fuente. Los tamaños posibles pueden no cubrir todo el rango. En el siguiente ejemplo, cuando el parámetro `size` aumenta de `101%` a `200%`, el glifo seleccionado sigue siendo el mismo, por lo que el tamaño real no cambia.

Example · Tamaño de ∫ que crece de forma discontinua

```
#for size in (
  100%, // short
  101%, 200%, // tall
  201%, 300%, 400%, 500%, 600%, // taller
) {
  $stretch(integral, size: #size)$
}
```



## 3.8.25 Tamaños

Estilos de tamaño forzado para las expresiones dentro de las fórmulas.

Estas funciones permiten configurar manualmente el tamaño de los elementos de una ecuación para que se vean como en una ecuación de visualización (display) o en línea, o como si se usaran en una raíz o en subíndices/superíndices.

### /1 Funciones

#### /1.1 `display` Desde: 0.5.0

Estilo de visualización (display) forzado en matemática.

Este es el tamaño normal de las ecuaciones en bloque.

```
$sum_i x_i/2 = display(sum_i x_i/2)$
```

$$\sum_i \frac{x_i}{2} = \sum_i \frac{x_i}{2}$$

```
math.display(
  content,
  cramped: bool,
) → content
```

##### /1.1.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da tamaño.

##### /1.1.2 **cramped** `bool` Por defecto: `false`

Si se impone una restricción de altura para los exponentes, como hacen los subíndices y superíndices normales.

#### /1.2 `inline` Desde: 0.5.0

Estilo en línea (de texto) forzado en matemática.

Este es el tamaño normal de las ecuaciones en línea.

```
$ sum_i x_i/2
    = inline(sum_i x_i/2) $
```

$$\sum_i \frac{x_i}{2} = \sum_i \frac{x_i}{2}$$

```
math.inline(
  content,
  cramped: bool,
) → content
```

##### /1.2.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da tamaño.

##### /1.2.2 **cramped** `bool` Por defecto: `false`

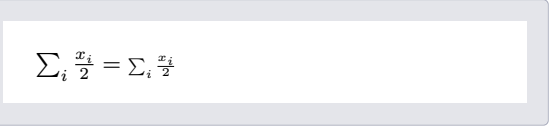
Si se impone una restricción de altura para los exponentes, como hacen los subíndices y superíndices normales.

#### /1.3 `script` Desde: 0.5.0

Estilo de subíndice/superíndice (script) forzado en matemática.

Este es el tamaño más pequeño que se usa en potencias o subíndices y superíndices.

```
$sum_i x_i/2 = script(sum_i x_i/2)$
```



```
math.script(
  content,
  cramped: bool,
) → content
```

/1.3.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da tamaño.

/1.3.2 **cramped** `bool` Por defecto: `true`

Si se impone una restricción de altura para los exponentes, como hacen los subíndices y superíndices normales.

## /1.4 sscript Desde: 0.5.0

Estilo de segundo subíndice/superíndice (script) forzado en matemática.

Este es el tamaño más pequeño, que se usa en los subíndices y superíndices de segundo nivel (script del script).

```
$sum_i x_i/2 = sscript(sum_i x_i/2)$
```



```
math.sscript(
  content,
  cramped: bool,
) → content
```

/1.4.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da tamaño.

/1.4.2 **cramped** `bool` Por defecto: `true`

Si se impone una restricción de altura para los exponentes, como hacen los subíndices y superíndices normales.

## 3.8.26 op Elemento Desde: siempre

Un operador de texto en una ecuación.

### /1 Ejemplo

```
$ tan x = (sin x)/(cos x) $
$ op("custom",
     limits: #true)_(n->oo) n $
```

$$\tan x = \frac{\sin x}{\cos x}$$

$$\operatorname{custom}_{n \to \infty} n$$

### /2 Operadores predefinidos

Typst predefine los operadores `arccos`, `arcsin`, `arctan`, `arg`, `cos`, `cosh`, `cot`, `coth`, `csc`, `csch`, `ctg`, `deg`, `det`, `dim`, `exp`, `gcd`, `lcm`, `hom`, `id`, `im`, `inf`, `ker`, `lg`, `lim`, `liminf`, `limsup`, `ln`, `log`, `max`, `min`, `mod`, `Pr`, `sec`, `sech`, `sin`, `sinc`, `sinh`, `sup`, `tan`, `tanh`, `tg` y `tr`.

### /3 Parámetros

```
math.op(
  content,
  limits: bool,
) → content
```

#### /3.1 text content *Obligatorio* *Posicional*

El texto del operador.

#### /3.2 limits bool *Configurable con set* Por defecto: false

Si el operador debe mostrar los adjuntos como límites en modo de visualización (display).

## 3.8.27 Variantes

Tipografías alternativas dentro de las fórmulas.

Estas funciones son distintas de la función `text`[3.7.12] porque las fuentes matemáticas contienen varias variantes de cada letra.

### /1 Funciones

#### /1.1 `serif` Desde: siempre

Estilo de fuente con serifas (romana) en matemática.

Este ya es el valor por defecto.

```
math.serif(content) → content
```

##### /1.1.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

#### /1.2 `sans` Desde: siempre

Estilo de fuente sin serifas en matemática.

```
$ sans(A B C) $
```

$ sans(A B C) $

```
math.sans(content) → content
```

##### /1.2.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

#### /1.3 `frak` Desde: siempre

Estilo de fuente Fraktur en matemática.

```
$ frak(P) $
```

$ frak(P) $

```
math.frak(content) → content
```

##### /1.3.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

#### /1.4 `mono` Desde: siempre

Estilo de fuente monoespaciada en matemática.

```
$ mono(x + y = z) $
```

$ mono(x + y = z) $

```
math.mono(content) → content
```

##### /1.4.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

### /1.5 **bb** Desde: siempre

Estilo de fuente de pizarra (doble trazo) en matemática.

Para las letras latinas mayúsculas, la pizarra también está disponible mediante símbolos[3.9.2] con la forma NN y RR.

```
$ bb(b) $
$ bb(N) = NN $
$ f: NN -> RR $
```

$$\mathbb{b}$$
$$\mathbb{N} = \mathbb{N}$$
$$f : \mathbb{N} \to \mathbb{R}$$

```
math.bb(content) → content
```

#### /1.5.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

### /1.6 **cal** Desde: siempre

Estilo de fuente caligráfico (chancery) en matemática.

```
Let $cal(P)$ be the set of ...
```

Let $\mathcal{P}$ be the set of ...

Este es el estilo caligráfico/de escritura por defecto de la mayoría de las fuentes matemáticas. Mirá scr[3.8.27/1.7] para más información sobre cómo obtener el otro estilo (roundhand).

```
math.cal(content) → content
```

#### /1.6.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

### /1.7 **scr** Desde: 0.14.0

Estilo de fuente de escritura (roundhand) en matemática.

```
$scr(L)$ is not the set of linear
maps $cal(L)$.
```

$\mathscr{L}$ is not the set of linear maps $\mathcal{L}$.

Hay dos maneras en que las fuentes pueden admitir la diferenciación entre `cal` y `scr`. La primera es usar secuencias de variación de Unicode. Esto funciona de inmediato en Typst, aunque por ahora solo unas pocas fuentes matemáticas lo admiten.

La otra manera es usar características de fuente[3.7.12/2.32]. Por ejemplo, el estilo roundhand podría estar disponible en una fuente mediante la característica del *conjunto estilístico*[3.7.12/2.24] 1 (`ss01`). Para usarlo en Typst, podrías definir tu propia versión de `scr`, como en el siguiente ejemplo.

Example · Recreación con el conjunto estilístico 1

```
#let scr(it) = text(
  stylistic-set: 1,
  $cal(it)$,
)

We establish $cal(P) != scr(P)$.
```

We establish $\mathcal{P} \neq \mathscr{P}$.

```
math.scr(content) → content
```

/1.7.1 **body** `content` *Obligatorio* *Posicional*

El contenido al que se da estilo.

## 3.8.28 `vec` Elemento Desde: siempre

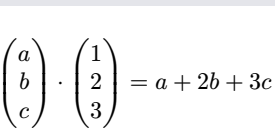
Un vector columna.

El contenido de los elementos del vector se puede alinear con el parámetro `align`[3.8.28/2.2] o con el símbolo `&`.

Esta función sirve para componer las componentes de un vector. Para componer un símbolo que representa un vector, se suelen usar `arrow`[3.8.9] y `bold`[3.8.18/1.3].

### /1 Ejemplo

```
$ vec(a, b, c) dot vec(1, 2, 3)
    = a + 2b + 3c $
```



### /2 Parámetros

```
math.vec(
  delim: none str symbol array,
  align: alignment,
  gap: relative,
  ..content,
) → content
```

#### /2.1 `delim` `none` or `str` or `symbol` or `array` *Configurable con set* Por defecto: `("(", ")")`

El delimitador que se usa.

Puede ser un único carácter que especifica el delimitador izquierdo, en cuyo caso se infiere el delimitador derecho. De lo contrario, puede ser un array que contenga un delimitador izquierdo y uno derecho.

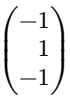
```
#set math.vec(delim: "[")
$ vec(1, 2) $
```



#### /2.2 `align` `alignment` *Configurable con set* Por defecto: `center`

La alineación horizontal que debe tener cada elemento.

```
#set math.vec(align: right)
$ vec(-1, 1, -1) $
```



#### /2.3 `gap` `relative` *Configurable con set* Por defecto: `0% + 0.2em`

El espacio entre elementos.

```
#set math.vec(gap: 1em)
$ vec(1, 2) $
```



/2.4 **children** `content` *Obligatorio* *Posicional* *Variádico*

Los elementos del vector.

## 3.9 Símbolos

Los módulos Símbolos generales[3.9.2] y Emoji[3.9.3] les dan nombres a los símbolos y a los emojis para que sea fácil insertarlos con un teclado normal. Como alternativa, siempre podés ingresar directamente símbolos Unicode en tu texto y en tus fórmulas. Además de los símbolos enumerados a continuación, el modo matemático define `dif` y `Dif`. No son valores de símbolo normales porque también afectan el espaciado y el estilo de fuente.

Podés definir símbolos personalizados con la función constructora del tipo `symbol`[3.5.29].

### 3.9.1 Atajos

Los atajos son secuencias concisas de caracteres que evocan glifos específicos. Los atajos y otras formas de producir símbolos se pueden usar indistintamente. Podés usar distintos conjuntos de atajos en modo matemático y en modo marcado. Algunos atajos, como ~ para un espacio de no separación, producen símbolos que no se imprimen, que se indican con texto provisorio en gris.

Podés desactivar la interpretación de un atajo escapando cualquiera de sus caracteres. Si escapás un solo carácter de un atajo, los caracteres restantes sin escapar pueden formar un atajo distinto.

#### /1 En modo marcado

| | |
|---|---|
| … | `...` |
| *nbsp* | `~` |
| – | `-` |
| – | `--` |
| — | `---` |
| *shy* | `-?` |

#### /2 En modo matemático

| | |
|---|---|
| … | `...` |
| − | `-` |
| ∗ | `*` |
| ∼ | `~` |
| ≠ | `!=` |
| ≔ | `:=` |
| ⩴ | `::=` |
| ≕ | `=:` |
| ≪ | `<<` |
| ⋘ | `<<<` |
| ≫ | `>>` |
| ⋙ | `>>>` |
| ≤ | `<=` |
| ≥ | `>=` |
| → | `->` |
| ⟶ | `-->` |
| ↦ | `\|->` |
| ↣ | `>->` |
| ↠ | `->>` |
| ← | `<-` |
| ⟵ | `<--` |
| ↢ | `<-<` |
| ↞ | `<<-` |
| ↔ | `<->` |
| ⟷ | `<-->` |
| ⇝ | `~>` |
| ⇜ | `<~` |
| ⇝ | `~~>` |
| ⇜ | `<~~` |
| ⇒ | `=>` |
| ⇨ | `\|=>` |
| ⟹ | `==>` |
| ⟸ | `<==` |
| ⇔ | `<=>` |
| ⟺ | `<==>` |
| ⟦ | `[\|` |
| ⟧ | `\|]` |
| ‖ | `\|\|` |

### 3.9.2 Símbolos generales

Símbolos generales con nombre.

Por ejemplo, `#sym.arrow` produce el símbolo →. Dentro de la matemática[3.8], estos símbolos se pueden usar sin el prefijo `#sym.`.

La `d` de `dx` en una integral se puede escribir como `$dif x$`. Fuera de las fórmulas matemáticas, se puede acceder a `dif` como `math.dif`.

| | |
|---|---|
| 𝔸 | AA |
| A | Alpha |
| 𝔹 | BB |
| B | Beta |
| ℂ | CC |
| X | Chi |
| 𝔻 | DD |
| Δ | Delta |
| Ϝ | Digamma |
| 𝔼 | EE |
| E | Epsilon |
| H | Eta |
| 𝔽 | FF |
| 𝔾 | GG |
| Γ | Gamma |
| ℍ | HH |
| 𝕀 | II |
| ℑ | Im |
| I | Iota |
| 𝕁 | JJ |
| 𝕂 | KK |
| K | Kappa |
| 𝕃 | LL |
| Λ | Lambda |
| 𝕄 | MM |
| M | Mu |
| ℕ | NN |
| N | Nu |
| 𝕆 | OO |
| Ω | Omega |
| ℧ | Omega.inv |
| O | Omicron |
| ℙ | PP |
| Φ | Phi |
| Π | Pi |
| Ψ | Psi |
| ℚ | QQ |
| ℝ | RR |
| ℜ | Re |
| P | Rho |
| 𝕊 | SS |
| Ш | Sha |
| Σ | Sigma |
| 𝕋 | TT |
| T | Tau |
| Θ | Theta |
| ϴ | Theta.alt |
| 𝕌 | UU |
| ϒ | Upsilon |
| 𝕍 | VV |
| 𝕎 | WW |
| 𝕏 | XX |
| Ξ | Xi |
| 𝕐 | YY |
| ℤ | ZZ |
| Z | Zeta |
| ´ | acute |
| ˝ | acute.double |
| ؋ | afghani |
| א | aleph |
| α | alpha |
| & | amp |
| ⅋ | amp.inv |
| ∧ | and |
| ⋀ | and.big |
| ⅄ | and.curly |
| ⟑ | and.dot |
| ⩓ | and.double |
| ∠ | angle |
| ⦟ | angle.acute |
| ∡ | angle.arc |
| ⦛ | angle.arc.rev |
| ⍼ | angle.azimuth |
| ⦠ | angle.obtuse |
| ⦜ | angle.rev |
| ∟ | angle.right |
| ⌟ | angle.right.rev |
| ⊾ | angle.right.arc |
| ⦝ | angle.right.dot |
| ⦜ | angle.right.square |
| ⦞ | angle.s |
| ⟀ | angle.spatial |
| ⦡ | angle.spheric |
| ⦢ | angle.spheric.rev |
| ⦣ | angle.spheric.t |
| Å | angstrom |
| ⍼ | angzarr |
| ≈ | approx |
| ≊ | approx.eq |
| ≉ | approx.not |
| ≋ | approx.hat |
| → | arrow.r |
| ⟼ | arrow.r.long.bar |
| ↦ | arrow.r.bar |
| ↪ | arrow.r.curve |
| ↻ | arrow.r.turn |
| ⇢ | arrow.r.dashed |
| ⇢ | arrow.r.dotted |
| ⇒ | arrow.r.double |
| ⤇ | arrow.r.double.bar |

⟹ arrow.r.double.long
⟾ arrow.r.double.long.bar
⇏ arrow.r.double.not
⇻ arrow.r.double.struck
➡ arrow.r.filled
↪ arrow.r.hook
⟶ arrow.r.long
⟿ arrow.r.long.squiggly
↬ arrow.r.loop
↛ arrow.r.not
⋙ arrow.r.quad
⇝ arrow.r.squiggly
⇥ arrow.r.stop
⇨ arrow.r.stroked
↦ arrow.r.struck
⇸ arrow.r.dstruck
↣ arrow.r.tail
⇾ arrow.r.tail.struck
⇻ arrow.r.tail.dstruck
⥴ arrow.r.tilde
⋙ arrow.r.triple
↠ arrow.r.twohead
⤇ arrow.r.twohead.bar
⤀ arrow.r.twohead.struck
⤅ arrow.r.twohead.dstruck
⤖ arrow.r.twohead.tail
⤔ arrow.r.twohead.tail.struck
⤕ arrow.r.twohead.tail.dstruck
→ arrow.r.open
⇜ arrow.r.wave
← arrow.l
⇤ arrow.l.bar
↵ arrow.l.curve
↩ arrow.l.turn
⇠ arrow.l.dashed
⇠ arrow.l.dotted
⇐ arrow.l.double
⇇ arrow.l.double.bar
⟸ arrow.l.double.long
⟽ arrow.l.double.long.bar
⇍ arrow.l.double.not
⇷ arrow.l.double.struck
⬅ arrow.l.filled
↩ arrow.l.hook
⟵ arrow.l.long
⟻ arrow.l.long.bar
⇜ arrow.l.long.squiggly
↫ arrow.l.loop
↚ arrow.l.not
⋘ arrow.l.quad
↜ arrow.l.squiggly
⇤ arrow.l.stop
⇦ arrow.l.stroked
↤ arrow.l.struck
⇷ arrow.l.dstruck
↢ arrow.l.tail
⇽ arrow.l.tail.struck
⇷ arrow.l.tail.dstruck
⥲ arrow.l.tilde
⇇ arrow.l.triple
↞ arrow.l.twohead
⇺ arrow.l.twohead.bar
⇽ arrow.l.twohead.struck
⇷ arrow.l.twohead.dstruck
↢ arrow.l.twohead.tail
⇽ arrow.l.twohead.tail.struck
⇷ arrow.l.twohead.tail.dstruck
← arrow.l.open
↜ arrow.l.wave
↑ arrow.t
↥ arrow.t.bar
↱ arrow.t.curve
↺ arrow.t.turn
⇡ arrow.t.dashed
⇑ arrow.t.double
⬆ arrow.t.filled
⟰ arrow.t.quad
⤒ arrow.t.stop
⇧ arrow.t.stroked
↟ arrow.t.struck
⇞ arrow.t.dstruck
⟰ arrow.t.triple
↟ arrow.t.twohead
↓ arrow.b
↧ arrow.b.bar
↴ arrow.b.curve
↻ arrow.b.turn
⇣ arrow.b.dashed
⇓ arrow.b.double
⬇ arrow.b.filled
⟱ arrow.b.quad
⤓ arrow.b.stop
⇩ arrow.b.stroked
↡ arrow.b.struck
⇟ arrow.b.dstruck
⟱ arrow.b.triple
↡ arrow.b.twohead
↔ arrow.l.r
⇔ arrow.l.r.double
⟺ arrow.l.r.double.long
⇎ arrow.l.r.double.not
⇼ arrow.l.r.double.struck
⬌ arrow.l.r.filled
⟷ arrow.l.r.long
↮ arrow.l.r.not
⇳ arrow.l.r.stroked
↭ arrow.l.r.struck
⇿ arrow.l.r.dstruck
↭ arrow.l.r.open
↭ arrow.l.r.wave
↕ arrow.t.b
⇕ arrow.t.b.double
⬍ arrow.t.b.filled
⇳ arrow.t.b.stroked
↗ arrow.tr
↗ arrow.tr.bar

arrow.tr.double
arrow.tr.filled
arrow.tr.hook
arrow.tr.stroked
arrow.br
arrow.br.bar
arrow.br.double
arrow.br.filled
arrow.br.hook
arrow.br.stroked
arrow.tl
arrow.tl.bar
arrow.tl.double
arrow.tl.filled
arrow.tl.hook
arrow.tl.stroked
arrow.bl
arrow.bl.bar
arrow.bl.double
arrow.bl.filled
arrow.bl.hook
arrow.bl.stroked
arrow.tl.br
arrow.tr.bl
arrow.ccw
arrow.ccw.half
arrow.cw
arrow.cw.half
arrow.zigzag
arrowhead.t
arrowhead.b
arrows.rr
arrows.ll
arrows.tt
arrows.bb
arrows.lr
arrows.lr.stop
arrows.rl
arrows.rl.long
arrows.tb
arrows.bt
arrows.rrr
arrows.lll
ast.op
ast.op.o
ast.basic
ast.low
ast.double
ast.triple
ast.square
asymp
asymp.not
at
backslash
backslash.o
backslash.not
bag.l
bag.r

baht
ballot
ballot.cross
ballot.check
ballot.check.heavy
bar.v
bar.v.double
bar.v.triple
bar.v.broken
bar.v.o
bar.h
because
beta
beta.alt
beth
bitcoin
bot
bowtie.stroked
bowtie.stroked.big
bowtie.stroked.big.l
bowtie.stroked.big.r
bowtie.stroked.big.l.r
bowtie.filled
bowtie.filled.l
bowtie.filled.r
brace.l
brace.l.stroked
brace.r
brace.r.stroked
brace.t
brace.b
bracket.l
bracket.l.tick.t
bracket.l.tick.b
bracket.l.stroked
bracket.r
bracket.r.tick.t
bracket.r.tick.b
bracket.r.stroked
bracket.t
bracket.b
breve
bullet
bullet.op
bullet.o
bullet.stroked
bullet.stroked.o
bullet.hole
bullet.hyph
bullet.tri
bullet.l
bullet.r
caret
caron
cc
cc.by
cc.nc

⊜ cc.nd
Ⓢ cc.public
Ⓢ cc.sa
Ⓞ cc.zero
₵ cedi
⌈ ceil.l
⌉ ceil.r
¢ cent
✓ checkmark
✓ checkmark.light
✓ checkmark.heavy
♗ chess.bishop.white
♝ chess.bishop.black
♔ chess.king.white
♚ chess.king.black
♘ chess.knight.white
♞ chess.knight.black
♙ chess.pawn.white
♟ chess.pawn.black
♕ chess.queen.white
♛ chess.queen.black
♖ chess.rook.white
♜ chess.rook.black
⟨ chevron.l
❬ chevron.l.curly
❮ chevron.l.dot
⟪ chevron.l.closed
⟪ chevron.l.double
⟩ chevron.r
❭ chevron.r.curly
❯ chevron.r.dot
⟫ chevron.r.closed
⟫ chevron.r.double
χ chi
○ circle.stroked
∘ circle.stroked.tiny
○ circle.stroked.small
◯ circle.stroked.big
● circle.filled
• circle.filled.tiny
● circle.filled.small
⬤ circle.filled.big
◌ circle.dotted
℅ co
: colon
₡ colon.currency
∷ colon.double
⁝ colon.tri
⁝ colon.tri.op
≔ colon.eq
⩴ colon.double.eq
, comma
, comma.inv
, comma.rev
∁ complement
∘ compose
⊚ compose.o

␆ control.ack
␇ control.bel
␈ control.bs
␘ control.can
␍ control.cr
␑ control.dc.one
␒ control.dc.two
␓ control.dc.three
␔ control.dc.four
␡ control.del
␐ control.dle
␙ control.em
␅ control.enq
␄ control.eot
␛ control.esc
␗ control.etb
␃ control.etx
␌ control.ff
␜ control.fs
␝ control.gs
␉ control.ht
␊ control.lf
␕ control.nak
␤ control.nl
␀ control.nul
␞ control.rs
␏ control.si
␎ control.so
␁ control.soh
␠ control.sp
␂ control.stx
␚ control.sub
␖ control.syn
␟ control.us
␋ control.vt
∗ convolve
⊛ convolve.o
℗ copyleft
© copyright
℗ copyright.sound
⌜ corner.l.t
⌞ corner.l.b
⌝ corner.r.t
⌟ corner.r.b
╳ crossmark
✘ crossmark.heavy
¤ currency
† dagger
‡ dagger.double
⁑ dagger.triple
† dagger.l

† dagger.r
‡ dagger.inv
ד daleth
– dash.en
— dash.em
⸺ dash.em.two
⸻ dash.em.three
‒ dash.fig
–: dash.colon
⊖ dash.o
~ dash.wave
〰 dash.wave.double
° degree
δ delta
¨ diaer
∅ diameter
◇ diamond.stroked
⋄ diamond.stroked.small
◇ diamond.stroked.medium
⟐ diamond.stroked.dot
◆ diamond.filled
◆ diamond.filled.medium
◆ diamond.filled.small
⚅ die.six
⚄ die.five
⚃ die.four
⚂ die.three
⚁ die.two
⚀ die.one
Ϝ digamma
÷ div
⨸ div.o
⁒ div.slanted.o
| divides
∤ divides.not
⫮ divides.not.rev
⫲ divides.struck
$ dollar
₫ dong
𓃗 dorome
· dot.op
. dot.basic
· dot.c
⊙ dot.o
⨀ dot.o.big
⊡ dot.square
¨ dot.double
⋯ dot.triple
⁘ dot.quad
ɪ dotless.i
ȷ dotless.j
⋯ dots.h.c
… dots.h
⋮ dots.v
⋱ dots.down
⋰ dots.up
ϯ dram
𓆑 earth
ð earth.alt
ℓ ell
⬭ ellipse.stroked.h
⬯ ellipse.stroked.v
⬬ ellipse.filled.h
⬮ ellipse.filled.v
∅ emptyset
ø emptyset.zero
⃗∅ emptyset.arrow.r
⃖∅ emptyset.arrow.l
⃥∅ emptyset.bar
⃝∅ emptyset.circle
⦸ emptyset.rev
ε epsilon
϶ epsilon.alt
ϵ epsilon.alt.rev
= eq
≛ eq.ast
≛ eq.star
⊜ eq.o
=: eq.colon
≐ eq.dot
≒ eq.dots
≕ eq.dots.down
≔ eq.dots.up
≝ eq.def
≜ eq.delta
≚ eq.equi
≙ eq.est
⩾ eq.gt
⪖ eq.gt.slant
⩽ eq.lt
⪕ eq.lt.slant
≞ eq.m
≠ eq.not
⪯ eq.prec
≟ eq.quest
⪰ eq.succ
≡ eq.triple
≢ eq.triple.not
≣ eq.quad
≡ equiv
≢ equiv.not
≦ equiv.lt
⪇ equiv.lt.slant
≧ equiv.gt
⪈ equiv.gt.slant
⧲ errorbar.square.stroked
⧳ errorbar.square.filled
⧰ errorbar.diamond.stroked
⧱ errorbar.diamond.filled
⧯ errorbar.circle.stroked
⧮ errorbar.circle.filled
η eta
€ euro
! excl
‼ excl.double
¡ excl.inv

⁉ excl.quest
∃ exists
∄ exists.not
⦚ fence.l
⦚ fence.l.double
⦚ fence.r
⦚ fence.r.double
⦙ fence.dotted
♭ flat
𝄫 flat.t
𝄫 flat.b
𝄫 flat.double
𝄫 flat.quarter
⌊ floor.l
⌋ floor.r
❦ floral
❦ floral.l
❦ floral.r
∀ forall
⊩ forces
⊮ forces.not
⌢ frown
ɣ gamma
♀ gender.female
⚢ gender.female.double
⚤ gender.female.male
⚥ gender.intersex
♂ gender.male
⚣ gender.male.double
⚤ gender.male.female
⚦ gender.male.stroke
⚨ gender.male.stroke.t
⚩ gender.male.stroke.r
⚲ gender.neuter
⚧ gender.trans
ג gimel
∇ gradient
` grave
> gt
⊜ gt.o
⩾ gt.dot
⩺ gt.quest
≳ gt.approx
◁ gt.arc
⊵ gt.arc.eq
▷ gt.closed
⊵ gt.closed.eq
⋭ gt.closed.eq.not
⋪ gt.closed.not
≫ gt.double
⋙ gt.double.nested
≥ gt.eq
⩾ gt.eq.slant
⋛ gt.eq.lt
⋛ gt.eq.lt.slant
≱ gt.eq.not
≧ gt.equiv
≧ gt.equiv.slant
≷ gt.lt
≹ gt.lt.not
≩ gt.neq
≩ gt.napprox
≩ gt.nequiv
≯ gt.not
≳ gt.ntilde
≳ gt.tilde
≳ gt.tilde.slant
≴ gt.tilde.not
⋙ gt.triple
⋙ gt.triple.nested
₲ guarani
⇀ harpoon.rt
⥛ harpoon.rt.bar
⥟ harpoon.rt.stop
⇁ harpoon.rb
⥟ harpoon.rb.bar
⥟ harpoon.rb.stop
↼ harpoon.lt
⥚ harpoon.lt.bar
⥞ harpoon.lt.stop
↽ harpoon.lb
⥚ harpoon.lb.bar
⥞ harpoon.lb.stop
↿ harpoon.tl
⥠ harpoon.tl.bar
⥣ harpoon.tl.stop
↾ harpoon.tr
⥠ harpoon.tr.bar
⥣ harpoon.tr.stop
⇂ harpoon.bl
⥡ harpoon.bl.bar
⥥ harpoon.bl.stop
⇃ harpoon.br
⥡ harpoon.br.bar
⥥ harpoon.br.stop
⥊ harpoon.lt.rt
⥎ harpoon.lb.rb
⥊ harpoon.lb.rt
⥎ harpoon.lt.rb
⥑ harpoon.tl.bl
⥤ harpoon.tr.br
⥑ harpoon.tl.br
⥤ harpoon.tr.bl
⇉ harpoons.rtrb
⥯ harpoons.blbr
⥣ harpoons.bltr
⥧ harpoons.lbrb
⇇ harpoons.ltlb
⥎ harpoons.ltrb
⥊ harpoons.ltrt
⇄ harpoons.rblb
⇌ harpoons.rtlb
⥋ harpoons.rtlt
⥯ harpoons.tlbr
⥣ harpoons.tltr
# hash

^ hat
⬡ hexa.stroked
⬢ hexa.filled
⧖ hourglass.stroked
⧗ hourglass.filled
₴ hryvnia
- hyph
- hyph.minus
- hyph.nobreak
· hyph.point
*shy* hyph.soft
⊷ image
∈ in
∉ in.not
∋ in.rev
∌ in.rev.not
∍ in.rev.small
∈ in.small
∞ infinity
⧜ infinity.bar
∾ infinity.incomplete
⧝ infinity.tie
∫ integral
⨍ integral.arrow.hook
⨑ integral.ccw
∮ integral.cont
∳ integral.cont.ccw
∲ integral.cont.cw
∱ integral.cw
⨏ integral.dash
⨐ integral.dash.double
∬ integral.double
⨌ integral.quad
⨙ integral.inter
⨎ integral.slash
⨖ integral.square
∯ integral.surf
⨘ integral.times
∭ integral.triple
∰ integral.union
⨋ integral.vol
∩ inter
⩎ inter.serif
⩙ inter.and
⋂ inter.big
⩀ inter.dot
⋒ inter.double
⊓ inter.sq
⩏ inter.sq.serif
⨅ inter.sq.big
⩋ inter.sq.double
⫴ interleave
⫼ interleave.big
⫳ interleave.struck
⁉ interrobang
⸘ interrobang.inv
ι iota
ɩ iota.inv
♃ jupiter
ϰ kappa
ϗ kappa.alt
⌥ keyboard.alt
⇪ keyboard.capslock
⍐ keyboard.capslock.pedestal
⌧ keyboard.clear
⌘ keyboard.cmd
⏎ keyboard.enter
↵ keyboard.enter.alt
⌫ keyboard.erase.l
⌦ keyboard.erase.r
⎋ keyboard.esc
⌥ keyboard.option
⇟ keyboard.pagedown
⎘ keyboard.pagedown.alt
⇞ keyboard.pageup
⎗ keyboard.pageup.alt
⇧ keyboard.shift
␣ keyboard.space
⇆ keyboard.tab
⇥ keyboard.tab.r
₭ kip
λ lambda
Δ laplace
₾ lari
⩾ lat
≥ lat.eq
⪾ lat.eq.slant
₺ lira
◊ lozenge.stroked
⋄ lozenge.stroked.small
◊ lozenge.stroked.medium
⧫ lozenge.filled
⬩ lozenge.filled.small
⧫ lozenge.filled.medium
*lrm* lrm
< lt
⦵ lt.o
⩿ lt.dot
⩻ lt.quest
≲ lt.approx
◅ lt.arc
⧀ lt.arc.eq
◁ lt.closed
⊴ lt.closed.eq
⋬ lt.closed.eq.not
⋪ lt.closed.not
≪ lt.double
⪡ lt.double.nested
≤ lt.eq
⩽ lt.eq.slant
⋚ lt.eq.gt
⪋ lt.eq.gt.slant
≰ lt.eq.not
≦ lt.equiv
⪇ lt.equiv.slant
≶ lt.gt

≯ lt.gt.not
≨ lt.neq
≩ lt.napprox
≨ lt.nequiv
≮ lt.not
≴ lt.ntilde
≲ lt.tilde
≲ lt.tilde.slant
≴ lt.tilde.not
⋘ lt.triple
⋘ lt.triple.nested
¯ macron
✠ maltese
₼ manat
↤ mapsfrom
⟻ mapsfrom.long
↦ mapsto
⟼ mapsto.long
♂ mars
☿ mercury
− minus
⊖ minus.o
∸ minus.dot
∓ minus.plus
⊟ minus.square
≂ minus.tilde
⨺ minus.triangle
⩪ miny
⊨ models
μ mu
⊸ multimap
⧟ multimap.double
∫ mustache.l
⎫ mustache.r
∇ nabla
₦ naira
♮ natural
𝄮 natural.t
𝄯 natural.b
♆ neptune
♆ neptune.alt
¬ not
⌙ not.inv
⌐ not.rev
♫ note.up
♫ note.down
𝅝 note.whole
𝅗𝅥 note.half
𝅘𝅥 note.quarter
♩ note.quarter.alt
𝅘𝅥𝅮 note.eighth
♪ note.eighth.alt
♫ note.eighth.beamed
𝅘𝅥𝅯 note.sixteenth
♫ note.sixteenth.beamed
𝆕 note.grace
𝆔 note.grace.slash

ø nothing
ø nothing.zero
⃗∅ nothing.arrow.r
⃖∅ nothing.arrow.l
⃥∅ nothing.bar
⃝∅ nothing.circle
⦸ nothing.rev
ν nu
№ numero
ω omega
ο omicron
∞ oo
∨ or
⋁ or.big
⋎ or.curly
⩒ or.dot
⩔ or.double
ª ordinal.feminine
º ordinal.masculine
⊶ original
∥ parallel
⫴ parallel.struck
⫼ parallel.o
⋕ parallel.eq
⧣ parallel.equiv
∦ parallel.not
⫽ parallel.slanted.eq
⫽ parallel.slanted.eq.tilde
⫻ parallel.slanted.equiv
⫲ parallel.tilde
▱ parallelogram.stroked
▰ parallelogram.filled
( paren.l
⟮ paren.l.flat
⦇ paren.l.closed
⦇ paren.l.stroked
) paren.r
⟯ paren.r.flat
⦈ paren.r.closed
⦈ paren.r.stroked
⏜ paren.t
⏝ paren.b
∂ partial
$ pataca
℘ pee
⬠ penta.stroked
⬟ penta.filled
% percent
‰ permille
‱ permyriad
⊥ perp
⦩ perp.o
$ peso
₱ peso.philippine
φ phi
ϕ phi.alt
π pi
ϖ pi.alt

¶ pilcrow
⁋ pilcrow.rev
ℏ planck
+ plus
⊕ plus.o
⨭ plus.o.l
⨮ plus.o.r
⟴ plus.o.arrow
⨁ plus.o.big
∔ plus.dot
⧺ plus.double
± plus.minus
⊞ plus.square
⧋ plus.triangle
⧻ plus.triple
⨣ plus.hat
£ pound
⏻ power.standby
⏽ power.on
⏾ power.off
⍟ power.on.off
⏾ power.sleep
≺ prec
≾ prec.approx
≼ ~~prec.curly.eq~~
⋠ ~~prec.curly.eq.not~~
⪻ prec.double
⪷ prec.eq
≼ prec.eq.slant
⋠ prec.eq.slant.not
⩱ prec.equiv
⪹ prec.napprox
⪵ prec.neq
⪱ prec.nequiv
⊀ prec.not
⪹ prec.ntilde
≾ prec.tilde
′ prime
‵ prime.rev
″ prime.double
‶ prime.double.rev
‴ prime.triple
‷ prime.triple.rev
⁗ prime.quad
∏ product
∐ product.co
∝ prop
ψ psi
■ qed
? quest
⁇ quest.double
⁈ quest.excl
¿ quest.inv
" quote.double
' quote.single
“ quote.l.double
‘ quote.l.single
” quote.r.double
’ quote.r.single
« quote.chevron.l.double
‹ quote.chevron.l.single
» quote.chevron.r.double
› quote.chevron.r.single
‟ quote.high.double
‛ quote.high.single
„ quote.low.double
‚ quote.low.single
∶ ratio
▭ rect.stroked.h
▯ rect.stroked.v
▬ rect.filled.h
▮ rect.filled.v
※ refmark
𝄻 rest.whole
𝄺 rest.multiple
𝄩 rest.multiple.measure
𝄼 rest.half
𝄽 rest.quarter
𝄾 rest.eighth
𝄿 rest.sixteenth
ρ rho
ϱ rho.alt
𐡙 riel
₽ riyal
rlm rlm
₽ ruble
₹ rupee.indian
Rs rupee.generic
𑓐 rupee.tamil
𞋿 rupee.wancho
♄ saturn
§ section
; semi
⁏ semi.inv
; semi.rev
ш sha
♯ sharp
𝆯 sharp.t
𝆰 sharp.b
𝆱 sharp.double
𝆲 sharp.quarter
₪ shekel
⟮ shell.l
⟬ shell.l.stroked
❨ shell.l.filled
⟯ shell.r
⟭ shell.r.stroked
❩ shell.r.filled
⏠ shell.t
⏡ shell.b
σ sigma
ς sigma.alt
/ slash
⊘ slash.o
⫽ slash.double

⫽ slash.triple
/ slash.big
※ smash
⌣ smile
⋖ smt
⪕ smt.eq
⪗ smt.eq.slant
⊜ som
␣ space
*nbsp* space.nobreak
*nnbsp* space.nobreak.narrow
*ensp* space.en
*emsp* space.quad
*⅓emsp* space.third
*¼emsp* space.quarter
*⅙emsp* space.sixth
*mmsp* space.med
*numsp* space.fig
*puncsp* space.punct
*thinsp* space.thin
*hairsp* space.hair
␣ ~~spacebar~~
□ square.stroked
▫ square.stroked.tiny
▫ square.stroked.small
□ square.stroked.medium
□ square.stroked.big
⬚ square.stroked.dotted
▢ square.stroked.rounded
■ square.filled
▪ square.filled.tiny
■ square.filled.small
■ square.filled.medium
■ square.filled.big
⋆ star.op
☆ star.stroked
★ star.filled
⊂ subset
⫅ subset.approx
⊏ subset.closed
⊑ subset.closed.eq
⪽ subset.dot
⫇ subset.double
⊆ subset.eq
⫅ subset.eq.dot
⊈ subset.eq.not
⊑ subset.eq.sq
⋢ subset.eq.sq.not
⫋ subset.equiv
⊊ subset.neq
⫋ subset.nequiv
⊄ subset.not
⪿ subset.plus
⊏ subset.sq
⋤ subset.sq.neq
⫅ subset.tilde
⫃ subset.times
≻ succ
⪸ succ.approx
~~⋡~~ ~~succ.curly.eq~~
~~⋡~~ ~~succ.curly.eq.not~~
⪼ succ.double
⪰ succ.eq
≽ succ.eq.slant
⋡ succ.eq.slant.not
⪴ succ.equiv
⪺ succ.napprox
⪶ succ.neq
⪶ succ.nequiv
⊁ succ.not
⪹ succ.ntilde
≿ succ.tilde
♣ suit.club.filled
♧ suit.club.stroked
♦ suit.diamond.filled
♢ suit.diamond.stroked
♥ suit.heart.filled
♡ suit.heart.stroked
♠ suit.spade.filled
♤ suit.spade.stroked
∑ sum
∯ sum.integral
⊙ sun
⊃ supset
⫆ supset.approx
⊐ supset.closed
⊒ supset.closed.eq
⪾ supset.dot
⫈ supset.double
⊇ supset.eq
⫆ supset.eq.dot
⊉ supset.eq.not
⊒ supset.eq.sq
⋣ supset.eq.sq.not
⫌ supset.equiv
⊋ supset.neq
⫌ supset.nequiv
⊅ supset.not
⫀ supset.plus
⊐ supset.sq
⋥ supset.sq.neq
⫆ supset.tilde
⫄ supset.times
⊢ tack.r
⊬ tack.r.not
⟝ tack.r.long
⊦ tack.r.short
⊩ tack.r.double
⊮ tack.r.double.not
⫣ tack.r.triple
⊨ tack.rr
⊭ tack.rr.not
⊧ tack.rr.short
⫤ tack.rr.double
⫤ tack.rr.double.not
⊫ tack.rrr

⊣ tack.l
⟞ tack.l.long
⊦ tack.l.short
⫤ tack.l.double
⫣ tack.ll
⫣ tack.ll.double
⊥ tack.t
⟘ tack.t.big
⊥ tack.t.short
⫫ tack.t.double.short
⫪ tack.tt
⊤ tack.b
⟙ tack.b.big
⊤ tack.b.short
⫟ tack.b.double.short
⫪ tack.bb
⫞ tack.l.r
𝍠 taka
𝍡 taman
τ tau
𝍫 tenge
∴ therefore
θ theta
ϑ theta.alt
∼ tilde.op
– tilde.basic
∾ tilde.dot
≃ tilde.eq
≄ tilde.eq.not
≂ tilde.eq.rev
≅ tilde.equiv
≇ tilde.equiv.not
≆ tilde.nequiv
≲ tilde.lt
≴ tilde.lt.slant
≳ tilde.gt
≵ tilde.gt.slant
≁ tilde.not
∽ tilde.rev
≌ tilde.rev.equiv
≋ tilde.triple
× times
⨯ times.big
⊗ times.o
⊗ times.o.l
⊗ times.o.r
⊗ times.o.hat
⨂ times.o.big
⋇ times.div
⋋ times.three.l
⋌ times.three.r
⋉ times.l
⋊ times.r
⊠ times.square
⟁ times.triangle
⨺ tiny
₮ togrog
⊤ top
™ trademark
® trademark.registered
℠ trademark.service
㎥ trademark.mc
㎤ trademark.md
△ triangle.stroked.t
▽ triangle.stroked.b
▷ triangle.stroked.r
◁ triangle.stroked.l
◺ triangle.stroked.bl
◿ triangle.stroked.br
◸ triangle.stroked.tl
◹ triangle.stroked.tr
▵ triangle.stroked.small.t
▿ triangle.stroked.small.b
▹ triangle.stroked.small.r
◃ triangle.stroked.small.l
🛆 triangle.stroked.rounded
⟁ triangle.stroked.nested
◬ triangle.stroked.dot
▲ triangle.filled.t
▼ triangle.filled.b
▶ triangle.filled.r
◀ triangle.filled.l
◣ triangle.filled.bl
◢ triangle.filled.br
◤ triangle.filled.tl
◥ triangle.filled.tr
▲ triangle.filled.small.t
▼ triangle.filled.small.b
▶ triangle.filled.small.r
◀ triangle.filled.small.l
_ underscore
∪ union
⋓ union.serif
⊎ union.arrow
⋃ union.big
⊍ union.dot
⨃ union.dot.big
⋓ union.double
⊎ union.minus
⋓ union.or
⊎ union.plus
⨄ union.plus.big
⊔ union.sq
⊔ union.sq.serif
⨆ union.sq.big
⊔ union.sq.double
υ upsilon
♅ uranus
⛢ uranus.alt
♀ venus
∖ without
*wjoin* wj
₩ won
≀ wreath
ξ xi
⊕ xor

| | |
|---|---|
| ⊕ | xor.big |
| ¥ | yen |
| ¥ | yuan |
| o | zero |
| o | zero.slashed |
| ζ | zeta |
| *zwj* | zwj |
| *zwnj* | zwnj |
| *zwsp* | zws |

### 3.9.3 Emoji

Emojis con nombre.

Por ejemplo, `#emoji.face` produce el emoji 😀. Si usás con frecuencia ciertos emojis, también podés importarlos del módulo `emoji` (`#import emoji: face`) para usarlos sin el prefijo `emoji.`.

| | |
|---|---|
| * | `ast` |
| # | `hash` |
| 😀 | `face.grin` |
| 😃 | `face.open` |
| 😄 | `face.smile` |
| 😁 | `face.beam` |
| 😆 | `face.squint` |
| 😅 | `face.smile.sweat` |
| 🤣 | `face.rofl` |
| 😂 | `face.joy` |
| 🙂 | `face.smile.slight` |
| 🙃 | `face.inv` |
| 🫠 | `face.melt` |
| 😉 | `face.wink` |
| 😊 | `face.happy` |
| 😇 | `face.halo` |
| 🥰 | `face.hearts` |
| 😍 | `face.heart` |
| 🤩 | `face.stars` |
| 😘 | `face.kiss.heart` |
| 😗 | `face.kiss` |
| ☺ | `face.friendly` |
| 😚 | `face.kiss.blush` |
| 😙 | `face.kiss.smile` |
| 🥲 | `face.smile.tear` |
| 😋 | `face.lick` |
| 😛 | `face.tongue` |
| 😜 | `face.tongue.wink` |
| 🤪 | `face.goofy` |
| 😝 | `face.tongue.squint` |
| 🤑 | `face.money` |
| 🤗 | `face.hug` |
| 🤭 | `face.cover` |
| 🤔 | `face.speak.not` |
| 🫣 | `face.peek` |
| 🤫 | `face.shush` |
| 🤔 | `face.think` |
| 🫡 | `face.salute` |
| 🤐 | `face.zip` |
| 🤨 | `face.skeptic` |
| 😐 | `face.neutral` |
| 😑 | `face.straight` |
| 😶 | `face.blank` |
| 🫥 | `face.dotted` |
| 😏 | `face.smirk` |
| 😒 | `face.meh` |
| 🙄 | `face.eyeroll` |
| 😬 | `face.teeth` |
| 🤥 | `face.lie` |
| 😯 | `face.shaking` |
| 😌 | `face.relief` |
| 😔 | `face.sad` |
| 😪 | `face.sleepy` |
| 🤤 | `face.drool` |
| 😴 | `face.sleep` |
| 😫 | `face.tired` |
| 😷 | `face.mask` |
| 🤒 | `face.fever` |
| 🤕 | `face.bandage` |
| 🤢 | `face.nausea` |
| 🤮 | `face.vomit` |
| 🤧 | `face.sneeze` |
| 🥵 | `face.heat` |
| 🥶 | `face.cold` |
| 🥴 | `face.woozy` |
| 😵 | `face.dizzy` |
| 🤯 | `face.explode` |
| 🤠 | `face.cowboy` |
| 🥳 | `face.party` |
| 🥸 | `face.disguise` |
| 😎 | `face.cool` |
| 🤓 | `face.nerd` |
| 🧐 | `face.monocle` |
| 😦 | `face.unhappy` |
| 😕 | `face.diagonal` |
| 😟 | `face.worry` |
| 🙁 | `face.frown.slight` |
| ☹ | `face.frown` |
| 😮 | `face.wow` |
| 😯 | `face.surprise` |
| 😲 | `face.astonish` |
| 😳 | `face.flush` |
| 🫨 | `face.distorted` |
| 🥹 | `face.plead` |
| 🥹 | `face.tear.withheld` |
| 😟 | `face.concern` |
| 😧 | `face.anguish` |
| 😨 | `face.fear` |
| 😰 | `face.fear.sweat` |
| 🥲 | `face.tear.relief` |
| 😢 | `face.tear` |
| 😭 | `face.cry` |
| 😱 | `face.shock` |
| 😣 | `face.suffer` |
| 😤 | `face.frust` |
| 😞 | `face.down` |
| 😓 | `face.down.sweat` |
| 😩 | `face.weary` |

😣 face.distress
🥱 face.yawn
😤 face.triumph
😡 face.angry.red
😠 face.angry
🤬 face.symbols
😈 face.devil.smile
👿 face.devil.frown
💀 skull
☠️ skull.bones
💩 poo
🤡 face.clown
👹 oni
👺 tengu
👻 ghost
👽 alien
👾 alien.monster
🤖 robot
😺 cat.face.smile
😸 cat.face.laugh
😹 cat.face.joy
😻 cat.face.heart
😼 cat.face.smirk
😽 cat.face.kiss
🙀 cat.face.shock
😿 cat.face.cry
😾 cat.face.angry
🙈 monkey.see.not
🙉 monkey.hear.not
🙊 monkey.speak.not
💌 letter.love
💘 heart.arrow
💝 heart.ribbon
💖 heart.spark
💗 heart.grow
💓 heart.beat
💞 heart.revolve
💕 heart.double
💟 heart.box
❣️ heart.excl
💔 heart.broken
❤️ heart
🩷 heart.pink
🧡 heart.orange
💛 heart.yellow
💚 heart.green
💙 heart.blue
🩵 heart.lightblue
💜 heart.purple
🤎 heart.brown
🖤 heart.black
🩶 heart.gray
🤍 heart.white
💋 kissmark

💯 hundred
💢 anger
🗯️ fightcloud
💥 explosion
💫 star.arc
💦 drops
💨 cloud.dust
🕳️ hole
💬 bubble.speech.r
🗨️ bubble.speech.l
💭 bubble.anger.r
🗯️ bubble.thought
💤 zzz
👋 hand.wave
🤚 hand.raised.alt
🖐️ hand.splay
✋ hand.raised
🖖 hand.part
👉 hand.r
👈 hand.l
👎 hand.b
🫱 hand.t
🫸 hand.pushing.l
🫷 hand.pushing.r
👌 hand.ok
🤌 fingers.pinch
🤏 hand.pinch
✌️ hand.peace
🤞 fingers.cross
🫰 fingers.snap
🤟 hand.love
🤘 hand.rock
🤙 hand.call
👈 finger.l
👉 finger.r
👆 finger.t
🖕 finger.m
👇 finger.b
☝️ finger.t.alt
🫵 finger.front
👍 thumb.up
👎 thumb.down
✊ fist.raised
👊 fist.front
🤛 fist.l
🤜 fist.r
👏 hands.clap
🙌 hands.raised
🫶 hands.heart
👐 hands.open
🤲 hands.palms
🤝 hands.shake
🙏 hands.folded
✍️ hand.write

- nails.polish
- arm.selfie
- arm.muscle
- arm.mech
- leg.mech
- leg
- foot
- ear
- ear.aid
- nose
- brain
- heart.real
- lungs
- tooth
- bone
- eyes
- eye
- tongue
- mouth
- mouth.bite
- baby
- child
- boy
- girl
- person
- person.blonde
- man
- person.beard
- woman
- person.old
- man.old
- woman.old
- person.frown
- person.angry
- person.no
- person.ok
- person.sassy
- person.raise
- person.deaf
- person.bow
- person.facepalm
- person.shrug
- policeofficer
- detective
- guard.man
- ninja
- builder
- person.crown
- man.crown
- woman.crown
- man.turban
- man.guapimao
- person.hijab
- man.tuxedo
- bride
- woman.pregnant
- man.pregnant
- person.pregnant
- feeding.breast
- baby.angel
- santa.man
- santa.woman
- superhero
- supervillain
- mage
- fairy
- vampire
- merperson
- elf
- genie
- zombie
- troll
- bigfoot
- person.massage
- haircut
- pedestrian
- person.stand
- person.kneel
- running
- dancing.ballet
- dancing.woman
- dancing.man
- man.levitate
- dancing.bunny
- ♀ dancing.bunny.women
- ♂ dancing.bunny.men
- person.steam
- climbing
- fencing
- horse.race
- skiing
- snowboarding
- golfing
- surfing
- boat.row
- swimming
- basketball
- weightlifting
- bicyclist
- bicyclist.mountain
- gymnastics
- wrestling
- waterpolo
- handball
- juggling
- person.lotus
- bathtub
- bed.person

- handholding
- handholding.woman.woman
- handholding.woman.man
- handholding.man.man
- kiss
- couple
- silhouette.speak
- silhouette
- silhouette.double
- silhouette.hug
- family
- prints.foot
- fingerprint
- unknown
- monkey.face
- monkey
- gorilla
- orangutan
- dog.face
- dog
- dog.guide
- dog.poodle
- wolf
- fox
- raccoon
- cat.face
- cat
- lion
- tiger.face
- tiger
- leopard
- horse.face
- moose
- donkey
- horse
- unicorn
- zebra
- deer
- bison
- cow.face
- ox
- buffalo.water
- cow
- pig.face
- pig
- boar
- pig.nose
- ram
- sheep
- goat
- camel.dromedar
- camel
- llama
- giraffe
- elephant
- mammoth
- rhino
- hippo
- mouse.face
- mouse
- rat
- hamster.face
- rabbit.face
- rabbit
- chipmunk
- beaver
- hedgehog
- bat
- bear
- koala
- panda
- sloth
- otter
- skunk
- kangaroo
- badger
- prints.paw
- turkey
- chicken
- chicken.male
- chicken.baby.egg
- chicken.baby.head
- chicken.baby
- bird
- penguin
- dove.peace
- eagle
- duck
- swan
- owl
- dodo
- feather
- flamingo
- peacock
- parrot
- wing
- goose
- frog.face
- crocodile
- turtle
- lizard
- snake
- dragon.face
- dragon
- dino.pod
- dino.rex
- whale.spout
- whale

- dolphin
- orca
- seal
- fish
- fish.tropical
- blowfish
- shark
- octopus
- shell.spiral
- coral
- jellyfish
- crab
- lobster
- shrimp
- squid
- oyster
- snail
- butterfly
- bug
- ant
- bee
- beetle
- beetle.lady
- cricket
- cockroach
- spider
- spiderweb
- scorpion
- mosquito
- fly
- worm
- microbe
- bouquet
- flower.pink
- flower.white
- flower.lotus
- rosette
- flower.rose
- flower.wilted
- flower.hibiscus
- flower.sun
- flower.yellow
- flower.tulip
- flower.hyacinth
- seedling
- plant
- tree.evergreen
- tree.deciduous
- tree.palm
- cactus
- rice.ear
- leaf.herb
- leaf.clover.three
- leaf.clover.four
- leaf.maple
- leaf.fall
- leaf.wind
- nest.empty
- nest.eggs
- mushroom
- tree.leafless
- grapes
- melon
- watermelon
- tangerine
- lemon
- banana
- pineapple
- mango
- apple.red
- apple.green
- pear
- peach
- cherries
- strawberry
- blueberries
- kiwi
- tomato
- olive
- coconut
- avocado
- aubergine
- potato
- carrot
- corn
- pepper.hot
- pepper
- cucumber
- spinach
- broccoli
- garlic
- onion
- peanuts
- beans
- chestnut
- ginger
- peapod
- beet
- bread
- croissant
- baguette
- flatbread
- pretzel
- bagel
- pancakes
- waffle
- cheese
- meat.bone

- chicken.leg
- meat
- bacon
- burger
- fries
- pizza
- hotdog
- sandwich
- taco
- burrito
- tamale
- kebab
- falafel
- egg
- cooking
- paella
- soup
- fondue
- bowl.spoon
- salad
- popcorn
- butter
- salt
- can
- bento
- rice.cracker
- rice.onigiri
- rice
- curry
- bowl.steam
- spaghetti
- potato.sweet
- skewer.oden
- sushi
- shrimp.fried
- cake.fish
- cake.moon
- skewer.dango
- dumpling
- cookie.fortune
- takeout
- icecream.soft
- icecream.shaved
- icecream
- donut
- cookie
- cake
- cake.slice
- cupcake
- pie
- chocolate
- candy
- lollipop
- custard
- honey
- babybottle
- glass.milk
- coffee
- teapot
- teacup
- sake
- champagne
- wine
- cocktail.martini
- cocktail.tropical
- beer
- beer.clink
- glass.clink
- glass.tumbler
- glass.pour
- cup.straw
- bubbletea
- juice
- mate
- ice
- chopsticks
- plate.cutlery
- cutlery
- spoon
- knife
- jar
- amphora
- globe.eu.af
- globe.am
- globe.as.au
- globe.meridian
- map.world
- map.jp
- compass
- mountain.snow
- mountain
- landslide
- volcano
- mountain.fuji
- camping
- beach.umbrella
- desert
- beach.palm
- park
- stadium
- museum
- crane
- brick
- rock
- wood
- hut
- house.multiple
- house.derelict

house
house.garden
office
post.jp
post.eu
hospital
bank
hotel
hotel.love
store.small
school
store.big
factory
castle.jp
castle.eu
church.love
tower.tokyo
liberty
church
mosque
temple
synagogue
shrine
kaaba
fountain
tent
bridge.fog
city.night
city
mountain.sunrise
sunrise
city.dusk
city.sunset
bridge.night
hotspring
horse.carousel
slide
ferriswheel
rollercoaster
barberpole
circus
train.steam
train.car
train.speed
train.speed.bullet
train
train.metro
train.light
train.stop
train.tram
train.mono
train.mountain
train.tram.car
bus
bus.front
bus.trolley
bus.small
ambulance
fireengine
car.police
car.police.front
taxi
taxi.front
car
car.front
car.suv
car.pickup
truck
truck.trailer
tractor
car.racing
motorcycle
scooter.motor
wheelchair
wheelchair.motor
car.rickshaw
bike
scooter
skateboard
shoe.roller
bus.stop
motorway
railway
oil
fuelpump
wheel
siren
trafficlight.h
trafficlight.v
sign.stop
construction
anchor
ringbuoy
boat.sail
boat.canoe
boat.speed
ship.cruise
ship.ferry
boat.motor
ship
airplane
airplane.small
airplane.takeoff
airplane.landing
parachute
seat
helicopter
train.suspend

🚠 cablecar
🚡 cablecar.small
🛰️ satellite
🚀 rocket
🛸 ufo
🛎️ bell.ding
🧳 luggage
⌛ hourglass
⏳ hourglass.flow
⌚ watch
⏰ clock.alarm
⏱️ clock.stop
⏲️ clock.timer
🕰️ clock.old
🕛 clock.twelve
🕧 clock.twelve.thirty
🕐 clock.one
🕜 clock.one.thirty
🕑 clock.two
🕝 clock.two.thirty
🕒 clock.three
🕞 clock.three.thirty
🕓 clock.four
🕟 clock.four.thirty
🕔 clock.five
🕠 clock.five.thirty
🕕 clock.six
🕡 clock.six.thirty
🕖 clock.seven
🕢 clock.seven.thirty
🕗 clock.eight
🕣 clock.eight.thirty
🕘 clock.nine
🕤 clock.nine.thirty
🕙 clock.ten
🕥 clock.ten.thirty
🕚 clock.eleven
🕦 clock.eleven.thirty
🌑 moon.new
🌒 moon.wax.one
🌓 moon.wax.two
🌔 moon.wax.three
🌕 moon.full
🌖 moon.wane.one
🌗 moon.wane.two
🌘 moon.wane.three
🌙 moon.crescent
🌚 moon.new.face
🌛 moon.wax.two.face
🌜 moon.wane.three.face
🌡️ thermometer
☀️ sun
🌝 moon.full.face
🌞 sun.face

🪐 planet
⭐ star
🌟 star.glow
🌠 star.shoot
🌌 milkyway
☁️ cloud
⛅ cloud.sun
⛈️ cloud.storm
🌤️ sun.cloud
🌥️ cloud.sun.hidden
🌦️ cloud.sun.rain
🌧️ cloud.rain
🌨️ cloud.snow
🌩️ cloud.thunder
🌪️ tornado
🌫️ fog
🌬️ wind
🌀 cyclone
🌈 rainbow
🌂 umbrella.closed
☂️ umbrella.open
☔ umbrella.rain
⛱️ umbrella.sun
⚡ lightning
❄️ snowflake
☃️ snowman.snow
⛄ snowman
☄️ comet
🔥 fire
💧 drop
🌊 wave
🎃 pumpkin.lantern
🎄 tree.xmas
🎆 fireworks
🎇 sparkler
🧨 firecracker
✨ sparkles
🎈 balloon
🎉 party
🎊 confetti
🎋 tanabata
🎍 kadomatsu
🎏 ningyo
🎐 koinobori
🪔 windchime
🎑 tsukimi
🧧 hongbao
🎀 ribbon
🎁 present
🎗️ ribbon.remind
🎟️ ticket.event
🎫 ticket.travel
🎖️ medal.military
🏆 trophy

- 🏅 medal.sports
- 🥇 medal.first
- 🥈 medal.second
- 🥉 medal.third
- ⚽ football
- ⚾ baseball
- 🥎 softball
- 🏀 basketball.ball
- 🏐 volleyball
- 🏈 football.am
- 🏉 rugby
- 🎾 tennis
- 🥏 frisbee
- 🎳 bowling
- 🏏 cricketbat
- 🏑 hockey
- 🏒 icehockey
- 🥍 lacrosse
- 🏓 pingpong
- 🏸 badminton
- 🥊 boxing
- 🥋 martialarts
- 🥅 goal
- ⛳ flag.golf
- ⛸ shoe.ice
- 🎣 fishing
- 🤿 diving
- 🎽 shirt.sports
- ⛷ shoe.ski
- 🛷 sled
- 🥌 curling
- 🎯 darts
- 🪀 yoyo
- 🪁 kite
- 🔫 pistol
- 🎱 billiards
- 🔮 crystal
- 🪄 wand
- 🎮 controller
- 🕹 joystick
- 🎰 slots
- 🎲 die
- 🧩 jigsaw
- 🧸 teddy
- 🪅 pinata
- 🪩 discoball
- 🪆 matryoshka
- ♠ suit.spade
- ♥ suit.heart
- ♦ suit.diamond
- ♣ suit.club
- ♟ chess
- 🃏 playingcard.joker
- 🀄 mahjong.dragon.red
- 🎴 playingcard.flower
- 🎭 masks
- 🖼 painting
- 🎨 palette
- 🧵 thread
- 🪡 needle
- 🧶 yarn
- 🪢 knot
- 👓 glasses
- 🕶 glasses.sun
- 🥽 goggles
- 🥼 coat.lab
- 🦺 safetyvest
- 👔 necktie
- 👕 shirt.t
- 👖 jeans
- 🧣 scarf
- 🧤 gloves
- 🧥 coat
- 🧦 socks
- 👗 dress
- 👘 dress.kimono
- 🥻 dress.sari
- 🩱 swimsuit
- 🩲 briefs
- 🩳 shorts
- 👙 bikini
- 👚 blouse
- 🪭 handfan
- 👛 purse
- 👜 handbag
- 👝 pouch
- 🛍 shopping
- 🎒 backpack
- 🩴 shoe.thong
- 👞 shoe
- 👟 shoe.sneaker
- 🥾 shoe.hike
- 🥿 shoe.flat
- 👠 shoe.heel
- 👡 shoe.sandal.heel
- 🩰 shoe.ballet
- 👢 shoe.tall
- 🪮 hairpick
- 👑 crown
- 👒 hat.ribbon
- 🎩 hat.top
- 🎓 mortarboard
- 🧢 basecap
- 🪖 helmet.military
- ⛑ helmet.cross
- 📿 beads
- 💄 lipstick
- 💍 ring

- gem
- speaker.not
- speaker
- speaker.wave
- speaker.waves
- megaphone
- megaphone.simple
- horn.postal
- bell
- bell.not
- musicalscore
- notes
- notes.triple
- microphone.studio
- slider
- mixer
- microphone
- headphone
- radio
- saxophone
- trumpet
- trombone
- accordion
- guitar
- piano
- violin
- banjo
- drum
- drum.big
- maracas
- flute
- harp
- phone
- phone.arrow
- phone.classic
- phone.receiver
- pager
- fax
- battery
- battery.low
- powerplug
- laptop
- computer
- printer
- keyboard
- computermouse
- trackball
- disc.mini
- floppy
- disc.cd
- disc.dvd
- abacus
- camera.movie
- film
- projector
- clapperboard
- tv
- camera
- camera.flash
- camera.video
- cassette
- magnify.l
- magnify.r
- candle
- lightbulb
- flashlight
- izakaya
- lamp.diya
- notebook.deco
- book.red
- book.open
- book.green
- book.blue
- book.orange
- books
- notebook
- book.spiral
- page.curl
- scroll
- page
- newspaper
- newspaper.rolled
- pages.tabs
- bookmark
- label
- coin
- money.bag
- treasure
- money.yen
- money.dollar
- money.euro
- money.pound
- money.wings
- card.credit
- receipt
- chart.yen.up
- mail
- email
- tray.mail
- mail.arrow
- tray.outbox
- tray.inbox
- package
- mailbox.closed.full
- mailbox.closed.empty
- mailbox.open.full
- mailbox.open.empty
- postbox

🗳️ ballotbox
✏️ pencil
🖋️ pen.fountain
🖊️ pen.ball
🖌️ brush
🖍️ crayon
📝 page.pencil
💼 briefcase
📁 folder
📂 folder.open
🗂️ filedividers
📅 calendar
📆 calendar.tearoff
🗒️ notepad
🗓️ calendar.spiral
📇 cardindex
📈 chart.up
📉 chart.down
📊 chart.bar
📋 clipboard
📌 pin
📍 pin.round
📎 clip
🖇️ clips
📏 ruler
📐 ruler.triangle
✂️ scissors
🗃️ filebox
🗄️ cabinet.file
🗑️ bin
🔒 lock
🔓 lock.open
🔏 lock.pen
🔐 lock.key
🔑 key
🗝️ key.old
🔨 hammer
🪓 axe
⛏️ pick
⚒️ hammer.pick
🛠️ hammer.wrench
🗡️ knife.dagger
⚔️ swords
💣 bomb
🪃 boomerang
🏹 bow
🛡️ shield
🪚 saw
🔧 wrench
🪛 screwdriver
🔩 bolt
⚙️ gear
🗜️ clamp
⚖️ scales

🦯 cane
🔗 chain
⛓️ chains
🪝 hook
🧰 toolbox
🧲 magnet
🪜 ladder
🪏 shovel
⚗️ alembic
🧪 testtube
🧫 petri
🧬 helix
🔬 microscope
🔭 telescope
📡 satdish
💉 syringe
🩸 blood
💊 pill
🩹 plaster
🩼 crutch
🩺 stethoscope
🩻 xray
🚪 door
🛗 elevator
🪞 mirror
🪟 window
🛏️ bed
🛋️ couch
🪑 chair
🚽 toilet
🪠 plunger
🚿 shower
🛁 bathtub.foam
🪤 mousetrap
🪒 razor
🧴 lotion
🧷 safetypin
🧹 broom
🧺 basket
🧻 toiletpaper
🪣 bucket
🧼 soap
🫧 bubbles
🪥 toothbrush
🧽 sponge
🧯 extinguisher
🛒 cart
🚬 cigarette
⚰️ coffin
🪦 tombstone
🏺 urn
🧿 nazar
🪬 hamsa
🗿 moyai

placard
card.id
atm
litter
tap
wheelchair.box
man.box
woman.box
restroom
baby.box
wc
bordercontrol
customs
baggageclaim
leftluggage
warning
sign.crossing
noentry
prohibited
bike.not
cigarette.not
litter.not
tap.not
pedestrian.not
phone.not
eighteen.not
radioactive
biohazard
arrow.t.filled
arrow.tr
arrow.r.filled
arrow.br
arrow.b.filled
arrow.bl
arrow.l.filled
arrow.tl
arrow.t.b
arrow.l.r
arrow.l.hook
arrow.r.hook
arrow.t.curve
arrow.b.curve
playback.repeat.v
arrows.cycle
arrow.l.back
arrow.l.end
arrow.l.r.on
arrow.r.soon
arrow.t.top
faith.worship
atom
faith.om
faith.judaism
faith.dharma
faith.yinyang
faith.christ
faith.orthodox
faith.islam
faith.peace
faith.menorah
faith.star.dot
faith.khanda
zodiac.aries
zodiac.taurus
zodiac.gemini
zodiac.cancer
zodiac.leo
zodiac.virgo
zodiac.libra
zodiac.scorpio
zodiac.sagit
zodiac.capri
zodiac.aquarius
zodiac.pisces
zodiac.ophi
playback.shuffle
playback.repeat
playback.repeat.once
triangle.r
playback.forward
playback.skip
playback.toggle
triangle.l
playback.rewind
playback.restart
triangle.t
playback.up
triangle.b
playback.down
playback.pause
playback.stop
playback.record
playback.eject
camera.movie.box
brightness.low
brightness.high
phone.signal
wireless
phone.vibrate
phone.off
transgender
excl.double
excl.quest
quest
quest.white
excl.white
excl
dash.wave.double
forex
dollar

- aesculapius
- recycling
- fleur
- trident
- namebadge
- shoshinsha
- circle.stroked
- checkmark.box
- ballot.check
- checkmark.heavy
- crossmark
- crossmark.box
- partalteration
- ast.box
- star.box
- sparkle.box
- copyright
- reg
- tm
- splatter
- ten
- ABCD
- abcd
- numbers
- symbols
- abc
- a
- ab
- b
- cl
- cool
- free
- info
- id
- metro
- new
- ng
- o
- ok
- parking
- sos
- up
- vs
- koko
- sa
- gachi
- yuryo
- yubi
- toku
- kachi
- muryo
- kon
- ka
- shin
- go
- kara
- shuku
- hi
- yo
- circle.red
- circle.orange
- circle.yellow
- circle.green
- circle.blue
- circle.purple
- circle.brown
- circle.black
- circle.white
- square.red
- square.orange
- square.yellow
- square.green
- square.blue
- square.purple
- square.brown
- square.black
- square.white
- square.black.medium
- square.white.medium
- square.black.small
- square.white.small
- square.black.tiny
- square.white.tiny
- diamond.orange
- diamond.blue
- diamond.orange.small
- diamond.blue.small
- triangle.t.red
- triangle.b.red
- diamond.dot
- button.radio
- button.alt
- button
- flag.goal
- flag.red
- flags.jp.crossed
- flag.black
- flag.white

# 3.10 Diseño

Disponer elementos en la página de distintas maneras.

Combinando funciones de diseño, podés crear diseños complejos y automáticos.

## Definiciones


3.10.1 Align Aligns content horizontally and vertically. 350
3.10.2 Alignment Where to align something along an axis. 352
3.10.3 Angle An angle describing a rotation. 355
3.10.4 Block A block-level container. 356
3.10.5 Box An inline-level container that sizes content. 360
3.10.6 Column Break Forces a column break. 363
3.10.7 Columns Separates a region into multiple equally sized columns. 364
3.10.8 Direction The four directions into which content can be laid out. 367
3.10.9 Fraction Defines how the remaining space in a layout is distributed. 371
3.10.10 Grid Arranges content in a grid. 372
3.10.11 Hide Hides content without affecting layout. 385
3.10.12 Layout Provides access to the current outer container's (or page's, if none) dimensions (width and height). 386
3.10.13 Length A size or distance, possibly expressed with contextual units. 388
3.10.14 Measure Measures the layouted size of content. 390
3.10.15 Move Moves content without affecting layout. 392
3.10.16 Padding Adds spacing around content. 393
3.10.17 Page Layouts its child onto one or multiple pages. 395
3.10.18 Page Break A manual page break. 406
3.10.19 Place Places content relatively to its parent container. 407
3.10.20 Ratio A ratio of a whole. 411
3.10.21 Relative Length A length in relation to some known length. 412
3.10.22 Repeat Repeats content to the available space. 414
3.10.23 Rotate Rotates content without affecting layout. 415
3.10.24 Scale Scales content without affecting layout. 417
3.10.25 Skew Skews content. 419
3.10.26 Spacing (H) Inserts horizontal spacing into a paragraph. 421
3.10.27 Spacing (V) Inserts vertical spacing into a flow of blocks. 423
3.10.28 Stack Arranges content and spacing horizontally or vertically. 424

### 3.10.1 `align` Elemento Desde: siempre

Alinea contenido horizontal y verticalmente.

#### /1 Ejemplo

Empecemos centrando nuestro contenido horizontalmente:

```
#set page(height: 120pt)
#set align(center)

Centered text, a sight to see \
In perfect balance, visually \
Not left nor right, it stands alone \
A work of art, a visual throne
```

Centered text, a sight to see
In perfect balance, visually
Not left nor right, it stands alone
A work of art, a visual throne

Para centrar algo verticalmente, usá la alineación *horizon*:

```
#set page(height: 120pt)
#set align(horizon)

Vertically centered, \
the stage had entered, \
a new paragraph.
```

Vertically centered,
the stage had entered,
a new paragraph.

#### /2 Combinar alineaciones

Podés combinar dos alineaciones con el operador +. Además, apliquemos esto solo a un fragmento de contenido usando la forma de función en lugar de una regla set:

```
#set page(height: 120pt)
Though left in the beginning ...

#align(right + bottom)[
  ... they were right in the end, \
  and with addition had gotten, \
  the paragraph to the bottom!
]
```

Though left in the beginning ...

... they were right in the end,
and with addition had gotten,
the paragraph to the bottom!

#### /3 Alineación anidada

Podés usar alineaciones distintas para los contenedores de diseño y los elementos que hay dentro de ellos. De esta manera, podés crear diseños intrincados:

```
#align(center, block[
  #set align(left)
  Though centered together \
  alone \
  we \
  are \
  left.
])
```

Though centered together
alone
we
are
left.

## /4 Alineación dentro de la misma línea

La función `align` realiza una alineación de bloque y, por lo tanto, siempre interrumpe el párrafo actual. Para tener una alineación distinta en partes de la misma línea, tenés que usar en su lugar espaciado fraccionario[3.10.26]:

```
Start #h(1fr) End
```

Start End

## /5 Parámetros

```
align(
  alignment,
  content,
) → content
```

### /5.1 alignment `alignment` *Posicional* *Configurable con set* Por defecto: `start + top`

La alineación[3.10.2] en ambos ejes.

```
#set page(height: 6cm)
#set text(lang: "ar")

مثال
#align(
  end + horizon,
  rect(inset: 12pt)[ركن]
)
```

مثال

ركن

### /5.2 body `content` *Obligatorio* *Posicional*

El contenido que se alinea.

### 3.10.2 `alignment` Desde: siempre

Dónde alinear algo a lo largo de un eje.

Los valores posibles son:
- Alineaciones horizontales: `start`[3.10.2/3.1], `end`[3.10.2/3.2], `left`[3.10.2/3.3], `center`[3.10.2/3.4] y `right`[3.10.2/3.5].
- Alineaciones verticales: `top`[3.10.2/3.6], `horizon`[3.10.2/3.7] y `bottom`[3.10.2/3.8].
- Alineaciones 2D[3.10.2/1], como `center + horizon`.

Estos valores están disponibles globalmente y también en el ámbito del tipo alignment, así que podés escribir cualquiera de las dos formas siguientes:

```
#align(center)[Hi]
#align(alignment.center)[Hi]
```

Hi

Hi

#### /1 Alineaciones 2D

Para alinear en ambos ejes al mismo tiempo, sumá las dos alineaciones con el operador +. Por ejemplo, `top + right` alinea el contenido a la esquina superior derecha.

```
#set page(height: 3cm)
#align(center + bottom)[Hi]
```

Hi

#### /2 Campos

Los campos `x` e `y` contienen los componentes horizontal y vertical de la alineación, respectivamente (como otro `alignment` más). Pueden ser none.

```
#(top + right).x \
#left.x \
#left.y (none)
```

right
left
(none)

#### /3 Constantes

##### /3.1 `start` `alignment` Desde: siempre

Alineación horizontal hacia el inicio[3.10.8/2.5] de la dirección del texto[3.7.12/2.19].

```
#set text(dir: ltr)
#align(start, emoji.arrow.r)

#set text(dir: rtl)
#align(start, emoji.arrow.l)
```

➡

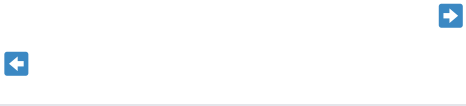
⬅

##### /3.2 `end` `alignment` Desde: siempre

Alineación horizontal hacia el final[3.10.8/2.6] de la dirección del texto[3.7.12/2.19].

```
#set text(dir: ltr)
#align(end, emoji.arrow.r)

#set text(dir: rtl)
#align(end, emoji.arrow.l)
```



/3.3 **left** alignment Desde: siempre

Alineación horizontal a la izquierda.

```
#align(left)[Left]
```

Left

/3.4 **center** alignment Desde: siempre

Alineación horizontal centrada.

```
#align(center)[Center]
```

Center

/3.5 **right** alignment Desde: siempre

Alineación horizontal a la derecha.

```
#align(right)[Right]
```

Right

/3.6 **top** alignment Desde: siempre

Alineación vertical superior.

```
#set page(height: 3cm)
#align(top)[Top]
```

Top

/3.7 **horizon** alignment Desde: siempre

Alineación vertical central.

```
#set page(height: 3cm)
#align(horizon)[Horizon]
```

Horizon

/3.8 **bottom** alignment Desde: siempre

Alineación vertical inferior.

```
#set page(height: 3cm)
#align(bottom)[Bottom]
```

Bottom

## /4 Funciones

### /4.1 axis Desde: 0.7.0

El eje al que pertenece esta alineación.
- "horizontal" para start, left, center, right y end
- "vertical" para top, horizon y bottom
- none para alineaciones bidimensionales

```
#left.axis() \
#bottom.axis()
```

horizontal
vertical

```
self.axis() → none str
```

### /4.2 inv Desde: 0.7.0

La alineación inversa.

```
#top.inv() \
#left.inv() \
#center.inv() \
#(left + bottom).inv()
```

bottom
right
center
right + top

```
self.inv() → alignment
```

### 3.10.3 `angle` Desde: siempre

Un ángulo que describe una rotación.

Typst admite las siguientes unidades angulares:

- Grados: `180deg`
- Radianes: `3.14rad`

## /1 Ejemplo

```
#rotate(10deg)[Hello there!]
```

Hello there!

## /2 Funciones

### /2.1 `rad` Desde: 0.7.0

Convierte este ángulo a radianes.

```
self.rad() → float
```

### /2.2 `deg` Desde: 0.7.0

Convierte este ángulo a grados.

```
self.deg() → float
```

## 3.10.4 `block` Elemento Desde: siempre

Un contenedor de bloque.

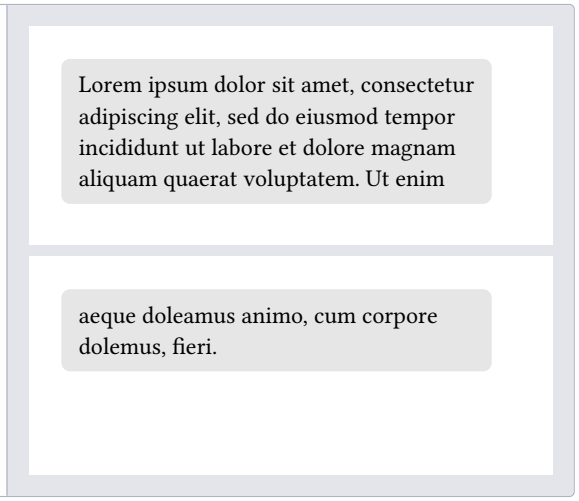
Un contenedor así se puede usar para separar contenido, para darle tamaño[3.10.4/2.1] o recortarlo[3.10.4/2.12], o para darle un fondo[3.10.4/2.4] o un borde[3.10.4/2.5].

Los bloques también son la forma principal de controlar si el texto pasa a formar parte de un párrafo o no. Mirá la documentación de párrafos[3.6.14/2] para más detalles.

### /1 Ejemplos

Con un bloque, podés darle un fondo al contenido y, aun así, permitir que se corte entre varias páginas.

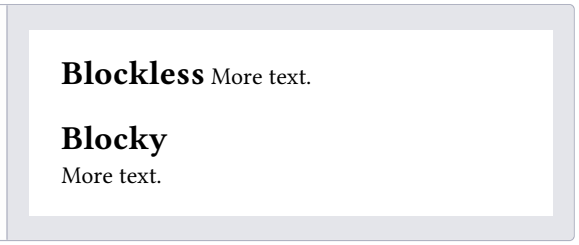
```
#set page(height: 100pt)
#block(
  fill: luma(230),
  inset: 8pt,
  radius: 4pt,
  lorem(30),
)
```




Los bloques también son útiles para forzar a que elementos que de otro modo serían en línea pasen a ser de nivel de bloque, especialmente al escribir reglas show.

```
#show heading: it => it.body
= Blockless
More text.

#show heading: it => block(it.body)
= Blocky
More text.
```




### /2 Parámetros

```
block(
  width: auto relative,
  height: auto relative fraction,
  breakable: bool,
  fill: none color gradient tiling,
  stroke: none length color gradient stroke tiling dictionary,
  radius: relative dictionary,
  inset: relative dictionary,
  outset: relative dictionary,
  spacing: auto relative fraction,
  above: auto relative fraction,
  below: auto relative fraction,
  clip: bool,
  sticky: bool,
```

```
  none content,
) → content
```

/2.1 **width** `auto` or `relative` *Configurable con set* Por defecto: `auto`

El ancho del bloque.

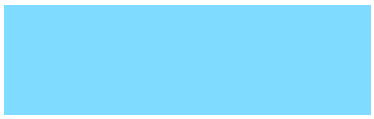
```
#set align(center)
#block(
  width: 60%,
  inset: 8pt,
  fill: silver,
  lorem(10),
)
```

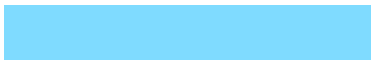
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

/2.2 **height** `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

La altura del bloque. Cuando la altura es mayor que el espacio restante de una página y `breakable`[3.10.4/2.3] es `true`, el bloque continuará en la página siguiente con la altura restante.

```
#set page(height: 80pt)
#set align(center)
#block(
  width: 80%,
  height: 150%,
  fill: aqua,
)
```





/2.3 **breakable** `bool` *Configurable con set* Por defecto: `true`

Si el bloque se puede cortar y continuar en la página siguiente.

```
#set page(height: 80pt)
The following block will
jump to its own page.
#block(
  breakable: false,
  lorem(15),
)
```

The following block will jump to its own page.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

/2.4 **fill** `none` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `none`

El color de fondo del bloque. Mirá la documentación del rectángulo[3.11.10/2.3] para más detalles.

/2.5 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary`
*Configurable con set* Por defecto: `(:)`

El color del borde del bloque. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

/2.6 **radius** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se redondean las esquinas del bloque. Mirá la documentación del rectángulo[3.11.10/2.5] para más detalles.

/2.7 **inset** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto relleno se agrega al contenido del bloque. Mirá la documentación de box[3.10.5/2.7] para más detalles.

/2.8 **outset** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se expande el tamaño del bloque sin afectar el diseño. Mirá la documentación de box[3.10.5/2.8] para más detalles.

/2.9 **spacing** `auto` or `relative` or `fraction` Por defecto: `1.2em`

El espaciado alrededor del bloque. Cuando es `auto`, hereda el `spacing`[3.6.14/4.2] del párrafo.

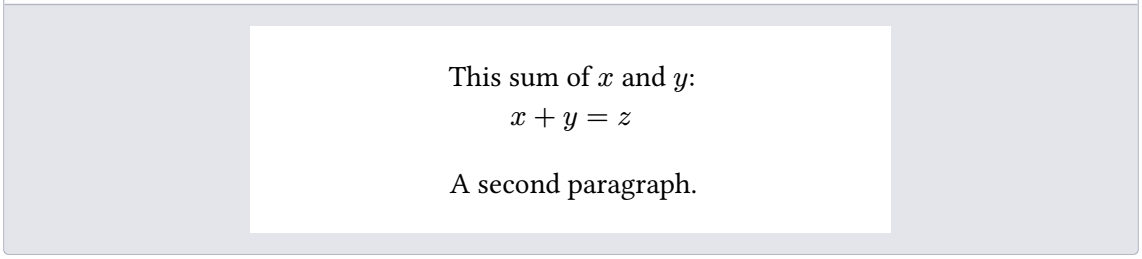
Entre dos bloques adyacentes, gana el mayor entre el espaciado `below` del primer bloque y el `above` del segundo. Además, el espaciado de bloques tiene prioridad sobre el `spacing`[3.6.14/4.2] del párrafo.

Tené en cuenta que esto es solo un atajo para establecer `above` y `below` en el mismo valor. Como los valores de `above` y `below` pueden diferir, un bloque de contexto[3.4] solo da acceso a `block.above` y `block.below`, no directamente a `block.spacing`.

Esta propiedad se puede usar en combinación con una regla show para ajustar el espaciado alrededor de elementos arbitrarios de bloque.

```
#set align(center)
#show math.equation: set block(above: 8pt, below: 16pt)

This sum of $x$ and $y$:
$ x + y = z $
A second paragraph.
```




/2.10 **above** `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

El espaciado entre este bloque y su antecesor.

/2.11 **below** `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

El espaciado entre este bloque y su sucesor.

/2.12 **clip** `bool` *Configurable con set* Por defecto: `false`

Si se recorta el contenido que está dentro del bloque.

Recortar es útil cuando el contenido del bloque es más grande que el propio bloque, ya que todo el contenido que exceda los límites del bloque quedará oculto.

```
#block(
  width: 50pt,
  height: 50pt,
  clip: true,
  image("tiger.jpg", width: 100pt, height: 100pt)
)
```



/2.13 **sticky** `bool` *Configurable con set* Por defecto: `false`

Si este bloque debe quedar pegado al siguiente, sin corte entre ambos.

Por defecto, esto se establece en los bloques de títulos para evitar títulos huérfanos al final de la página.

```
// Disable stickiness of headings.
#show heading: set block(sticky: false)
#lorem(20)

= Chapter
#lorem(10)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

## Chapter

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

/2.14 **body** `none` or `content` *Posicional* *Configurable con set* Por defecto: `none`

El contenido del bloque.

## 3.10.5 box Elemento Desde: siempre

Un contenedor en línea que da tamaño al contenido.

Todos los elementos, excepto la matemática en línea, el texto y los boxes, son de nivel de bloque y no pueden aparecer dentro de un párrafo[3.6.14]. La función box se puede usar para integrar esos elementos en un párrafo. Por defecto, los boxes toman el tamaño de su contenido, pero también se les puede dar tamaño[3.10.5/2.1] explícitamente.

De manera similar a los bloques, los boxes se pueden usar para recortar[3.10.5/2.9] contenido o para darle un fondo[3.10.5/2.4] o un borde[3.10.5/2.5]. Sin embargo, solo deberían usarse con estos fines si el contenedor resultante se usa en un contexto en línea. De lo contrario, es preferible un `block`[3.10.4].

### /1 Ejemplo

```
Refer to the docs
#box(
  height: 9pt,
  image("docs.svg")
)
for more information.
```

Refer to the docs for more information.

### /2 Parámetros

```
box(
  width: auto relative fraction,
  height: auto relative,
  baseline: auto relative dictionary alignment,
  fill: none color gradient tiling,
  stroke: none length color gradient stroke tiling dictionary,
  radius: relative dictionary,
  inset: relative dictionary,
  outset: relative dictionary,
  clip: bool,
  none content,
) → content
```

#### /2.1 width `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

El ancho del box.

Los boxes pueden tener anchos fraccionarios[3.10.9], como demuestra el siguiente ejemplo.

*Nota:* Actualmente, dentro de los párrafos solo los boxes, y solo sus anchos, pueden tener tamaño fraccionario. En el futuro se podría agregar compatibilidad con imágenes, formas y más con tamaño fraccionario.

```
Line in #box(width: 1fr, line(length: 100%)) between.
```

Line in _____ between.

#### /2.2 height `auto` or `relative` *Configurable con set* Por defecto: `auto`

La altura del box.

/2.3 **baseline** `auto` or `relative` or `dictionary` or `alignment` *Configurable con set*

Por defecto: `(at: auto, shift: 0% + 0pt)`

La posición vertical de la línea base del box. Se usa para alinear el box con el texto que lo rodea en un párrafo, ya que la línea base está pensada para ir justo debajo del texto por defecto.

Por defecto, la línea base del box coincidirá con la línea base de su contenido (por ejemplo, del texto o la ecuación que contiene): esta es la opción `auto`. Sin embargo, la línea base se puede ajustar de dos maneras. La primera es simplemente elegir una alineación[3.10.2] vertical, como `top`, `horizon` o `bottom`, para colocar la línea base en esa posición, en relación con la altura total del box (incluido el inset).

La otra manera de ajustarla es desplazar verticalmente la línea base por defecto una cierta cantidad, especificada como una longitud relativa[3.10.21]. Por ejemplo, un valor de `2pt` la moverá hacia arriba esa longitud exacta (lo que hace que el contenido baje, ya que el punto de alineación sube), mientras que `-40%` desplazará la línea base *hacia abajo* un 40 % de la altura total del box, incluido el inset (lo que hace que el contenido suba).

Ambas opciones se pueden especificar al mismo tiempo mediante un diccionario con las claves `at` y `shift`, respectivamente. Por ejemplo, al especificar `(at: bottom, shift: 10pt)`, la línea base del box se establecerá a una altura exactamente `10pt` por encima de la parte inferior de su contenido.

```
Image: #box(baseline: 40%, image("tiger.jpg", width: 2cm)).
```

Image:  .

/2.4 **fill** `none` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: none

El color de fondo del box. Mirá la documentación del rectángulo[3.11.10/2.3] para más detalles.

/2.5 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary`
*Configurable con set* Por defecto: `(:)`

El color del borde del box. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

/2.6 **radius** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se redondean las esquinas del box. Mirá la documentación del rectángulo[3.11.10/2.5] para más detalles.

/2.7 **inset** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto relleno se agrega al contenido del box.

Puede ser una única longitud para todos los lados o un diccionario de longitudes para lados individuales. Al pasar un diccionario, puede contener las siguientes claves, en orden de precedencia: `top`, `right`, `bottom`, `left` (que controlan los lados respectivos de la celda), `x`, `y` (que controlan los insets vertical y horizontal) y `rest` (que abarca todos los insets no definidos por otras entradas del diccionario). Todas las claves son opcionales; las claves omitidas usarán su valor establecido previamente o el valor por defecto si nunca se establecieron.

Las longitudes relativas[3.10.21] de este parámetro son relativas al tamaño del box sin contar el outset[3.10.5/2.8]. Tené en cuenta que los insets y outsets relativos son distintos de los anchos[3.10.5/2.1] y altos[3.10.5/2.2] relativos, que son relativos al contenedor.

*Nota:* Cuando el box contiene texto, su tamaño exacto depende de los bordes de texto[3.7.12/2.14] actuales.

```
#rect(inset: 0pt)[Tight]
```

Tight

/2.8 **outset** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se expande el tamaño del box sin afectar el diseño.

Puede ser una única longitud para todos los lados o un diccionario de longitudes para lados individuales. Las longitudes relativas[3.10.21] de este parámetro son relativas al tamaño del box sin contar el outset. Mirá la documentación de inset[3.10.5/2.7] de arriba para más detalles.

Esto es útil para evitar que el relleno afecte el diseño de las líneas. Para una versión generalizada del ejemplo de abajo, mirá la documentación del parámetro block del texto raw[3.7.6/4.2].

```
An inline
#box(
  fill: luma(235),
  inset: (x: 3pt, y: 0pt),
  outset: (y: 3pt),
  radius: 2pt,
)[rectangle].
```

An inline rectangle.

/2.9 **clip** `bool` *Configurable con set* Por defecto: `false`

Si se recorta el contenido que está dentro del box.

Recortar es útil cuando el contenido del box es más grande que el propio box, ya que todo el contenido que exceda los límites del box quedará oculto.

```
#box(
  width: 50pt,
  height: 50pt,
  clip: true,
  image("tiger.jpg", width: 100pt, height: 100pt)
)
```



/2.10 **body** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido del box.

### 3.10.6 `colbreak` Elemento Desde: siempre

Fuerza un salto de columna.

La función se comporta como un salto de página[3.10.18] cuando se usa en un diseño de una sola columna o en la última columna de una página. De lo contrario, el contenido posterior al salto de columna se ubicará en la siguiente columna.

#### /1 Ejemplo

```
#set page(columns: 2)
Preliminary findings from our
ongoing research project have
revealed a hitherto unknown
phenomenon of extraordinary
significance.

#colbreak()
Through rigorous experimentation
and analysis, we have discovered
a hitherto uncharacterized process
that defies our current
understanding of the fundamental
laws of nature.
```

Preliminary findings from our ongoing research project have revealed a hitherto unknown phenomenon of extraordinary significance.

Through rigorous experimentation and analysis, we have discovered a hitherto uncharacterized process that defies our current understanding of the fundamental laws of nature.

#### /2 Parámetros

```
colbreak(weak: bool) → content
```

**/2.1 weak** `bool` *Configurable con set* Por defecto: `false`

Si es `true`, el salto de columna se omite si la columna actual ya está vacía.

### 3.10.7 `columns` Elemento Desde: siempre

Separa una región en varias columnas de igual tamaño.

La función `column` te permite separar el contenido de cualquier contenedor en varias columnas. Por defecto, las columnas ocupan la altura de su contenedor o la altura restante de la página y se van llenando una tras otra. Usá `colbreak`[3.10.6] para terminar explícitamente una columna y continuar en la siguiente. Usá el parámetro `balanced`[3.10.7/4.3] para igualar automáticamente la altura de las columnas.

#### /1 Ejemplo

```
#columns(2, gutter: 8pt)[
  This text is in the
  first column.

  #colbreak()

  This text is in the
  second column.
]
```

This text is in the first column. This text is in the second column.

#### /2 Columnas de página

Si necesitás insertar columnas en todo tu documento, usá en su lugar el parámetro `columns`[3.10.17/4.8] de la función `page`. Esto creará las columnas directamente en la página en lugar de envolver todo tu contenido en un contenedor de diseño. Como resultado, cosas como los saltos de página[3.10.18], las notas al pie[3.6.8] y los números de línea[3.6.14/5.1] seguirán funcionando como se espera. Para más información, leé también la parte relevante de la guía de configuración de página[4.2.4].

#### /3 Salir de las columnas

Para salir temporalmente de las columnas (por ejemplo, para el título de un trabajo), usá una ubicación flotante con ámbito en el padre:

```
#set page(columns: 2, height: 150pt)

#place(
  top + center,
  scope: "parent",
  float: true,
  text(1.4em, weight: "bold")[
    My document
  ],
)

#lorem(40)
```

**My document**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod

#### /4 Parámetros

```
columns(
  int,
  gutter: relative,
  balanced: bool,
  separator: none content,
  content,
) → content
```

/4.1 **count** `int` *Posicional* *Configurable con set* Por defecto: `2`

La cantidad de columnas.

/4.2 **gutter** `relative` *Configurable con set* Por defecto: `4% + 0pt`

El tamaño del espacio entre columnas (gutter).

```
#set page(columns: 2, height: 5cm)
#set par(justify: true)
#set columns(gutter: 15mm)
#lorem(30)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

/4.3 **balanced** `bool` *Configurable con set* Por defecto: `false`

Si se iguala la altura de las columnas cortándolas antes.

```
#set page(columns: 2, height: 5cm)
#set par(justify: true)
#set columns(balanced: true)
#lorem(30)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

/4.4 **separator** `none` or `content` *Configurable con set* Por defecto: `none`

El separador que se coloca entre cada par de columnas no vacías.

Por defecto, el contenido se centra y las líneas[3.11.8] usan una longitud de `100%` y un ángulo de `90deg`, salvo que se especifique otra cosa.

```
#set page(columns: 2, height: 5cm)
#set par(justify: true)
#set columns(
  gutter: 2em,
  separator: line(),
)
#lorem(35)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si.

Example · Separador con línea punteada extendida

```
#set page(columns: 2, height: 5cm)
#set par(justify: true)
#set columns(
  gutter: 2em,
  separator: line(
    stroke: (dash: "dotted"),
    length: 100% + 2 * 4pt,
  ),
)
#lorem(35)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si.

Example · Adorno de separador personalizado

```
#set page(columns: 3, width: 15cm, height: 3cm)
#set par(justify: true)
#set columns(
  gutter: 30pt,
  separator: curve(curve.cubic((-8pt, 0%), (8pt, 50%), (0pt, 50%))
))
#lorem(36)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod.

/4.5 **body** `content` *Obligatorio* *Posicional*

El contenido que se dispone en las columnas.

### 3.10.8 `direction` Desde: siempre

Las cuatro direcciones en las que se puede disponer el contenido.

Los valores posibles son: `ltr`[3.10.8/1.1], `rtl`[3.10.8/1.2], `ttb`[3.10.8/1.3] y `btt`[3.10.8/1.4].

Estos valores están disponibles globalmente y también en el ámbito del tipo direction, así que podés escribir cualquiera de las dos formas siguientes:

```
#stack(dir: rtl)[A][B][C]
#stack(dir: direction.rtl)[A][B][C]
```

CBA

CBA

## /1 Constantes

### /1.1 `ltr` direction Desde: siempre

De izquierda a derecha.

```
#stack(
  dir: ltr,
  spacing: 0.2cm,
  circle(),
  line(),
  square(),
)
```



### /1.2 `rtl` direction Desde: siempre

De derecha a izquierda.

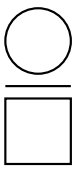
```
#stack(
  dir: rtl,
  spacing: 0.2cm,
  circle(),
  line(),
  square(),
)
```



### /1.3 `ttb` direction Desde: siempre

De arriba hacia abajo.

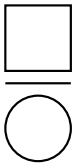
```
#stack(
  dir: ttb,
  spacing: 0.2cm,
  circle(),
  line(),
  square(),
)
```



### /1.4 `btt` direction Desde: siempre

De abajo hacia arriba.

```
#stack(
  dir: btt,
  spacing: 0.2cm,
  circle(),
  line(),
  square(),
)
```



## /2 Funciones

### /2.1 from Desde: 0.14.0

Devuelve una dirección a partir de un punto de partida.

Esta función hace lo opuesto de direction.start[3.10.8/2.5].

```
#direction.from(left) \
#direction.from(right) \
#direction.from(top) \
#direction.from(bottom)
```

```
ltr
rtl
ttb
btt
```

```
direction.from( alignment ) → direction
```

#### /2.1.1 side alignment *Obligatorio* *Posicional*

El punto de partida de la dirección.

No puede ser start ni end.

### /2.2 to Desde: 0.14.0

Devuelve una dirección a partir de un punto final.

Esta función hace lo opuesto de direction.end[3.10.8/2.6].

```
#direction.to(left) \
#direction.to(right) \
#direction.to(top) \
#direction.to(bottom)
```

```
rtl
ltr
btt
ttb
```

```
direction.to( alignment ) → direction
```

#### /2.2.1 side alignment *Obligatorio* *Posicional*

El punto final de la dirección.

No puede ser start ni end.

### /2.3 axis Desde: 0.7.0

El eje al que pertenece esta dirección, "horizontal" o "vertical".

```
#ltr.axis() \
#ttb.axis()
```

```
horizontal
vertical
```

```
self.axis() → str
```

### /2.4 sign Desde: 0.14.0

El signo correspondiente, para usar en cálculos.

Es el signo del vector que va en esta dirección a lo largo del eje correspondiente.

```
#ltr.sign() \
#rtl.sign() \
#ttb.sign() \
#btt.sign()
```

```
1
−1
1
−1
```

```
self.sign() → int
```

### /2.5 start Desde: 0.7.0

El punto de partida de esta dirección.

Esta función hace lo opuesto de direction.from[3.10.8/2.1].

```
#ltr.start() \
#rtl.start() \
#ttb.start() \
#btt.start()
```

```
left
right
top
bottom
```

```
self.start() → alignment
```

### /2.6 end Desde: 0.7.0

El punto final de esta dirección.

Esta función hace lo opuesto de direction.to[3.10.8/2.2].

```
#ltr.end() \
#rtl.end() \
#ttb.end() \
#btt.end()
```

```
right
left
bottom
top
```

```
self.end() → alignment
```

### /2.7 inv Desde: siempre

La dirección inversa (opuesta).

```
#ltr.inv() \
#rtl.inv() \
#ttb.inv() \
#btt.inv()
```

```
rtl
ltr
btt
ttb
```

self.inv() → direction

### 3.10.9 `fraction` Desde: siempre

Define cómo se distribuye el espacio restante en un diseño.

Cada elemento con tamaño fraccionario recibe espacio según la proporción entre su fracción y la suma de todas las fracciones.

Para más detalles, mirá también las funciones h[3.10.26] y v[3.10.27] y la función `grid`[3.10.10].

#### /1 Ejemplo

```
Left #h(1fr) Left-ish #h(2fr) Right
```



## 3.10.10 `grid` Elemento Desde: siempre

Dispone contenido en una grilla.

El elemento grid te permite disponer contenido en una grilla. Podés definir la cantidad de filas y columnas, así como el tamaño de los espacios (gutters) entre ellas. Hay varios modos de dimensionamiento para columnas y filas que se pueden usar para crear diseños complejos.

Si bien los elementos grid y table funcionan de manera muy similar, están pensados para distintos casos de uso y tienen distinta semántica. El elemento grid está pensado para fines de presentación y de diseño, mientras que el elemento `table`[3.6.19] está pensado, en términos generales, para presentar varios datos relacionados. Las reglas set y show sobre uno de estos elementos no afectan al otro. Consultá la sección de accesibilidad[3.10.10/4] para saber cómo se presentan las grillas y las tablas a los usuarios de tecnologías de asistencia (TA), como los lectores de pantalla.

### /1 Dimensionar las pistas

El dimensionamiento de una grilla se determina por los tamaños de pista especificados en los argumentos. Hay varios parámetros de dimensionamiento: `columns`[3.10.10/5.1], `rows`[3.10.10/5.2] y `gutter`[3.10.10/5.3]. Como cada uno de los parámetros de dimensionamiento acepta los mismos valores, los vamos a explicar una sola vez, acá. Cada argumento de dimensionamiento acepta un array de tamaños de pista individuales. Un tamaño de pista es uno de los siguientes:

- `auto`: La pista se dimensionará para ajustarse a su contenido. Será, como máximo, tan grande como el espacio restante. Si hay más de un ancho de pista `auto` y, en conjunto, reclaman más que el espacio disponible, las pistas `auto` se repartirán el espacio disponible de forma equitativa.
- Una longitud fija o relativa (por ejemplo, `10pt` o `20% - 1cm`): La pista tendrá exactamente ese tamaño.
- Una longitud fraccionaria (por ejemplo, `1fr`): Una vez dimensionadas todas las demás pistas, el espacio restante se dividirá entre las pistas fraccionarias según sus fracciones. Por ejemplo, si hay dos pistas fraccionarias, cada una con una fracción de `1fr`, cada una ocupará la mitad del espacio restante.

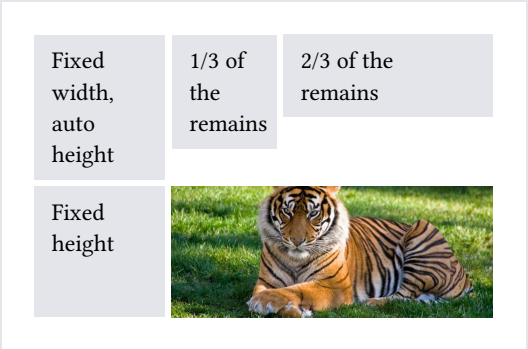
Para especificar una sola pista, se puede omitir el array y usar un único valor. Para especificar varias pistas `auto`, ingresá la cantidad de pistas en lugar de un array. Por ejemplo, `columns: 3` equivale a `columns: (auto, auto, auto)`.

### /2 Ejemplos

El siguiente ejemplo demuestra las distintas opciones de dimensionamiento de pistas. También muestra cómo podés usar `grid.cell`[3.10.10/6.1] para que una celda individual abarque dos pistas de la grilla.

```
// We use `rect` to emphasize the
// area of cells.
#set rect(
  inset: 8pt,
  fill: rgb("e4e5ea"),
  width: 100%,
)

#grid(
  columns: (60pt, 1fr, 2fr),
  rows: (auto, 60pt),
  gutter: 3pt,
  rect[Fixed width, auto height],
  rect[1/3 of the remains],
  rect[2/3 of the remains],
  rect(height: 100%)[Fixed height],
  grid.cell(
    colspan: 2,
    image("tiger.jpg", width: 100%),
```



```
  ),
)
```

También podés expandir[3.5.1/2] un array de cadenas de texto o de contenido en una grilla para completar sus celdas.

```
#grid(
  columns: 5,
  gutter: 5pt,
  ..range(25).map(str)
)
```

0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
20 21 22 23 24

## /3 Dar estilo a la grilla

Los elementos grid y table funcionan de manera similar. Para una explicación práctica, consultá la guía de tablas[4.3.4]; para un panorama rápido, seguí leyendo.

El aspecto de la grilla se puede personalizar mediante distintos parámetros. Estos son los más importantes:

- `align`[3.10.10/5.7] para cambiar cómo se alinean las celdas
- `inset`[3.10.10/5.6] para agregar opcionalmente relleno interno a las celdas
- `fill`[3.10.10/5.8] para darles un fondo a las celdas
- `stroke`[3.10.10/5.9] para habilitar opcionalmente líneas de grilla con cierto trazo

Para cubrir distintas necesidades, hay varias maneras de establecerlos.

Si necesitás reemplazar las opciones anteriores en celdas individuales, podés usar el elemento `grid.cell`[3.10.10/6.1]. Del mismo modo, podés reemplazar líneas individuales de la grilla con los elementos `grid.hline`[3.10.10/6.2] y `grid.vline`[3.10.10/6.3].

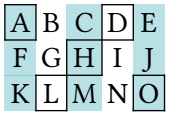
Para configurar un estilo general para una grilla, en cambio, podés especificar la opción de cualquiera de las siguientes maneras:

- Como un único valor que se aplica a todas las celdas.
- Como un array de valores correspondientes a cada columna. El array se recorrerá cíclicamente si hay más columnas que elementos en el array.
- Como una función de la forma `(x, y) => value`. Recibe los índices de columna y fila de la celda (ambos desde cero) y debe devolver el valor que se aplica a esa celda.

```
#grid(
  columns: 5,

  // By a single value
  align: center,
  // By a single but more complicated value
  inset: (x: 2pt, y: 3pt),
  // By an array of values (cycling)
  fill: (rgb("#239dad50"), none),
  // By a function that returns a value
  stroke: (x, y) => if calc.rem(x + y, 3) == 0 { 0.5pt },

  ..range(5 * 3).map(n => numbering("A", n + 1))
)
```




Además, podés aplicar reglas de estilo[3.2] a grid[3.10.10] y a grid.cell[3.10.10/6.1]. Especialmente, los campos x[3.10.10/6.1.2] y y[3.10.10/6.1.3] de grid.cell se pueden usar en un selector where[3.5.16/8.2], lo que permite dar estilo a celdas de columnas o filas específicas, o de posiciones individuales.

## /3.1 Precedencia del estilo de trazo

Como se explicó arriba, hay tres maneras de establecer el trazo de una celda de la grilla: mediante el campo stroke de grid.cell[3.10.10/6.1.9], con grid.hline[3.10.10/6.2] y grid.vline[3.10.10/6.3], o estableciendo el campo stroke de grid[3.10.10/5.9]. Cuando varias de estas configuraciones están presentes y entran en conflicto, las de hline y vline tienen la precedencia más alta, seguidas de las de cell y, por último, las de grid.

Además, los trazos de un encabezado o pie de grilla repetido tendrán precedencia sobre los trazos de las celdas normales.

# /4 Accesibilidad

Las grillas no tienen ninguna semántica especial. Las tecnologías de asistencia (TA) no ofrecen la posibilidad de navegar bidimensionalmente por celdas en las grillas. Si querés presentar datos tabulares, usá en su lugar el elemento table[3.6.19].

Las TA leerán las celdas de la grilla en su orden semántico. Normalmente, es el orden en que se las pasaste a la grilla. Sin embargo, si las posicionaste manualmente con los argumentos x e y de grid.cell[3.10.10/6.1.2], las celdas se leerán fila por fila, de izquierda a derecha (en documentos de izquierda a derecha). Una celda se leerá cuando se llegue por primera vez a su posición.

# /5 Parámetros

```
grid(
  columns: auto int relative fraction array,
  rows: auto int relative fraction array,
  gutter: auto int relative fraction array,
  column-gutter: auto int relative fraction array,
  row-gutter: auto int relative fraction array,
  inset: relative array dictionary function,
  align: auto array function alignment,
  fill: none color gradient tiling array function,
  stroke: none length color gradient stroke tiling array dictionary function,
  ..content,
) → content
```

## /5.1 columns `auto` or `int` or `relative` or `fraction` or `array` *Configurable con set* Por defecto: ()

Los tamaños de las columnas.

Especificá un array de tamaños de pista o proporcioná un entero para crear una grilla con esa cantidad de columnas de tamaño auto. Tené en cuenta que, a diferencia de las filas y los gutters, proporcionar un único tamaño de pista siempre crea una sola columna.

Mirá la sección de tamaño de pistas[3.10.10/1] de arriba para más detalles.

## /5.2 rows `auto` or `int` or `relative` or `fraction` or `array` *Configurable con set* Por defecto: ()

Los tamaños de las filas.

Si hay más celdas de las que entran en las filas definidas, la última fila se repite hasta que no queden más celdas.

Mirá la sección de tamaño de pistas[3.10.10/1] de arriba para más detalles.

/5.3 **gutter** `auto` or `int` or `relative` or `fraction` or `array` Por defecto: ()

Los espacios entre filas y columnas. Es un atajo para establecer `column-gutter`[3.10.10/5.4] y `row-gutter`[3.10.10/5.5] en el mismo valor.

Si hay más gutters que tamaños definidos, se repite el último gutter.

Mirá la sección de tamaño de pistas[3.10.10/1] de arriba para más detalles.

/5.4 **column-gutter** `auto` or `int` or `relative` or `fraction` or `array` *Configurable con set*
Por defecto: ()

Los espacios entre columnas.

/5.5 **row-gutter** `auto` or `int` or `relative` or `fraction` or `array` *Configurable con set*
Por defecto: ()

Los espacios entre filas.

/5.6 **inset** `relative` or `array` or `dictionary` or `function` *Configurable con set* Por defecto: (:)

Cuánto relleno se agrega al contenido de las celdas.

Para especificar un inset uniforme para todas las celdas, podés usar una única longitud para todos los lados o un diccionario de longitudes para lados individuales. Mirá la documentación de box[3.10.5/2.7] para más detalles.

Para especificar insets distintos para distintas celdas, podés:
- usar un único inset para todas las celdas
- usar un array de insets correspondientes a cada columna
- usar una función que asigne la posición de una celda a su inset

Mirá la sección de estilos[3.10.10/3] de arriba para más detalles.

Además, podés encontrar un ejemplo en el parámetro `table.inset`[3.6.19/3.6].

/5.7 **align** `auto` or `array` or `function` or `alignment` *Configurable con set* Por defecto: `auto`

Cómo alinear el contenido de las celdas.

Si se establece en `auto`, se usa la alineación exterior.

Podés especificar la alineación de cualquiera de las siguientes maneras:
- usar una única alineación para todas las celdas
- usar un array de alineaciones correspondientes a cada columna
- usar una función que asigne la posición de una celda a su alineación

Mirá la sección de estilos[3.10.10/3] de arriba para más detalles.

Además, podés encontrar un ejemplo en el parámetro `table.align`[3.6.19/3.7].

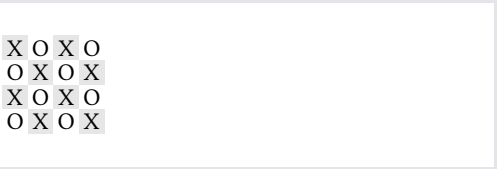
/5.8 **fill** `none` or `color` or `gradient` or `tiling` or `array` or `function` *Configurable con set*
Por defecto: `none`

Cómo rellenar las celdas.

Puede ser:
- un único color para todas las celdas
- un array de colores correspondientes a cada columna
- una función que asigne la posición de una celda a su color

Sobre todo, los arrays y las funciones son útiles para crear grillas con rayas. Mirá la sección de estilos[3.10.10/3] de arriba para más detalles.

```
#grid(
  fill: (x, y) =>
    if calc.even(x + y) { luma(230) }
    else { white },
  align: center + horizon,
  columns: 4,
  inset: 2pt,
  [X], [O], [X], [O],
  [O], [X], [O], [X],
  [X], [O], [X], [O],
  [O], [X], [O], [X],
)
```




/5.9 **stroke** none or length or color or gradient or stroke or tiling or array or dictionary or function *Configurable con set* Por defecto: (:)

Cómo trazar[3.11.12] las celdas.

Las grillas no tienen trazos por defecto, lo que se puede cambiar estableciendo esta opción en el trazo deseado.

Si hace falta ubicar líneas que puedan cruzar el espacio entre celdas producido por la opción gutter[3.10.10/5.3], o reemplazar el trazo entre varias celdas específicas, considerá especificar uno o más de grid.hline[3.10.10/6.2] y grid.vline[3.10.10/6.3] junto con las celdas de tu grilla.

Para especificar el mismo trazo para todas las celdas, podés usar un único trazo[3.11.12] para todos los lados o un diccionario de trazos[3.11.12] para lados individuales. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

Para especificar trazos distintos para distintas celdas, podés:
- usar un único trazo para todas las celdas
- usar un array de trazos correspondientes a cada columna
- usar una función que asigne la posición de una celda a su trazo

Mirá la sección de estilos[3.10.10/3] de arriba para más detalles.

Example · Pasar una función para establecer un trazo según la posición

```
#set page(width: 420pt)
#set text(number-type: "old-style")
#show grid.cell.where(y: 0): set text(size: 1.3em)

#grid(
  columns: (1fr, 2fr, 2fr),
  row-gutter: 1.5em,
  inset: (left: 0.5em),
  stroke: (x, y) => if x > 0 { (left: 0.5pt + gray) },
  align: horizon,

  [Winter \ 2007 \ Season],
  [Aaron Copland \ *The Tender Land* \ January 2007],
  [Eric Satie \ *Gymnopedie 1, 2* \ February 2007],

  [],
  [Jan 12 \ *Middlebury College \ Center for the Arts* \ 20:00],
  [Feb 2 \ *Johnson State College Dibden Center for the Arts* \ 19:30],

  [],
  [Skip a week \ #text(0.8em)[_Prepare your exams!_]],
  [Feb 9 \ *Castleton State College \ Fine Arts Center* \ 19:30],

  [],
  [Jan 26, 27 \ *Lyndon State College Alexander Twilight Theater* \ 20:00],
  [
```

```
    Feb 17 --- #smallcaps[Anniversary] \
    *Middlebury College \ Center for the Arts* \
    19:00 #text(0.7em)[(for a special guest)]
  ],
)
```




Example · Plegar el diccionario de trazos

```
#set page(height: 13em, width: 26em)

#let cv(..jobs) = grid(
  columns: 2,
  inset: 5pt,
  stroke: (x, y) => if x == 0 and y > 0 {
    (right: (
      paint: luma(180),
      thickness: 1.5pt,
      dash: "dotted",
    ))
  },
  grid.header(grid.cell(colspan: 2)[
    *Professional Experience*
    #box(width: 1fr, line(length: 100%, stroke: luma(180)))
  ]),
  ..{
    let last = none
    for job in jobs.pos() {
      (
        if job.year != last [*#job.year*],
        [
          *#job.company* - #job.role _(#job.timeframe)_ \
          #job.details
        ]
      )
      last = job.year
    }
```

```
  }
)

#cv(
  (
    year: 2012,
    company: [Pear Seed & Co.],
    role: [Lead Engineer],
    timeframe: [Jul - Dec],
    details: [
      - Raised engineers from 3x to 10x
      - Did a great job
    ],
  ),
  (
    year: 2012,
    company: [Mega Corp.],
    role: [VP of Sales],
    timeframe: [Mar - Jun],
    details: [- Closed tons of customers],
  ),
  (
    year: 2013,
    company: [Tiny Co.],
    role: [CEO],
    timeframe: [Jan - Dec],
    details: [- Delivered 4x more shareholder value],
  ),
  (
    year: 2014,
    company: [Glorbocorp Ltd],
    role: [CTO],
    timeframe: [Jan - Mar],
    details: [- Drove containerization forward],
  ),
)
```






/5.10 **children** content *Obligatorio* *Posicional* *Variádico*

El contenido de las celdas de la grilla, más las líneas de grilla adicionales especificadas con los elementos `grid.hline`[3.10.10/6.2] y `grid.vline`[3.10.10/6.3].

Las celdas se completan por filas.

## /6 Funciones

/6.1 **cell** Elemento Desde: 0.11.0

Una celda de la grilla. Podés usar esta función en la lista de argumentos de una grilla para reemplazar las propiedades de estilo de la grilla en una celda individual o para posicionarla manualmente dentro de la grilla. También podés usar esta función en reglas show para aplicar ciertos estilos a varias celdas a la vez.

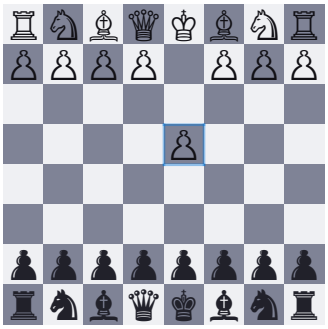
Por ejemplo, podés reemplazar la posición y el trazo de una sola celda:

```
#set text(15pt, font: "Noto Sans Symbols 2")
#show regex("[♚-♟]"): set text(fill: rgb("21212A"))
#show regex("[♔-♙]"): set text(fill: rgb("111015"))

#grid(
  fill: (x, y) => rgb(
    if calc.odd(x + y) { "7F8396" }
    else { "EFF0F3" }
  ),
  columns: (1em,) * 8,
  rows: 1em,
  align: center + horizon,

  [♖], [♘], [♗], [♕], [♔], [♗], [♘], [♖],
  [♙], [♙], [♙], [♙], [],  [♙], [♙], [♙],
  grid.cell(
    x: 4, y: 3,
    stroke: blue.transparentize(60%)
  )[♙],

  ..(grid.cell(y: 6)[♟],) * 8,
  ..([♜], [♞], [♝], [♛], [♚], [♝], [♞], [♜])
    .map(grid.cell.with(y: 7)),
)
```



También podés aplicar una regla show sobre `grid.cell` para dar estilo a todas las celdas a la vez, lo que te permite, por ejemplo, aplicar estilos según la posición de una celda. Consultá los ejemplos del elemento `table.cell`[3.6.19/4.1] para aprender más sobre esto.

```
grid.cell(
  content,
  x: auto int,
  y: auto int,
  colspan: int,
  rowspan: int,
  inset: auto relative dictionary,
  align: auto alignment,
  fill: none auto color gradient tiling,
  stroke: none length color gradient stroke tiling dictionary,
  breakable: auto bool,
) → content
```

/6.1.1 **body** `content` *Obligatorio* *Posicional*

El cuerpo de la celda.

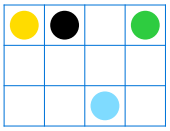
/6.1.2 **x** `auto` or `int` *Configurable con set* Por defecto: `auto`

La columna de la celda (con índice desde cero). Este campo se puede usar en reglas show para dar estilo a una celda según su columna.

Podés reemplazar este campo para elegir en qué columna se debe ubicar la celda. Si no se elige una fila (`y`), la celda se ubicará en la primera fila (empezando por la fila 0) en la que esa columna esté disponible (o en una fila nueva, si no hay ninguna). En cambio, si se eligen tanto `x` como `y`, la celda se ubicará exactamente en esa posición. Se produce un error si esa posición no está disponible (por eso, suele ser prudente especificar las celdas con posición personalizada antes que las celdas con posición automática).

```
#let circ(c) = circle(
  fill: c,
  width: 5mm,
)

#grid(
  columns: 4,
  rows: 7mm,
  stroke: .5pt + blue,
  align: center + horizon,
  inset: 1mm,
```



```
  grid.cell(x: 2, y: 2, circ(aqua)),
  circ(yellow),
  grid.cell(x: 3, circ(green)),
  circ(black),
)
```

/6.1.3 **y** `auto` or `int` *Configurable con set* Por defecto: `auto`

La fila de la celda (con índice desde cero). Este campo se puede usar en reglas show para dar estilo a una celda según su fila.

Podés reemplazar este campo para elegir en qué fila se debe ubicar la celda. Si no se elige una columna (x), la celda se ubicará en la primera columna disponible (empezando por la columna 0) de la fila elegida. Si todas las columnas de la fila elegida ya están ocupadas, se produce un error.

```
#let tri(c) = polygon.regular(
  fill: c,
  size: 5mm,
  vertices: 3,
)

#grid(
  columns: 2,
  stroke: blue,
  inset: 1mm,

  tri(black),
  grid.cell(y: 1, tri(teal)),
  grid.cell(y: 1, tri(red)),
  grid.cell(y: 2, tri(orange))
)
```



/6.1.4 **colspan** `int` *Configurable con set* Por defecto: `1`

La cantidad de columnas que abarca esta celda.

/6.1.5 **rowspan** `int` *Configurable con set* Por defecto: `1`

La cantidad de filas que abarca esta celda.

/6.1.6 **inset** `auto` or `relative` or `dictionary` *Configurable con set* Por defecto: `auto`

El reemplazo de inset[3.10.10/5.6] de la celda.

/6.1.7 **align** `auto` or `alignment` *Configurable con set* Por defecto: `auto`

El reemplazo de alineación[3.10.10/5.7] de la celda.

/6.1.8 **fill** `none` or `auto` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `auto`

El reemplazo de relleno[3.10.10/5.8] de la celda.

/6.1.9 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `(:)`

El reemplazo de trazo[3.10.10/5.9] de la celda.

/6.1.10 **breakable** `auto` or `bool` *Configurable con set* Por defecto: `auto`

Si las filas que abarca esta celda se pueden ubicar en páginas distintas. Cuando es igual a `auto`, una celda que abarca solo filas de tamaño fijo no se puede cortar, mientras que una celda que abarca al menos una fila de tamaño `auto` sí se puede cortar.

/6.2 **hline** Elemento Desde: 0.11.0

Una línea horizontal en la grilla.

Reemplaza cualquier trazo por celda, incluido el trazo especificado mediante el campo `stroke` de la grilla. Puede cruzar el espacio entre celdas creado mediante la opción `column-gutter` de la grilla.

Se puede encontrar un ejemplo de esta función en el elemento `table.hline`[3.6.19/4.2].

```
grid.hline(
  y: auto int,
  start: int,
  end: none int,
  stroke: none length color gradient stroke tiling dictionary,
  position: alignment,
) → content
```

/6.2.1 **y** `auto` or `int` *Configurable con set* Por defecto: `auto`

La fila encima de la cual se ubica la línea horizontal (con índice desde cero). Si el campo `position` se establece en `bottom`, la línea se ubica en cambio debajo de la fila con el índice dado (mirá `grid.hline.position`[3.10.10/6.2.5] para más detalles).

Especificar `auto` hace que la línea se ubique en la fila siguiente a la última celda posicionada automáticamente (es decir, sin coordenadas reemplazadas) anterior a la línea entre los hijos de la grilla. Si no hay ninguna celda así antes de la línea, se ubica en la parte superior de la grilla (fila 0). Tené en cuenta que especificar en esta opción exactamente la cantidad total de filas de la grilla hace que esta línea horizontal reemplace el borde inferior de la grilla, mientras que un valor de 0 reemplaza el borde superior.

/6.2.2 **start** `int` *Configurable con set* Por defecto: `0`

La columna en la que empieza la línea horizontal (con índice desde cero, inclusivo).

/6.2.3 **end** `none` or `int` *Configurable con set* Por defecto: `none`

La columna antes de la cual termina la línea horizontal (con índice desde cero, exclusivo). Por lo tanto, la línea horizontal se dibujará hasta la columna `end - 1` inclusive, y a lo largo de ella.

Un valor igual a `none` o a la cantidad de columnas hace que se extienda hasta el final de la grilla.

/6.2.4 **stroke** `none` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `stroke()`

El trazo de la línea.

Especificar `none` elimina cualquier línea ubicada previamente a lo largo del rango de esta línea, incluidas las hlines o el trazo por celda que estén debajo.

/6.2.5 **position** `alignment` *Configurable con set* Por defecto: `top`

La posición en la que se ubica la línea, dada su fila (`y`): `top` para dibujarla por encima o `bottom` para dibujarla por debajo.

Esta configuración solo es relevante cuando el gutter de filas está habilitado (y no se debe usar de otro modo; conviene simplemente aumentar en uno el campo `y`), ya que en ese caso la posición debajo de una fila pasa a ser distinta de la posición encima de la fila siguiente, debido al espacio entre ambas.

## /6.3 vline Elemento Desde: 0.11.0

Una línea vertical en la grilla.

Reemplaza cualquier trazo por celda, incluido el trazo especificado mediante el campo `stroke` de la grilla. Puede cruzar el espacio entre celdas creado mediante la opción `row-gutter` de la grilla.

```
grid.vline(
  x: auto int,
  start: int,
  end: none int,
```

```
  stroke: none length color gradient stroke tiling dictionary,
  position: alignment,
) → content
```

/6.3.1 **x** auto or int *Configurable con set* Por defecto: auto

La columna antes de la cual se ubica la línea vertical (con índice desde cero). Si el campo position se establece en end, la línea se ubica en cambio después de la columna con el índice dado (mirá grid.vline.position[3.10.10/6.3.5] para más detalles).

Especificar auto hace que la línea se ubique en la columna siguiente a la última celda posicionada automáticamente (es decir, sin coordenadas reemplazadas) anterior a la línea entre los hijos de la grilla. Si no hay ninguna celda así antes de la línea, se ubica antes de la primera columna de la grilla (columna 0). Tené en cuenta que especificar en esta opción exactamente la cantidad total de columnas de la grilla hace que esta línea vertical reemplace el borde final de la grilla (derecho en LTR, izquierdo en RTL), mientras que un valor de 0 reemplaza el borde inicial (izquierdo en LTR, derecho en RTL).

/6.3.2 **start** int *Configurable con set* Por defecto: 0

La fila en la que empieza la línea vertical (con índice desde cero, inclusivo).

/6.3.3 **end** none or int *Configurable con set* Por defecto: none

La fila encima de la cual termina la línea vertical (con índice desde cero, exclusivo). Por lo tanto, la línea vertical se dibujará hasta la fila end - 1 inclusive, y a lo largo de ella.

Un valor igual a none o a la cantidad de filas hace que se extienda hasta la parte inferior de la grilla.

/6.3.4 **stroke** none or length or color or gradient or stroke or tiling or dictionary *Configurable con set* Por defecto: stroke()

El trazo de la línea.

Especificar none elimina cualquier línea ubicada previamente a lo largo del rango de esta línea, incluidas las vlines o el trazo por celda que estén debajo.

/6.3.5 **position** alignment *Configurable con set* Por defecto: start

La posición en la que se ubica la línea, dada su columna (x): start para dibujarla antes o end para dibujarla después.

También se aceptan los valores left y right, pero no se recomiendan, ya que hacen que tu grilla sea inconsistente entre documentos de izquierda a derecha y de derecha a izquierda.

Esta configuración solo es relevante cuando el gutter de columnas está habilitado (y no se debe usar de otro modo; conviene simplemente aumentar en uno el campo x), ya que en ese caso la posición después de una columna pasa a ser distinta de la posición antes de la columna siguiente, debido al espacio entre ambas.

## /6.4 header Elemento Desde: 0.11.0

Un encabezado de grilla repetible.

Si repeat se establece en true, el encabezado se repetirá en todas las páginas. Para ver un ejemplo, consultá el elemento table.header[3.6.19/4.4] y el parámetro grid.stroke[3.10.10/5.9].

```
grid.header(
  repeat: bool,
  level: int,
  ..content,
) → content
```

/6.4.1 **repeat** bool *Configurable con set* Por defecto: true

Si este encabezado se debe repetir en todas las páginas.

/6.4.2 **level** `int` *Configurable con set* Por defecto: `1`

El nivel del encabezado. No debe ser cero.

Esto permite repetir varios encabezados a la vez. Los encabezados con distintos niveles pueden repetirse juntos, siempre que tengan niveles ascendentes.

En particular, cuando un encabezado de nivel más bajo empieza a repetirse, todos los encabezados de nivel más alto o igual dejan de repetirse (son "reemplazados" por el nuevo encabezado).

/6.4.3 **children** `content` *Obligatorio* *Posicional* *Variádico*

Las celdas y líneas dentro del encabezado.

## /6.5 footer Elemento Desde: 0.11.0

Un pie de grilla repetible.

Igual que el elemento `grid.header`[3.10.10/6.4], el pie puede repetirse en cada página de la grilla.

No se pueden ubicar otras celdas de la grilla después del pie.

```
grid.footer(
  repeat: bool,
  ..content,
) → content
```

/6.5.1 **repeat** `bool` *Configurable con set* Por defecto: `true`

Si este pie se debe repetir en todas las páginas.

/6.5.2 **children** `content` *Obligatorio* *Posicional* *Variádico*

Las celdas y líneas dentro del pie.

### 3.10.11 `hide` Elemento Desde: siempre

Oculta contenido sin afectar el diseño.

La función `hide` te permite ocultar contenido mientras el diseño todavía lo "ve". Esto es útil para crear espacio en blanco que sea exactamente tan grande como algún contenido.

#### /1 Ejemplo

```
Hello Jane \
#hide[Hello] Joe
```

Hello Jane
Joe

#### /2 Censura

Esta función también puede ser útil para censurar contenido, ya que sus argumentos no están presentes visualmente ni son accesibles para las tecnologías de asistencia. Dicho esto, puede haber *algunos* rastros del contenido oculto (como un título con marcador en el esquema del documento del PDF).

Tené en cuenta que, según las circunstancias, podría ser posible reconstruir el contenido a partir de su tamaño en el diseño. Por eso no recomendamos usar esta función para ocultar información muy sensible.

#### /3 Parámetros

```
hide(content) → content
```

##### /3.1 **body** `content` *Obligatorio* *Posicional*

El contenido que se oculta.

### 3.10.12 `layout` Desde: 0.2.0

Proporciona acceso a las dimensiones (ancho y alto) del contenedor exterior actual (o de la página, si no hay ninguno).

Acepta una función que recibe un único parámetro, que es un diccionario con las claves `width` y `height`, ambas de tipo `length`[3.10.13]. A la función se le proporciona contexto[3.4], lo que significa que no necesitás usarla en combinación con la palabra clave `context`. Por eso se puede llamar a `measure`[3.10.14] en el siguiente ejemplo.

```
#let text = lorem(30)
#layout(size => [
  #let (height,) = measure(
    width: size.width,
    text,
  )
  This text is #height high with
  the current page width: \
  #text
])
```

This text is 64.79pt high with the current page width:
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

Tené en cuenta que la función `layout` fuerza su contenido dentro de un contenedor a nivel de bloque[3.10.4], por lo que dentro de ella no es posible ubicar elementos relativos a la página ni hacer saltos de página.

Si la llamada a `layout` se coloca dentro de un box con un ancho de `800pt` y una altura de `400pt`, la función especificada recibirá el argumento `(width: 800pt, height: 400pt)`. Si se coloca directamente en la página, recibe las dimensiones de la página menos sus márgenes. Esto es útil sobre todo en combinación con la medición[3.10.14].

Para obtener la altura *restante* de la página en lugar de su tamaño completo, podés envolver tu llamada a `layout` en un `block(height: 1fr)`. Esto funciona porque el bloque crece automáticamente para llenar el espacio restante (mirá la documentación de fracciones[3.10.9] para más detalles).

```
#set page(height: 150pt)

#lorem(20)

#block(height: 1fr, layout(size => [
  Remaining height: #size.height
]))
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Remaining height: 56.4pt

También podés usar esta función para resolver un `ratio`[3.10.20] en una longitud fija. Esto puede venir bien si estás construyendo tus propias abstracciones de diseño.

```
#layout(size => {
  let half = 50% * size.width
  [Half a page is #half wide.]
})
```

Half a page is 105pt wide.

Tené en cuenta que el ancho o la altura que proporciona `layout` serán infinitos si la dimensión de página correspondiente está establecida en `auto`.

#### /1 Parámetros

```
layout(function) → content
```

/1.1 **func** `function` *Obligatorio* *Posicional*

Una función a la que se llama con el tamaño del contenedor exterior. Su valor de retorno se muestra en el documento.

El tamaño del contenedor se da como un diccionario[3.5.12] con las claves `width` y `height`, ambas de tipo `length`[3.10.13].

Esta función se llama una vez por cada vez que el contenido devuelto por `layout` aparece en el documento. Esto permite generar contenido que dependa de las dimensiones de su contenedor.

### 3.10.13 `length` Desde: siempre

Un tamaño o una distancia, posiblemente expresados con unidades contextuales.

Typst admite las siguientes unidades de longitud:

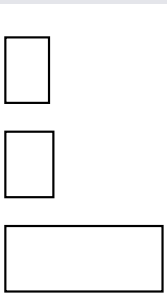
- Puntos: `72pt`
- Milímetros: `254mm`
- Centímetros: `2.54cm`
- Pulgadas: `1in`
- Relativa al tamaño de la fuente: `2.5em`

Podés multiplicar longitudes por enteros y flotantes, y dividirlas por ellos.

#### /1 Ejemplo

```
#rect(width: 20pt)
#rect(width: 2em)
#rect(width: 1in)

#(3em + 5pt).em \
#(20pt).em \
#(40em + 2pt).abs \
#(5em).abs
```



3
0
2pt
0pt

#### /2 Campos

- `abs`: Una longitud con solo el componente absoluto de la longitud actual (es decir, sin el componente `em`).
- `em`: La cantidad de unidades `em` de esta longitud, como flotante[3.5.15].

#### /3 Funciones

##### /3.1 `pt` Desde: 0.7.0

Convierte esta longitud a puntos.

Falla con un error si esta longitud tiene unidades `em` distintas de cero (como `5em + 2pt` en lugar de solo `2pt`). Usá el campo `abs` (como en `(5em + 2pt).abs.pt()`) para ignorar el componente `em` de la longitud (y así convertir solo su componente absoluto).

```
self.pt() → float
```

##### /3.2 `mm` Desde: 0.7.0

Convierte esta longitud a milímetros.

Falla con un error si esta longitud tiene unidades `em` distintas de cero. Mirá el método pt[3.10.13/3.1] para más detalles.

```
self.mm() → float
```

### /3.3 cm Desde: 0.7.0

Convierte esta longitud a centímetros.

Falla con un error si esta longitud tiene unidades em distintas de cero. Mirá el método pt[3.10.13/3.1] para más detalles.

```
self.cm() → float
```

### /3.4 inches Desde: 0.7.0

Convierte esta longitud a pulgadas.

Falla con un error si esta longitud tiene unidades em distintas de cero. Mirá el método pt[3.10.13/3.1] para más detalles.

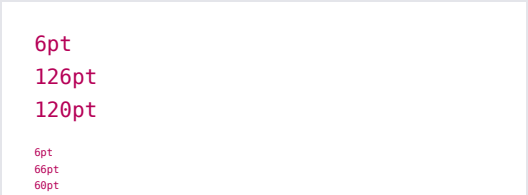
```
self.inches() → float
```

### /3.5 to-absolute Desde: 0.11.0

Resuelve esta longitud a una longitud absoluta.

```
#set text(size: 12pt)
#context [
  #(6pt).to-absolute() \
  #(6pt + 10em).to-absolute() \
  #(10em).to-absolute()
]

#set text(size: 6pt)
#context [
  #(6pt).to-absolute() \
  #(6pt + 10em).to-absolute() \
  #(10em).to-absolute()
]
```




```
self.to-absolute() → length
```

### 3.10.14 `measure` Contextual Desde: siempre

Mide el tamaño dispuesto del contenido.

La función `measure` te permite determinar el tamaño dispuesto de un contenido. Por defecto se supone un espacio infinito, por lo que las dimensiones medidas no necesariamente coinciden con las dimensiones finales del contenido. Si querés medir con las dimensiones de diseño actuales, podés combinar `measure` y `layout`[3.10.12].

#### /1 Ejemplo

El mismo contenido puede tener un tamaño distinto según el contexto[3.4] en el que se lo coloque. En el siguiente ejemplo, el `#content` es, por supuesto, más grande cuando aumentamos el tamaño de la fuente.

```
#let content = [Hello!]
#content
#set text(14pt)
#content
```

Hello! Hello!

Por este motivo, solo podés medir cuando hay contexto disponible.

```
#let thing(body) = context {
  let size = measure(body)
  [Width of "#body" is #size.width]
}

#thing[Hey] \
#thing[Welcome]
```

Width of “Hey” is 18.53pt
Width of “Welcome” is 41.26pt

La función measure devuelve un diccionario con las entradas `width`, `height` y `baseline`, todas de tipo `length`[3.10.13]. La entrada `baseline` es la posición vertical de la línea base del contenido relativa a su borde superior, con valores positivos que se extienden hacia abajo.

#### /2 Parámetros

```
measure(
  width: auto length,
  height: auto length,
  content,
) → dictionary
```

##### /2.1 `width` `auto` or `length` Por defecto: `auto`

El ancho disponible para disponer el contenido.

Establecerlo en `auto` indica ancho disponible infinito.

Tené en cuenta que usar los parámetros `width` y `height` de esta función es distinto de medir un `block`[3.10.4] con tamaño que contiene el contenido. En el siguiente ejemplo, lo primero obtendrá las dimensiones del contenido interior en lugar de las dimensiones del bloque.

```
#context measure(lorem(100), width: 400pt)

#context measure(block(lorem(100), width: 400pt))
```

```
(
  width: 395.89pt,
  height: 122.34pt,
  baseline: 7.24pt,
)
```

```
(width: 400pt, height: 122.34pt,
baseline: 7.24pt)
```

/2.2 **height** `auto` or `length` Por defecto: `auto`

La altura disponible para disponer el contenido.

Establecerla en `auto` indica altura disponible infinita.

/2.3 **content** `content` *Obligatorio* *Posicional*

El contenido cuyo tamaño se mide.

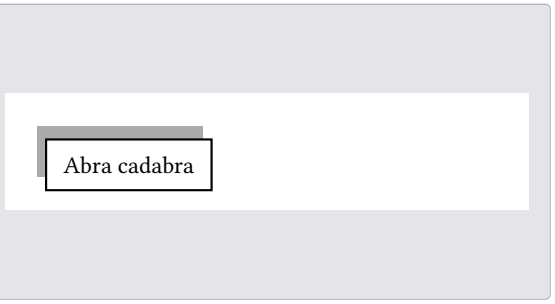
### 3.10.15 `move` Elemento Desde: siempre

Mueve contenido sin afectar el diseño.

La función `move` te permite mover contenido mientras el diseño todavía lo "ve" en las posiciones originales. Los contenedores seguirán teniendo el tamaño que tendrían si el contenido no se hubiera movido.

#### /1 Ejemplo

```
#rect(inset: 0pt, fill: gray, move(
  dx: 4pt, dy: 6pt,
  rect(
    inset: 8pt,
    fill: white,
    stroke: black,
    [Abra cadabra]
  )
))
```




#### /2 Accesibilidad

Mover es transparente para las tecnologías de asistencia (TA). Tu contenido se leerá en el orden en que aparece en el código fuente, sin importar ningún movimiento visual. Si necesitás ocultar contenido por completo a las TA en la exportación a PDF, considerá usar `pdf.artifact`[3.14.4/4].

#### /3 Parámetros

```
move(
  dx: relative,
  dy: relative,
  content,
) → content
```

/3.1 **dx** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El desplazamiento horizontal del contenido.

/3.2 **dy** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El desplazamiento vertical del contenido.

/3.3 **body** `content` *Obligatorio* *Posicional*

El contenido que se mueve.

### 3.10.16 pad Elemento Desde: siempre

Agrega espaciado alrededor del contenido.

El espaciado se puede especificar para cada lado individualmente o para todos los lados a la vez mediante un argumento posicional.

#### /1 Ejemplo

```
#set align(center)

#pad(x: 16pt, image("typing.jpg"))
_Typing speeds can be
 measured in words per minute._
```



*Typing speeds can be measured in words per minute.*

#### /2 Parámetros

```
pad(
  left: relative,
  top: relative,
  right: relative,
  bottom: relative,
  x: relative,
  y: relative,
  rest: relative,
  content,
) → content
```

/2.1 **left** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El relleno del lado izquierdo.

/2.2 **top** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El relleno del lado superior.

/2.3 **right** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El relleno del lado derecho.

/2.4 **bottom** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El relleno del lado inferior.

/2.5 **x** `relative` Por defecto: `0% + 0pt`

Un atajo para establecer `left` y `right` en el mismo valor.

/2.6 **y** `relative` Por defecto: `0% + 0pt`

Un atajo para establecer `top` y `bottom` en el mismo valor.

/2.7 **rest** `relative` Por defecto: `0% + 0pt`

Un atajo para establecer los cuatro lados en el mismo valor.

/2.8 **body** `content` *Obligatorio* *Posicional*

El contenido al que se le agrega relleno en los lados.

## 3.10.17 page Elemento Desde: siempre

Dispone su hijo en una o varias páginas.

Aunque esta función se usa principalmente en reglas set para afectar las propiedades de la página, también se puede usar para renderizar explícitamente su argumento en un conjunto de páginas propio.

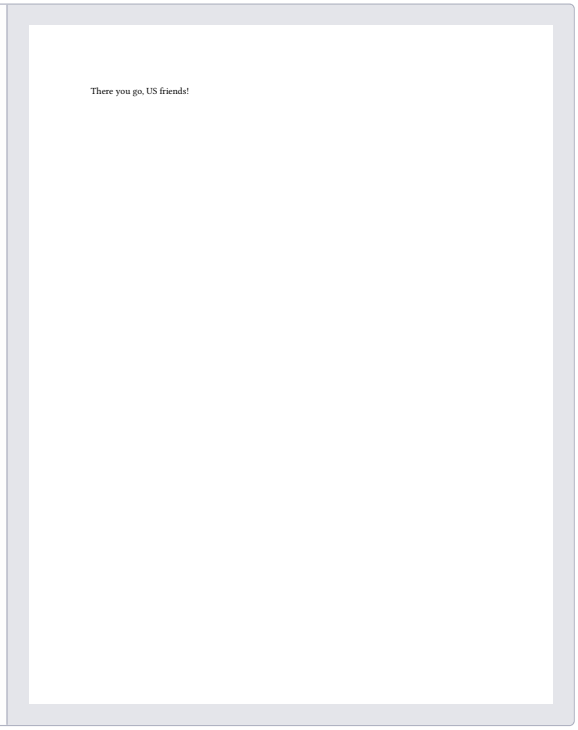
Las páginas se pueden configurar con `auto` como ancho o alto. En ese caso, las páginas crecerán para ajustarse a su contenido en el eje respectivo.

La guía de configuración de página[4.2] explica, con muchos ejemplos, cómo usar esta función y otras relacionadas para configurar un documento.

### /1 Ejemplo

```
#set page("us-letter")

There you go, US friends!
```




### /2 Accesibilidad

El contenido del encabezado, el pie, el primer plano y el fondo de la página es invisible para las tecnologías de asistencia (TA), como los lectores de pantalla. Las TA solo leen el cuerpo de la página. No incluyas en estas áreas información vital que no esté incluida en otro lugar del documento.

### /3 Estilos

Tené en cuenta que el elemento page[3.10.17] no puede ser objetivo de reglas show; escribir `show page: ..` no tiene efecto. Para repetir contenido en todas las páginas, podés configurar en cambio las propiedades `header`[3.10.17/4.13], `footer`[3.10.17/4.15], `background`[3.10.17/4.17] y `foreground`[3.10.17/4.18] con una regla set.

### /4 Parámetros

```
page(
  paper: str,
  width: auto length,
  height: auto length,
  flipped: bool,
  margin: auto relative dictionary,
  bleed: relative dictionary,
```

```
  binding: auto alignment,
  columns: int,
  fill: none auto color gradient tiling,
  numbering: none str function,
  supplement: none auto content,
  number-align: alignment,
  header: none auto content,
  header-ascent: relative,
  footer: none auto content,
  footer-descent: relative,
  background: none content,
  foreground: none content,
  content,
) → content
```

/4.1 **paper** `str` Por defecto: `"a4"`

Un tamaño de papel estándar para establecer el ancho y el alto.

Es solo un atajo para establecer `width` y `height` y, como tal, no se puede obtener en una expresión de contexto.

| Variante | Detalles |
|---|---|
| `"a0"` | Un papel de 841.0 × 1189.0 milímetros. |
| `"a1"` | Un papel de 594.0 × 841.0 milímetros. |
| `"a2"` | Un papel de 420.0 × 594.0 milímetros. |
| `"a3"` | Un papel de 297.0 × 420.0 milímetros. |
| `"a4"` | Un papel de 210.0 × 297.0 milímetros. |
| `"a5"` | Un papel de 148.0 × 210.0 milímetros. |
| `"a6"` | Un papel de 105.0 × 148.0 milímetros. |
| `"a7"` | Un papel de 74.0 × 105.0 milímetros. |
| `"a8"` | Un papel de 52.0 × 74.0 milímetros. |
| `"a9"` | Un papel de 37.0 × 52.0 milímetros. |
| `"a10"` | Un papel de 26.0 × 37.0 milímetros. |
| `"a11"` | Un papel de 18.0 × 26.0 milímetros. |
| `"iso-b1"` | Un papel de 707.0 × 1000.0 milímetros. |
| `"iso-b2"` | Un papel de 500.0 × 707.0 milímetros. |
| `"iso-b3"` | Un papel de 353.0 × 500.0 milímetros. |
| `"iso-b4"` | Un papel de 250.0 × 353.0 milímetros. |
| `"iso-b5"` | Un papel de 176.0 × 250.0 milímetros. |
| `"iso-b6"` | Un papel de 125.0 × 176.0 milímetros. |
| `"iso-b7"` | Un papel de 88.0 × 125.0 milímetros. |
| `"iso-b8"` | Un papel de 62.0 × 88.0 milímetros. |
| `"iso-c3"` | Un papel de 324.0 × 458.0 milímetros. |
| `"iso-c4"` | Un papel de 229.0 × 324.0 milímetros. |
| `"iso-c5"` | Un papel de 162.0 × 229.0 milímetros. |
| `"iso-c6"` | Un papel de 114.0 × 162.0 milímetros. |
| `"iso-c7"` | Un papel de 81.0 × 114.0 milímetros. |
| `"iso-c8"` | Un papel de 57.0 × 81.0 milímetros. |
| `"din-d3"` | Un papel de 272.0 × 385.0 milímetros. |
| `"din-d4"` | Un papel de 192.0 × 272.0 milímetros. |

| Variante | Detalles |
|---|---|
| `"din-d5"` | Un papel de 136.0 × 192.0 milímetros. |
| `"din-d6"` | Un papel de 96.0 × 136.0 milímetros. |
| `"din-d7"` | Un papel de 68.0 × 96.0 milímetros. |
| `"din-d8"` | Un papel de 48.0 × 68.0 milímetros. |
| `"sis-g5"` | Un papel de 169.0 × 239.0 milímetros. |
| `"sis-e5"` | Un papel de 115.0 × 220.0 milímetros. |
| `"ansi-a"` | Un papel de 216.0 × 279.0 milímetros. |
| `"ansi-b"` | Un papel de 279.0 × 432.0 milímetros. |
| `"ansi-c"` | Un papel de 432.0 × 559.0 milímetros. |
| `"ansi-d"` | Un papel de 559.0 × 864.0 milímetros. |
| `"ansi-e"` | Un papel de 864.0 × 1118.0 milímetros. |
| `"arch-a"` | Un papel de 229.0 × 305.0 milímetros. |
| `"arch-b"` | Un papel de 305.0 × 457.0 milímetros. |
| `"arch-c"` | Un papel de 457.0 × 610.0 milímetros. |
| `"arch-d"` | Un papel de 610.0 × 914.0 milímetros. |
| `"arch-e1"` | Un papel de 762.0 × 1067.0 milímetros. |
| `"arch-e"` | Un papel de 914.0 × 1219.0 milímetros. |
| `"jis-b0"` | Un papel de 1030.0 × 1456.0 milímetros. |
| `"jis-b1"` | Un papel de 728.0 × 1030.0 milímetros. |
| `"jis-b2"` | Un papel de 515.0 × 728.0 milímetros. |
| `"jis-b3"` | Un papel de 364.0 × 515.0 milímetros. |
| `"jis-b4"` | Un papel de 257.0 × 364.0 milímetros. |
| `"jis-b5"` | Un papel de 182.0 × 257.0 milímetros. |
| `"jis-b6"` | Un papel de 128.0 × 182.0 milímetros. |
| `"jis-b7"` | Un papel de 91.0 × 128.0 milímetros. |
| `"jis-b8"` | Un papel de 64.0 × 91.0 milímetros. |
| `"jis-b9"` | Un papel de 45.0 × 64.0 milímetros. |
| `"jis-b10"` | Un papel de 32.0 × 45.0 milímetros. |
| `"jis-b11"` | Un papel de 22.0 × 32.0 milímetros. |
| `"sac-d0"` | Un papel de 764.0 × 1064.0 milímetros. |
| `"sac-d1"` | Un papel de 532.0 × 760.0 milímetros. |
| `"sac-d2"` | Un papel de 380.0 × 528.0 milímetros. |
| `"sac-d3"` | Un papel de 264.0 × 376.0 milímetros. |
| `"sac-d4"` | Un papel de 188.0 × 260.0 milímetros. |
| `"sac-d5"` | Un papel de 130.0 × 184.0 milímetros. |
| `"sac-d6"` | Un papel de 92.0 × 126.0 milímetros. |
| `"iso-id-1"` | Un papel de 85.6 × 53.98 milímetros. |
| `"iso-id-2"` | Un papel de 74.0 × 105.0 milímetros. |
| `"iso-id-3"` | Un papel de 88.0 × 125.0 milímetros. |
| `"asia-f4"` | Un papel de 210.0 × 330.0 milímetros. |
| `"jp-shiroku-ban-4"` | Un papel de 264.0 × 379.0 milímetros. |
| `"jp-shiroku-ban-5"` | Un papel de 189.0 × 262.0 milímetros. |
| `"jp-shiroku-ban-6"` | Un papel de 127.0 × 188.0 milímetros. |
| `"jp-kiku-4"` | Un papel de 227.0 × 306.0 milímetros. |
| `"jp-kiku-5"` | Un papel de 151.0 × 227.0 milímetros. |
| `"jp-business-card"` | Un papel de 91.0 × 55.0 milímetros. |

| Variante | Detalles |
|---|---|
| `"cn-business-card"` | Un papel de 90.0 × 54.0 milímetros. |
| `"eu-business-card"` | Un papel de 85.0 × 55.0 milímetros. |
| `"fr-tellière"` | Un papel de 340.0 × 440.0 milímetros. |
| `"fr-couronne-écriture"` | Un papel de 360.0 × 460.0 milímetros. |
| `"fr-couronne-édition"` | Un papel de 370.0 × 470.0 milímetros. |
| `"fr-raisin"` | Un papel de 500.0 × 650.0 milímetros. |
| `"fr-carré"` | Un papel de 450.0 × 560.0 milímetros. |
| `"fr-jésus"` | Un papel de 560.0 × 760.0 milímetros. |
| `"uk-brief"` | Un papel de 406.4 × 342.9 milímetros. |
| `"uk-draft"` | Un papel de 254.0 × 406.4 milímetros. |
| `"uk-foolscap"` | Un papel de 203.2 × 330.2 milímetros. |
| `"uk-quarto"` | Un papel de 203.2 × 254.0 milímetros. |
| `"uk-crown"` | Un papel de 508.0 × 381.0 milímetros. |
| `"uk-book-a"` | Un papel de 111.0 × 178.0 milímetros. |
| `"uk-book-b"` | Un papel de 129.0 × 198.0 milímetros. |
| `"us-letter"` | Un papel de 215.9 × 279.4 milímetros. |
| `"us-legal"` | Un papel de 215.9 × 355.6 milímetros. |
| `"us-tabloid"` | Un papel de 279.4 × 431.8 milímetros. |
| `"us-executive"` | Un papel de 184.15 × 266.7 milímetros. |
| `"us-foolscap-folio"` | Un papel de 215.9 × 342.9 milímetros. |
| `"us-statement"` | Un papel de 139.7 × 215.9 milímetros. |
| `"us-ledger"` | Un papel de 431.8 × 279.4 milímetros. |
| `"us-oficio"` | Un papel de 215.9 × 340.36 milímetros. |
| `"us-gov-letter"` | Un papel de 203.2 × 266.7 milímetros. |
| `"us-gov-legal"` | Un papel de 215.9 × 330.2 milímetros. |
| `"us-business-card"` | Un papel de 88.9 × 50.8 milímetros. |
| `"us-digest"` | Un papel de 139.7 × 215.9 milímetros. |
| `"us-trade"` | Un papel de 152.4 × 228.6 milímetros. |
| `"newspaper-compact"` | Un papel de 280.0 × 430.0 milímetros. |
| `"newspaper-berliner"` | Un papel de 315.0 × 470.0 milímetros. |
| `"newspaper-broadsheet"` | Un papel de 381.0 × 578.0 milímetros. |
| `"presentation-16-9"` | Un papel de 297.0 × 167.0625 milímetros. |
| `"presentation-4-3"` | Un papel de 280.0 × 210.0 milímetros. |

/4.2 **`width`** `auto` or `length` *Configurable con set* Por defecto: `595.28pt`

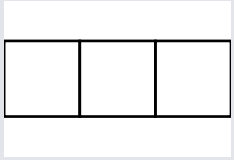
El ancho de la página final, después de aplicar los recortes.

En las configuraciones de impresión profesional, puede ser menor que el tamaño de la hoja que se le da a la impresora.

Mirá el parámetro `bleed`[3.10.17/4.6] para más detalles sobre cómo establecer un área de recorte o de sangrado.

```
#set page(
  width: 3cm,
  margin: (x: 0cm),
)

#for i in range(3) {
  box(square(width: 1cm))
}
```



/4.3 **height** `auto` or `length` *Configurable con set* Por defecto: `841.89pt`

La altura del área final de la página, después de aplicar los recortes.

Si se establece en `auto`, los saltos de página solo se pueden provocar manualmente insertando un salto de página[3.10.18] o agregando otra regla set de página no vacía. La mayoría de los ejemplos de esta documentación usan `auto` como altura de página para que crezca y se reduzca dinámicamente según su contenido.

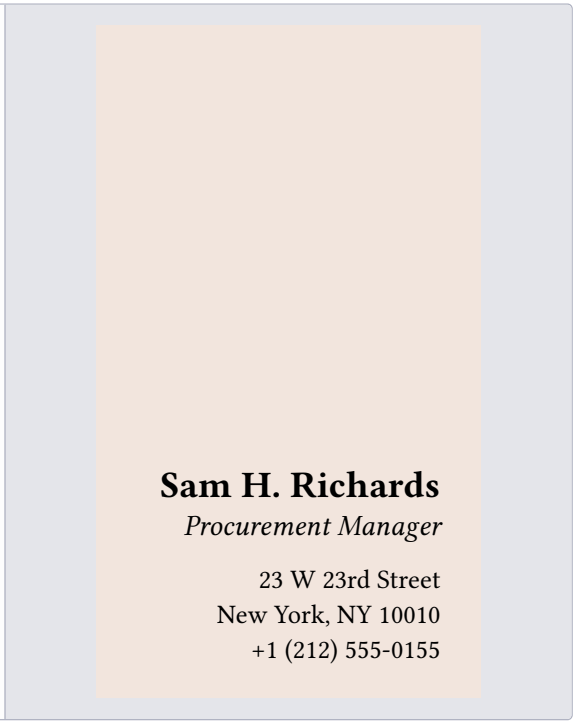
/4.4 **flipped** `bool` *Configurable con set* Por defecto: `false`

Si la página se gira a orientación horizontal.

```
#set page(
  "us-business-card",
  flipped: true,
  fill: rgb("f2e5dd"),
)

#set align(bottom + end)
#text(14pt)[*Sam H. Richards*] \
_Procurement Manager_

#set text(10pt)
23 W 23rd Street \
New York, NY 10010 \
+1 (212) 555-0155
```




/4.5 **margin** `auto` or `relative` or `dictionary` *Configurable con set* Por defecto: `auto`

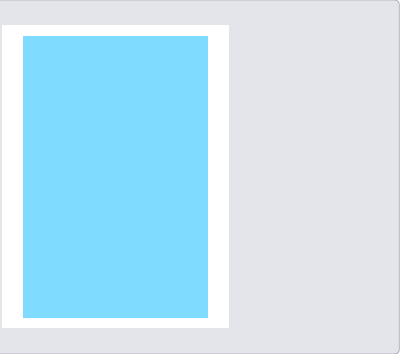
Los márgenes de la página.

- `auto`: Los márgenes se establecen automáticamente en 2,5/21 veces la menor dimensión de la página. Esto da márgenes de 2,5 cm en una página A4.
- Una única longitud: el mismo margen en todos los lados.
- Un diccionario: con un diccionario, los márgenes se pueden establecer individualmente. El diccionario puede contener las siguientes claves, en orden de precedencia:
  - `top`: El margen superior.
  - `right`: El margen derecho.
  - `bottom`: El margen inferior.
  - `left`: El margen izquierdo.
  - `inside`: El margen del lado interior de la página (donde está la encuadernación[3.10.17/4.7]).
  - `outside`: El margen del lado exterior de la página (opuesto a la encuadernación[3.10.17/4.7]).
  - `x`: Los márgenes horizontales.
  - `y`: Los márgenes verticales.
  - `rest`: Los márgenes de todos los lados excepto aquellos para los que el diccionario establece explícitamente un tamaño.

Todas las claves son opcionales; las claves omitidas usarán su valor establecido previamente o el margen por defecto si nunca se establecieron. Además, los valores de `left` y `right` son mutuamente excluyentes con los de `inside` y `outside`. Los valores deben ser longitudes relativas o `auto`.

```
#set page(
 width: 3cm,
 height: 4cm,
 margin: (x: 8pt, y: 4pt),
)

#rect(
  width: 100%,
  height: 100%,
  fill: aqua,
)
```



/4.6 **bleed** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

El margen de sangrado de la página.

El sangrado es el área de contenido que se extiende más allá del tamaño final recortado de la página. Asegura que no aparezcan bordes sin imprimir en el producto final, incluso si hay pequeños desajustes en el recorte.

Valores aceptados:

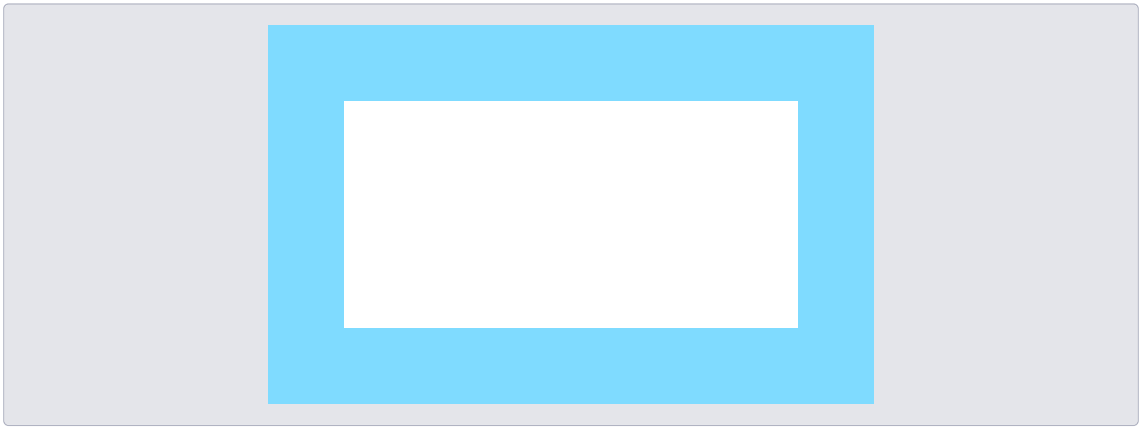
- Una única longitud: el mismo sangrado en todos los lados.
- Un diccionario: con un diccionario, los márgenes de sangrado se pueden establecer individualmente. El diccionario puede incluir las siguientes claves, enumeradas en orden de precedencia:
  - `top`: El margen de sangrado superior.
  - `right`: El margen de sangrado derecho.
  - `bottom`: El margen de sangrado inferior.
  - `left`: El margen de sangrado izquierdo.
  - `inside`: El margen de sangrado del lado interior de la página (donde está la encuadernación[3.10.17/4.7]).
  - `outside`: El margen de sangrado del lado exterior de la página (opuesto a la encuadernación[3.10.17/4.7]).
  - `x`: Los márgenes de sangrado horizontales.
  - `y`: Los márgenes de sangrado verticales.
  - `rest`: Los márgenes de sangrado de todos los lados excepto aquellos para los que el diccionario establece explícitamente un tamaño.

Todas las claves son opcionales; las claves omitidas usarán su valor establecido previamente o `0pt` si nunca se establecieron. Además, los valores de `left` y `right` son mutuamente excluyentes con los de `inside` y `outside`. Los valores deben ser longitudes relativas.

En la exportación a PDF, si el sangrado no es cero, se define un `TrimBox` para la página.

```
#set page(
  width: 8cm,
  height: 5cm,
  margin: 1cm,
  // The bleed is not visible in the preview;
  // it exceeds beyond the page.
  bleed: 0.5cm,
  // Fills the entire bleed area, so there will
  // be no white strips after printing and trimming.
  background: rect(width: 100%, height: 100%, fill: aqua),
)

#rect(width: 100%, height: 100%, fill: white)
```



### /4.7 binding `auto` or `alignment` *Configurable con set* Por defecto: auto

De qué lado se encuadernarán las páginas.

- auto: Equivale a `left` si la dirección del texto[3.7.12/2.19] es de izquierda a derecha y a `right` si es de derecha a izquierda.
- `left`: Encuadernación del lado izquierdo.
- `right`: Encuadernación del lado derecho.

Esto afecta el significado de las opciones `inside` y `outside` de los márgenes.

### /4.8 columns `int` *Configurable con set* Por defecto: 1

Cuántas columnas tiene la página.

Si necesitás insertar columnas en una página u otro contenedor, también podés usar la función `columns`[3.10.7].

```
#set page(columns: 2, height: 4.8cm)
Climate change is one of the most
pressing issues of our time, with
the potential to devastate
communities, ecosystems, and
economies around the world. It's
clear that we need to take urgent
action to reduce our carbon
emissions and mitigate the impacts
of a rapidly changing climate.
```

Climate change is one of the most pressing issues of our time, with the potential to devastate communities, ecosystems, and economies around the world. It's clear that we need to take urgent action to reduce our carbon emissions and mitigate the impacts

### /4.9 fill `none` or `auto` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: auto

El relleno de fondo de la página.

Establecerlo en algo no transparente le indica a la impresora que coloree la página completa. Si estás considerando tiradas de producción más grandes, puede ser más amigable con el ambiente y más económico conseguir páginas teñidas de antemano y no establecer esta propiedad.

Cuando se establece en none, el fondo se vuelve transparente. Tené en cuenta que las páginas PDF igualmente aparecerán con un fondo (generalmente blanco) en los visores, pero en realidad son transparentes. (Si las imprimís, no se usa ningún color para el fondo.)

El valor por defecto auto resulta en none para la salida PDF y en `white` para PNG y SVG.

```
#set page(fill: rgb("444352"))
#set text(fill: rgb("fdfdfd"))
*Dark mode enabled.*
```

**Dark mode enabled.**

### /4.10 numbering `none` or `str` or `function` *Configurable con set* Por defecto: none

Cómo numerar las páginas. Podés consultar la guía de configuración de página sobre cómo personalizar los números de página[4.2.3].

Acepta una función o patrón de numeración[3.6.12] que toma uno o dos números:

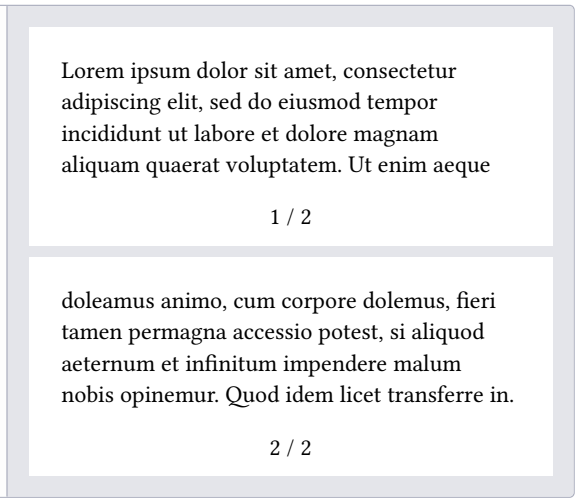
1. El primer número es el número de la página actual.
2. El segundo número es la cantidad total de páginas. En un patrón de numeración, el segundo número se puede omitir. Si se pasa una función, recibirá un argumento en el contexto de enlaces o referencias y dos argumentos al producir los números de página visibles.

Estos son números lógicos controlados por el contador de páginas y, por lo tanto, pueden no coincidir con los números físicos. Específicamente, son el valor actual[3.12.1/11.1] y el valor final[3.12.1/11.4] de `counter(page)`. Mirá la documentación de `counter`[3.12.1/5] para más detalles.

Si se da un `footer`[3.10.17/4.15] explícito (o un `header`[3.10.17/4.13] en el caso de numeración alineada arriba[3.10.17/4.12]), la numeración se ignora.

```
#set page(
  height: 100pt,
  margin: (top: 16pt, bottom: 24pt),
  numbering: "1 / 1",
)

#lorem(48)
```




/4.11 **supplement** `none` or `auto` or `content` *Configurable con set* Por defecto: `auto`

Un suplemento para las páginas.

En las referencias a páginas, se agrega antes del número de página.

```
#set page(numbering: "1.", supplement: [p.])

= Introduction <intro>
We are on #ref(<intro>, form: "page")!
```




/4.12 **number-align** `alignment` *Configurable con set* Por defecto: `center + bottom`

La alineación de la numeración de páginas.

Si el componente vertical es `top`, la numeración se coloca en el encabezado y, si es `bottom`, se coloca en el pie de página. La alineación horizon está prohibida. Si se da un `header` o `footer` explícito que coincida, la numeración se ignora.

```
#set page(
  margin: (top: 16pt, bottom: 24pt),
  numbering: "1",
  number-align: right,
)

#lorem(30)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

1

/4.13 **header** `none` or `auto` or `content` *Configurable con set* Por defecto: `auto`

El encabezado de la página. Llena el margen superior de cada página.

- Contenido: Muestra el contenido como encabezado.
- `auto`: Muestra el número de página si se estableció una `numbering`[3.10.17/4.10] y `number-align`[3.10.17/4.12] es `top`.
- `none`: Suprime el encabezado.

```
#set par(justify: true)
#set page(
  margin: (top: 32pt, bottom: 20pt),
  header: [
    #set text(8pt)
    #smallcaps[Typst Academy]
    #h(1fr) _Exercise Sheet 3_
  ],
)

#lorem(19)
```

Typst Academy *Exercise Sheet 3*

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam.

/4.14 **header-ascent** `relative` *Configurable con set* Por defecto: `30% + 0pt`

Cuánto se sube el encabezado hacia el margen superior. Las proporciones son relativas a la altura del margen superior.

/4.15 **footer** `none` or `auto` or `content` *Configurable con set* Por defecto: `auto`

El pie de página. Llena el margen inferior de cada página.

- Contenido: Muestra el contenido como pie de página.
- `auto`: Muestra el número de página si se estableció una `numbering`[3.10.17/4.10] y `number-align`[3.10.17/4.12] es `bottom`.
- `none`: Suprime el pie de página.

Para un simple número de página, normalmente alcanza con la propiedad `numbering`. Si querés crear un pie de página personalizado pero seguir mostrando el número de página, podés acceder directamente al contador de páginas[3.12.1].

```
#set par(justify: true)
#set page(
  height: 100pt,
  margin: 20pt,
  footer: context [
    #set align(right)
    #set text(8pt)
    #counter(page).display(
      "1 of I",
      both: true,
    )
  ]
)

#lorem(48)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus

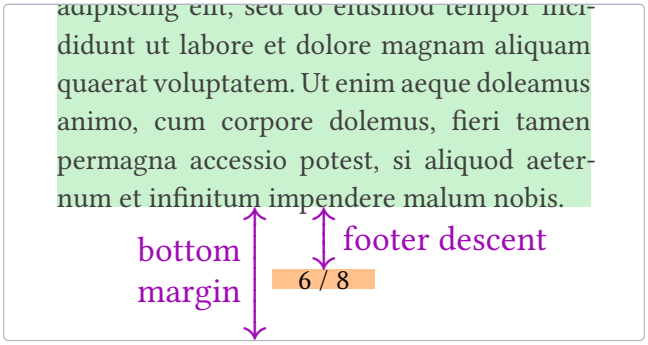
1 of II

animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in.

2 of II

/4.16 **footer-descent** `relative` *Configurable con set* Por defecto: `30% + 0pt`

Cuánto se baja el pie de página hacia el margen inferior. Las proporciones son relativas a la altura del margen inferior.




/4.17 **background** `none` or `content` *Configurable con set* Por defecto: none

Contenido en el fondo de la página.

Este contenido se colocará detrás del cuerpo de la página. Se puede usar para colocar una imagen de fondo o una marca de agua.

Por comodidad, las longitudes relativas[3.10.21] se resuelven respecto del tamaño de la página incluido el `bleed` de la página[3.10.17/4.6] cuando se usan en el contenido de fondo. Por ejemplo, en una página de `100mm` de ancho con un sangrado de `5mm`, un ancho de `100%` se calcula como `5mm + 100mm + 5mm`.

```
#set page(background: rotate(24deg,
  text(18pt, fill: rgb("FFCBC4"))[
    *CONFIDENTIAL*
  ]
))

= Typst's secret plans
In the year 2023, we plan to take
over the world (of typesetting).
```




/4.18 **foreground** `none` or `content` *Configurable con set* Por defecto: none

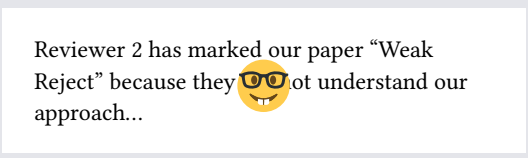
Contenido en el primer plano de la página.

Este contenido se superpondrá al cuerpo de la página.

Las longitudes relativas se resuelven respecto del tamaño de la página incluido el `bleed`[3.10.17/4.6], con el mismo comportamiento que `background`[3.10.17/4.17].

```
#set page(foreground: text(24pt)[🤓])

Reviewer 2 has marked our paper
"Weak Reject" because they did
not understand our approach...
```



/4.19 **body** `content` *Obligatorio* *Posicional*

El contenido de la(s) página(s).

Se crearán varias páginas si el contenido no entra en una sola. Después de componer el cuerpo, se creará una página nueva con las propiedades de página anteriores a la invocación de la función.

## 3.10.18 pagebreak Elemento Desde: siempre

Un salto de página manual.

No debe usarse dentro de ningún contenedor.

### /1 Ejemplo

```
The next page contains
more details on compound theory.
#pagebreak()

== Compound Theory
In 1984, the first ...
```

The next page contains more details on compound theory.

**Compound Theory**
In 1984, the first ...

Incluso sin saltos de página manuales, el contenido se paginará automáticamente según el tamaño de página configurado. Podés establecer la altura de la página[3.10.17/4.3] en `auto` para que la página crezca dinámicamente hasta que ocurra un salto de página manual.

La paginación intenta evitar líneas de texto sueltas al principio o al final de una página (se llaman *viudas* y *huérfanas*). Podés ajustar el parámetro `text.costs`[3.7.12/2.21] para desactivar este comportamiento.

### /2 Parámetros

```
pagebreak(
  weak: bool,
  to: none str,
) → content
```

#### /2.1 weak bool *Configurable con set* Por defecto: false

Si es `true`, el salto de página se omite si la página actual ya está vacía.

#### /2.2 to none or str *Configurable con set* Por defecto: none

Si se indica, asegura que la página siguiente sea par o impar, con una página vacía entremedio si es necesario.

```
#set page(height: 30pt)

First.
#pagebreak(to: "odd")
Third.
```

First.

Third.

| Variante | Detalles |
|---|---|
| "even" | La página siguiente será una página par. |
| "odd" | La página siguiente será una página impar. |

## 3.10.19 `place` Elemento Desde: siempre

Ubica contenido en relación con su contenedor padre.

El contenido ubicado puede estar superpuesto (por defecto) o flotante. El contenido superpuesto se alinea con el contenedor padre según la `alignment`[3.10.19/4.1] indicada y se muestra sobre cualquier otro contenido agregado hasta el momento en el contenedor. El contenido flotante se ubica en la parte superior o inferior del contenedor y desplaza el resto del contenido hacia abajo o hacia arriba, respectivamente. En ambos casos, la posición del contenido se puede ajustar con los desplazamientos `dx`[3.10.19/4.5] y `dy`[3.10.19/4.6] sin afectar el diseño.

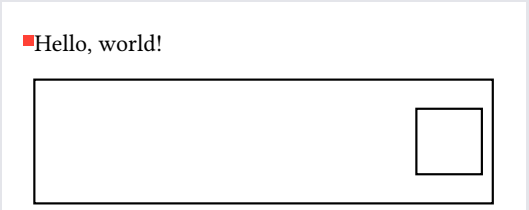
El padre puede ser cualquier contenedor, como un `block`[3.10.4], un `box`[3.10.5], un `rect`[3.11.10], etcétera. Una llamada a `place` de nivel superior ubicará el contenido directamente en el área de texto de la página actual. Esto se puede usar para posicionamiento absoluto en la página: con una `alignment`[3.10.19/4.1] `top + left`, los desplazamientos `dx` y `dy` establecerán la posición de la esquina superior izquierda del elemento en relación con la esquina superior izquierda del área de texto. Para posicionamiento absoluto en toda la página, incluidos los márgenes, podés usar `place` en `page.foreground`[3.10.17/4.18] o `page.background`[3.10.17/4.17].

### /1 Ejemplos

```
#set page(height: 120pt)
Hello, world!

#rect(
  width: 100%,
  height: 2cm,
  place(horizon + right, square()),
)

#place(
  top + left,
  dx: -5pt,
  square(size: 5pt, fill: red),
)
```




### /2 Efecto en la posición de otros elementos

Los elementos superpuestos no ocupan espacio en el flujo del contenido, pero una llamada a `place` inserta un elemento invisible de bloque en el flujo. Esto puede afectar el diseño al cortar el párrafo actual. Para evitarlo, podés envolver la llamada a `place` en un `box`[3.10.5] cuando la llamada se hace en medio de un párrafo. La alineación y los desplazamientos serán entonces relativos a este box de tamaño cero. Para asegurarte de que no interfiera con el espaciado, el box debería unirse a una palabra mediante un word joiner.

Por ejemplo, lo siguiente define una función para adjuntar una anotación a la palabra siguiente:

```
#let annotate(..args) = {
  box(place(..args))
  sym.wj
  h(0pt, weak: true)
}

A placed #annotate(square(), dy: 2pt)
square in my text.
```




El espaciado débil de ancho cero sirve para descartar los espacios entre la llamada a la función y la palabra siguiente.

### /3 Accesibilidad

Las tecnologías de asistencia (TA) siempre leerán el elemento ubicado en el punto donde aparece lógicamente en el documento, sin importar adónde lo haya movido físicamente esta función. Poné su marcado donde tenga más sentido en el orden de lectura.

## /4 Parámetros

```
place(
  auto alignment,
  scope: str,
  float: bool,
  clearance: length,
  dx: relative,
  dy: relative,
  content,
) → content
```

### /4.1 alignment — auto or alignment — *Posicional* — *Configurable con set* — Por defecto: start

En relación con qué posición del contenedor padre se ubica el contenido.

- Si `float` es `false`, puede ser cualquier alineación distinta de `auto`.
- Si `float` es `true`, debe ser `auto`, `top` o `bottom`.

Cuando `float` es `false` y no se especifica una alineación vertical, el contenido se ubica en la posición actual del eje vertical.

### /4.2 scope — str — *Configurable con set* — Por defecto: "column"

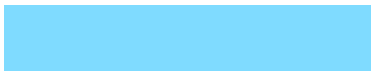
En relación con qué ámbito contenedor se ubica algo.

El ámbito padre se usa principalmente con figuras y, por este motivo, la función figure tiene un parámetro scope[3.6.7/5.4] espejo. De todos modos, también puede ser útil en general para salir de las columnas. Un ejemplo típico sería crear una sección de título de una sola columna[4.2.4] en un documento de dos columnas.

Tené en cuenta que la ubicación con ámbito en el padre actualmente solo se admite si `float` es `true`. Esto podría cambiar en el futuro.

```
#set page(height: 150pt, columns: 2)
#place(
  top + center,
  scope: "parent",
  float: true,
  rect(width: 80%, fill: aqua),
)

#lorem(25)
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus.

| Variante | Detalles |
|---|---|
| "column" | Se ubica en la columna actual. |
| "parent" | Se ubica respecto del elemento padre, y el contenido abarca todas las columnas. |

### /4.3 float — bool — *Configurable con set* — Por defecto: false

Si el elemento ubicado tiene un diseño flotante.

Los elementos flotantes se posicionan en la parte superior o inferior del contenedor padre y desplazan el contenido en flujo. Siempre se ubican en el orden del flujo entre sí, y antes de cualquier contenido que siga a un elemento `place.flush`[3.10.19/5.1] posterior.

```
#set page(height: 150pt)
#let note(where, body) = place(
  center + where,
  float: true,
  clearance: 6pt,
  rect(body),
)

#lorem(10)
#note(bottom)[Bottom 1]
#note(bottom)[Bottom 2]
#lorem(40)
#note(top)[Top]
#lorem(10)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam

Bottom 1

Bottom 2

Top

aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

/4.4 **clearance** `length` *Configurable con set* Por defecto: `1.5em`

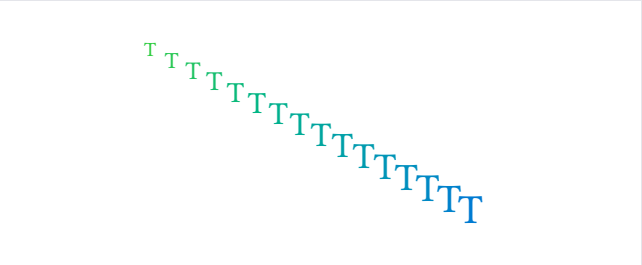
El espaciado entre el elemento ubicado y otros elementos en un diseño flotante.

No tiene efecto si `float` es `false`.

/4.5 **dx** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El desplazamiento horizontal del contenido ubicado.

```
#set page(height: 100pt)
#for x in range(-8, 8) {
  place(center + horizon,
    dx: x * 8pt, dy: x * 4pt,
    text(
      size: calc.root(x + 10, 3) * 6pt,
      fill: color.mix((green, 8 - x), (blue, 8 + x)),
    )[T]
  )
}
```



Esto no afecta el diseño del contenido en flujo. En otras palabras, el contenido ubicado se trata como si estuviera envuelto en un elemento `move`[3.10.15].

/4.6 **dy** `relative` *Configurable con set* Por defecto: `0% + 0pt`

El desplazamiento vertical del contenido ubicado.

Esto no afecta el diseño del contenido en flujo. En otras palabras, el contenido ubicado se trata como si estuviera envuelto en un elemento move[3.10.15].

/4.7 **body** `content` *Obligatorio* *Posicional*

El contenido que se ubica.

## /5 Funciones

/5.1 **`flush`** Elemento Desde: 0.12.0

Le pide al algoritmo de diseño que ubique los elementos flotantes pendientes antes de continuar con el contenido.

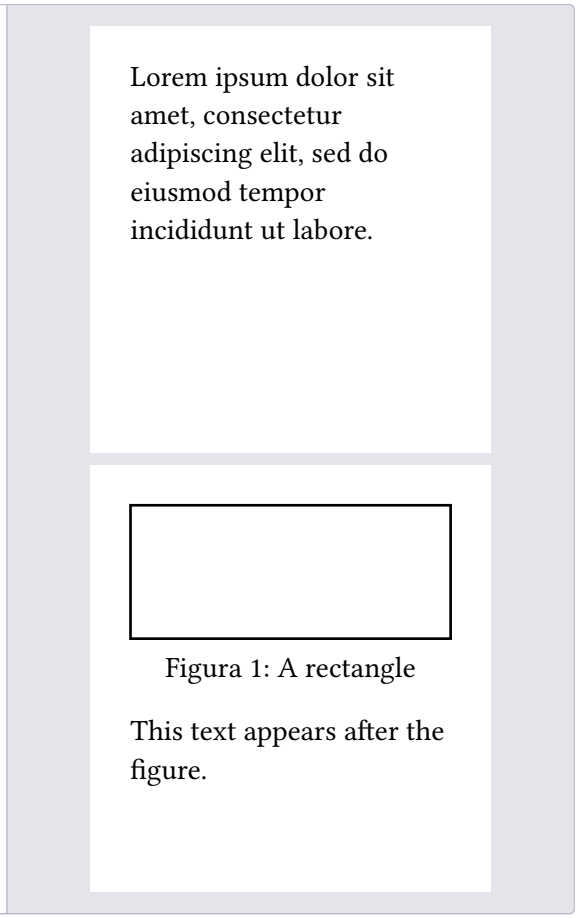
Esto es útil para evitar que las figuras flotantes se desborden a la sección siguiente.

```
#lorem(15)

#figure(
  rect(width: 100%, height: 50pt),
  placement: auto,
  caption: [A rectangle],
)

#place.flush()

This text appears after the figure.
```



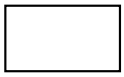

```
place.flush() → content
```

### 3.10.20 `ratio` Desde: siempre

Una proporción de un todo.

Una proporción se escribe como un número seguido de un signo de porcentaje. Las proporciones aparecen con más frecuencia como parte de una longitud relativa[3.10.21], para especificar el tamaño de algún elemento de diseño relativo a la página o a algún contenedor.

```
#rect(width: 25%)
```



Sin embargo, también pueden describir cualquier otra propiedad que sea relativa a una base, por ejemplo, una cantidad de escalado horizontal[3.10.24/2.2] o la altura de los paréntesis[3.8.20/2.1.1] relativa a la altura del contenido que encierran.

#### /1 Scripting

En tu propio código, podés usar las proporciones como quieras. Podés multiplicarlas por varios otros tipos, como se muestra a continuación:

| Multiplicar por | Ejemplo | Resultado |
|---|---|---|
| `ratio`[3.10.20] | `27% * 10%` | `2.7%` |
| `length`[3.10.13] | `27% * 100pt` | `27pt` |
| `relative`[3.10.21] | `27% * (10% + 100pt)` | `2.7% + 27pt` |
| `angle`[3.10.3] | `27% * 100deg` | `27deg` |
| `int`[3.5.17] | `27% * 2` | `54%` |
| `float`[3.5.15] | `27% * 0.37037` | `10%` |
| `fraction`[3.10.9] | `27% * 3fr` | `0.81fr` |

Cuando las proporciones se muestran[3.5.25] en el documento, se redondean a dos cifras significativas para mayor legibilidad.

### 3.10.21 `relative` Desde: siempre

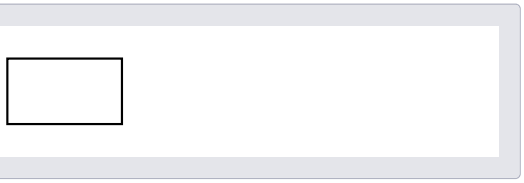
Una longitud en relación con alguna longitud conocida.

Este tipo es una combinación de una longitud[3.10.13] con una proporción[3.10.20]. Resulta de la suma y resta de una longitud y una proporción. Dondequiera que se espere una longitud relativa, también podés usar una longitud o una proporción a secas.

#### /1 Relativa a la página

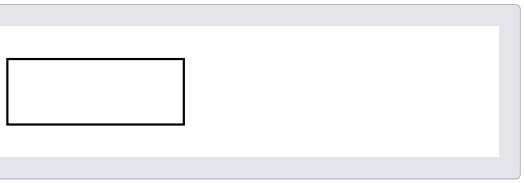
Un caso de uso común es establecer el ancho o el alto de un elemento de diseño (por ejemplo, block[3.10.4], rect[3.11.10], etcétera) como cierto porcentaje del ancho de la página. Acá, el ancho del rectángulo se establece en `25%`, así que ocupa un cuarto del ancho *interior* de la página (el ancho menos los márgenes).

```
#rect(width: 25%)
```



Las longitudes o proporciones a secas siempre son válidas donde se esperan longitudes relativas, pero las dos también se pueden mezclar libremente:

```
#rect(width: 25% + 1cm)
```



Para el contenido del `background`[3.10.17/4.17] y el `foreground`[3.10.17/4.18] de la página, las longitudes relativas se resuelven respecto del tamaño de la página incluido el `bleed`[3.10.17/4.6]. Esta elección se hace por comodidad, ya que crear un fondo que tenga bien en cuenta el sangrado requiere, inherentemente, extenderlo hacia el área de sangrado.

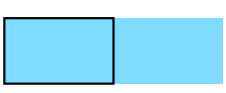
Si estás intentando dar tamaño a un elemento para que ocupe el ancho *completo* de la página, tenés algunas opciones (esto depende mucho de tu caso de uso exacto):

1. Establecer los márgenes de página en `0pt` (`#set page(margin: 0pt)`)
2. Multiplicar la proporción por el ancho completo conocido de la página (`21cm * 69%`)
3. Usar relleno que anule los márgenes (`#pad(x: -2.5cm, ..)`)
4. Usar el campo `background`[3.10.17/4.17] o `foreground`[3.10.17/4.18] de la página, ya que no tienen en cuenta los márgenes (tené en cuenta que renderizará el contenido fuera del flujo del documento; mirá `place`[3.10.19] para controlar la posición del contenido)

#### /2 Relativa a un contenedor

Cuando un elemento de diseño (por ejemplo, un rect[3.11.10]) está anidado en otro contenedor de diseño (por ejemplo, un block[3.10.4]) en lugar de ser un descendiente directo de la página, los anchos relativos pasan a ser relativos al contenedor:

```
#block(
  width: 100pt,
  fill: aqua,
  rect(width: 50%),
)
```



#### /3 Scripting

Podés multiplicar longitudes relativas por proporciones[3.10.20], enteros[3.5.17] y flotantes[3.5.15].

Una longitud relativa tiene los siguientes campos:

- `length`: Su componente de longitud[3.10.13].
- `ratio`: Su componente de proporción[3.10.20].

```
#(100% - 50pt).length \
#(100% - 50pt).ratio
```

```
-50pt
100%
```

## 3.10.22 `repeat` Elemento Desde: siempre

Repite contenido hasta llenar el espacio disponible.

Esto puede ser útil al implementar un índice, una referencia o un índice de contenidos personalizados.

Se puede insertar espacio entre las instancias del parámetro body, así que asegurate de ajustar el parámetro justify[3.10.22/3.3] en consecuencia.

Da un error si no hay límites en el espacio disponible, ya que crearía contenido infinito.

### /1 Ejemplo

```
Sign on the dotted line:
#box(width: 1fr, repeat[.])

#set text(10pt)
#v(8pt, weak: true)
#align(right)[
  Berlin, the 22nd of December, 2022
]
```

Sign on the dotted line: .....
Berlin, the 22nd of December, 2022

### /2 Accesibilidad

El contenido repetido se marca automáticamente como artefacto[3.14.4/4] y se oculta a las tecnologías de asistencia (TA). No uses esta función para crear contenido que contribuya al significado de tu documento.

### /3 Parámetros

```
repeat(
  content,
  gap: length,
  justify: bool,
) → content
```

#### /3.1 **body** `content` *Obligatorio* *Posicional*

El contenido que se repite.

#### /3.2 **gap** `length` *Configurable con set* Por defecto: `0pt`

El espacio entre cada instancia del cuerpo.

#### /3.3 **justify** `bool` *Configurable con set* Por defecto: `true`

Si se aumenta el espacio entre las instancias para llenar por completo el espacio disponible.

## 3.10.23 `rotate` Elemento Desde: siempre

Rota contenido sin afectar el diseño.

Rota un elemento un ángulo dado. El diseño actuará como si el elemento no estuviera rotado, salvo que especifiques `reflow: true`.

### /1 Ejemplo

```
#stack(
  dir: ltr,
  spacing: 1fr,
  ..range(16)
    .map(i => rotate(24deg * i)[X]),
)
```



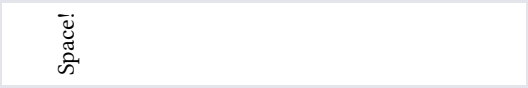
### /2 Parámetros

```
rotate(
  angle,
  origin: alignment,
  reflow: bool,
  content,
) → content
```

#### /2.1 `angle` angle *Posicional* *Configurable con set* Por defecto: `0deg`

La cantidad de rotación.

```
#rotate(-1.571rad)[Space!]
```



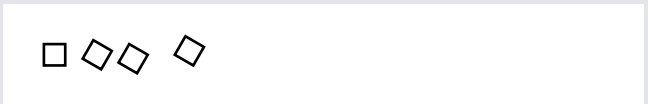

#### /2.2 `origin` alignment *Configurable con set* Por defecto: `center + horizon`

El origen de la rotación.

Si, por ejemplo, quisieras que la esquina inferior izquierda del elemento rotado siguiera alineada con la línea base, lo establecerías en `bottom + left`.

```
#set text(spacing: 8pt)
#let square = square.with(width: 8pt)

#box(square())
#box(rotate(30deg, origin: center, square()))
#box(rotate(30deg, origin: top + left, square()))
#box(rotate(30deg, origin: bottom + right, square()))
```

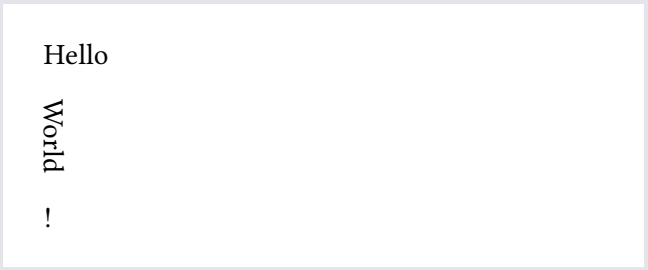


#### /2.3 `reflow` bool *Configurable con set* Por defecto: `false`

Si la rotación afecta el diseño.

Si se establece en `false`, el contenido rotado conservará el cuadro delimitador del contenido original. Si se establece en `true`, el cuadro delimitador tendrá en cuenta la rotación del contenido y ajustará el diseño en consecuencia.

```
Hello #rotate(90deg, reflow: true)[World]!
```




/2.4 **body** `content` *Obligatorio* *Posicional*

El contenido que se rota.

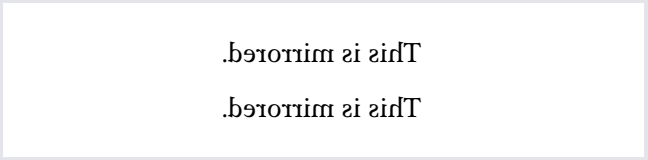
## 3.10.24 `scale` Elemento Desde: siempre

Escala contenido sin afectar el diseño.

Te permite reflejar contenido especificando una escala negativa en un solo eje.

### /1 Ejemplo

```
#set align(center)
#scale(x: -100%)[This is mirrored.]
#scale(x: -100%, reflow: true)[This is mirrored.]
```




### /2 Parámetros

```
scale(
  auto length ratio,
  x: auto length ratio,
  y: auto length ratio,
  origin: alignment,
  reflow: bool,
  content,
) → content
```

#### /2.1 **factor** `auto` or `length` or `ratio` *Posicional* Por defecto: `100%`

El factor de escala para ambos ejes, como argumento posicional. Es solo una notación abreviada opcional para establecer x e y en el mismo valor.

#### /2.2 **x** `auto` or `length` or `ratio` *Configurable con set* Por defecto: `100%`

El factor de escala horizontal.

El cuerpo se reflejará horizontalmente si el parámetro es negativo.

#### /2.3 **y** `auto` or `length` or `ratio` *Configurable con set* Por defecto: `100%`

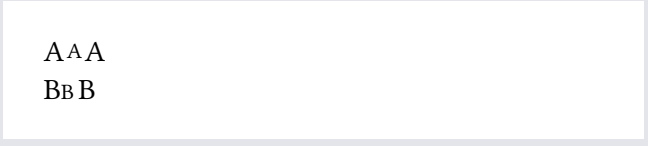
El factor de escala vertical.

El cuerpo se reflejará verticalmente si el parámetro es negativo.

#### /2.4 **origin** `alignment` *Configurable con set* Por defecto: `center + horizon`

El origen de la transformación.

```
A#box(scale(75%)[A])A \
B#box(scale(75%, origin: bottom + left)[B])B
```

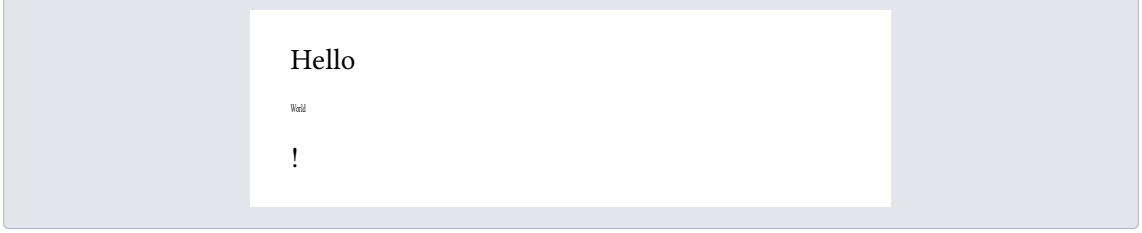



#### /2.5 **reflow** `bool` *Configurable con set* Por defecto: `false`

Si el escalado afecta el diseño.

Si se establece en `false`, el contenido escalado podrá superponerse con otro contenido. Si se establece en `true`, se calculará el nuevo tamaño del contenido escalado y se ajustará el diseño en consecuencia.

```
Hello #scale(x: 20%, y: 40%, reflow: true)[World]!
```




### /2.6 **body** `content` *Obligatorio* *Posicional*

El contenido que se escala.

## 3.10.25 skew Elemento Desde: 0.12.0

Inclina contenido.

Inclina un elemento en dirección horizontal y/o vertical. El diseño actuará como si el elemento no estuviera inclinado, salvo que especifiques `reflow: true`.

### /1 Ejemplo

```
#skew(ax: -12deg)[
  This is some fake italic text.
]
```

*This is some fake italic text.*

### /2 Parámetros

```
skew(
  ax: angle,
  ay: angle,
  origin: alignment,
  reflow: bool,
  content,
) → content
```

#### /2.1 ax `angle` *Configurable con set* Por defecto: `0deg`

El ángulo de inclinación horizontal.

```
#skew(ax: 30deg)[Skewed]
```

Skewed

#### /2.2 ay `angle` *Configurable con set* Por defecto: `0deg`

El ángulo de inclinación vertical.

```
#skew(ay: 30deg)[Skewed]
```

Skewed

#### /2.3 origin `alignment` *Configurable con set* Por defecto: `center + horizon`

El origen de la transformación de inclinación.

El origen permanecerá fijo durante la operación.

```
X #box(skew(ax: -30deg, origin: center + horizon)[X]) X \
X #box(skew(ax: -30deg, origin: bottom + left)[X]) X \
X #box(skew(ax: -30deg, origin: top + right)[X]) X
```

X X X
X X X
X X X

#### /2.4 reflow `bool` *Configurable con set* Por defecto: `false`

Si la transformación de inclinación afecta el diseño.

Si se establece en `false`, el contenido inclinado conservará el cuadro delimitador del contenido original. Si se establece en `true`, el cuadro delimitador tendrá en cuenta la transformación del contenido y ajustará el diseño en consecuencia.

```
Hello #skew(ay: 30deg, reflow: true, "World")!
```




/2.5 **body** `content` *Obligatorio* *Posicional*

El contenido que se inclina.

## 3.10.26 h Elemento Desde: siempre

Inserta espaciado horizontal en un párrafo.

El espaciado puede ser absoluto, relativo o fraccionario. En el último caso, el espacio restante de la línea se distribuye entre todos los espaciados fraccionarios según sus fracciones relativas.

### /1 Ejemplo

```
First #h(1cm) Second \
First #h(30%) Second
```

First Second
First Second

### /2 Espaciado fraccionario

Con el espaciado fraccionario, podés alinear cosas dentro de una línea sin forzar un salto de párrafo (como haría align[3.10.1]). Cada elemento con tamaño fraccionario recibe espacio según la proporción entre su fracción y la suma de todas las fracciones.

```
First #h(1fr) Second \
First #h(1fr) Second #h(1fr) Third \
First #h(2fr) Second #h(1fr) Third
```

First Second
First Second Third
First Second Third

### /3 Espaciado matemático

En las fórmulas matemáticas[3.8], además podés usar estas constantes para agregar espaciado entre elementos: `thin` (1/6 em), `med` (2/9 em), `thick` (5/18 em), `quad` (1 em), `wide` (2 em).

### /4 Parámetros

```
h(
  relative fraction,
  weak: bool,
) → content
```

#### /4.1 **amount** `relative` or `fraction` *Obligatorio* *Posicional*

Cuánto espaciado se inserta.

#### /4.2 **weak** `bool` *Configurable con set* Por defecto: `false`

Si es `true`, el espaciado colapsa al principio o al final de un párrafo. Además, de varios espaciados débiles adyacentes, colapsan todos menos el más grande.

El espaciado débil en el marcado también hace que se eliminen todos los espacios de marcado adyacentes, sin importar la cantidad de espaciado insertado. Para forzar un espacio junto a un espaciado débil, podés escribir explícitamente `#" "` (para un espacio normal) o `~` (para un espacio de no separación). Lo último puede ser útil para crear una construcción que siempre se una a la palabra anterior con un espacio de no separación, independientemente de si existía o no un espacio de marcado delante.

```
#h(1cm, weak: true)
We identified a group of _weak_
specimens that fail to manifest
in most cases. However, when
#h(8pt, weak: true) supported
#h(8pt, weak: true) on both sides,
they do show up.

Further #h(0pt, weak: true) more,
even the smallest of them swallow
adjacent markup spaces.
```

We identified a group of *weak* specimens that fail to manifest in most cases. However, when supported on both sides, they do show up.

Furthermore, even the smallest of them swallow adjacent markup spaces.

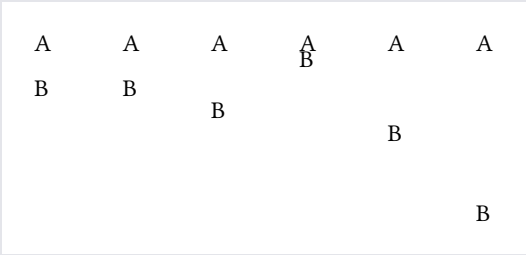
### 3.10.27 V Elemento Desde: siempre

Inserta espaciado vertical en un flujo de bloques.

El espaciado puede ser absoluto, relativo o fraccionario. En el último caso, el espacio restante de la página se distribuye entre todos los espaciados fraccionarios según sus fracciones relativas.

## /1 Ejemplo

```
#grid(
  rows: 3cm,
  columns: 6,
  gutter: 1fr,
  [A #parbreak() B],
  [A #v(0pt) B],
  [A #v(10pt) B],
  [A #v(0pt, weak: true) B],
  [A #v(40%, weak: true) B],
  [A #v(1fr) B],
)
```




## /2 Parámetros

```
v(
  relative fraction,
  weak: bool,
) → content
```

### /2.1 amount relative or fraction *Obligatorio* *Posicional*

Cuánto espaciado se inserta.

### /2.2 weak bool *Configurable con set* Por defecto: false

Si es true, el espaciado colapsa al principio o al final de un flujo. Además, de varios espaciados débiles adyacentes, colapsan todos menos el más grande. Los espaciados débiles siempre colapsan el espaciado de párrafo adyacente, incluso si el espaciado de párrafo es mayor.

```
The following theorem is
foundational to the field:
#v(4pt, weak: true)
$ x^2 + y^2 = r^2 $
#v(4pt, weak: true)
The proof is simple:
```

The following theorem is foundational to the field:

$$x^2 + y^2 = r^2$$

The proof is simple:

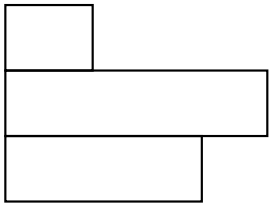
## 3.10.28 `stack` Elemento Desde: siempre

Dispone contenido y espaciado horizontal o verticalmente.

La pila (stack) ubica una lista de elementos a lo largo de un eje, con espaciado opcional entre cada elemento.

### /1 Ejemplo

```
#stack(
  dir: ttb,
  rect(width: 40pt),
  rect(width: 120pt),
  rect(width: 90pt),
)
```



### /2 Accesibilidad

Las pilas no tienen ninguna semántica especial. Las tecnologías de asistencia (TA) leen el contenido de la pila en el orden en que se lo pasó a esta función.

### /3 Parámetros

```
stack(
  dir: direction,
  spacing: none relative fraction,
  ..relative fraction content,
) → content
```

#### /3.1 `dir` `direction` *Configurable con set* Por defecto: `ttb`

La dirección a lo largo de la cual se apilan los elementos. Los valores posibles son:

- `ltr`: De izquierda a derecha.
- `rtl`: De derecha a izquierda.
- `ttb`: De arriba hacia abajo.
- `btt`: De abajo hacia arriba.

Podés usar los métodos `start` y `end` para obtener los puntos inicial y final, respectivamente, de una dirección, como `alignment`. También podés usar el método `axis` para determinar si una dirección es `"horizontal"` o `"vertical"`. El método `inv` devuelve la dirección inversa de una dirección.

Por ejemplo, `ttb.start()` es `top`, `ttb.end()` es `bottom`, `ttb.axis()` es `"vertical"` y `ttb.inv()` es igual a `btt`.

#### /3.2 `spacing` `none` or `relative` or `fraction` *Configurable con set* Por defecto: `none`

Espaciado que se inserta entre los elementos donde no se proporcionó un espaciado explícito.

#### /3.3 `children` `relative` or `fraction` or `content` *Obligatorio* *Posicional* *Variádico*

Los hijos que se apilan a lo largo del eje.

# 3.11 Visualización

Dibujo y visualización de datos.

Si querés crear dibujos o gráficos más avanzados, mirá también el paquete CeTZ[A], así como otros paquetes[B] más especializados para tu caso de uso.

## 3.11.1 Accesibilidad

Todas las formas y rutas dibujadas por Typst se marcan automáticamente como artefactos[3.14.4/4] para que sean invisibles para las tecnologías de asistencia (TA) durante la exportación a PDF. Sin embargo, su contenido (si lo hay) sigue siendo accesible.

Si estás usando las funciones de esta categoría para crear una ilustración con significado semántico, hacela accesible envolviéndola en una llamada a la función `figure`[3.6.7]. Usá su parámetro `alt`[3.6.7/5.2] para proporcionar una descripción alternativa[4.4.7].

## Definiciones


3.11.2 Circle A circle with optional content. . . . . 426
3.11.3 Color A color in a specific color space. . . . . 428
3.11.4 Curve A curve consisting of movements, lines, and Bézier segments. . . . . 442
3.11.5 Ellipse An ellipse with optional content. . . . . 449
3.11.6 Gradient A color gradient. . . . . 450
3.11.7 Image A raster or vector graphic. . . . . 459
3.11.8 Line A line from one point to another. . . . . 464
3.11.9 Polygon A closed polygon. . . . . 466
3.11.10 Rectangle A rectangle with optional content. . . . . 468
3.11.11 Square A square with optional content. . . . . 471
3.11.12 Stroke Defines how to draw a line. . . . . 473
3.11.13 Tiling A repeating tiling fill. . . . . 478


[A] https://github.com/johannes-wolf/cetz
[B] https://typst.app/universe

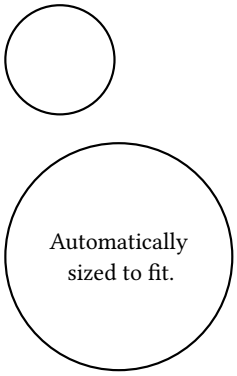
### 3.11.2 `circle` Elemento Desde: siempre

Un círculo con contenido opcional.

#### /1 Ejemplo

```
// Without content.
#circle(radius: 25pt)

// With content.
#circle[
  #set align(center + horizon)
  Automatically \
  sized to fit.
]
```




#### /2 Parámetros

```
circle(
  radius: length,
  width: auto relative,
  height: auto relative fraction,
  fill: none color gradient tiling,
  stroke: none auto length color gradient stroke tiling dictionary,
  inset: relative dictionary,
  outset: relative dictionary,
  none content,
) → content
```

##### /2.1 `radius` `length` Por defecto: `0pt`

El radio del círculo. Es mutuamente excluyente con `width` y `height`.

##### /2.2 `width` `auto` or `relative` *Configurable con set* Por defecto: `auto`

El ancho del círculo. Es mutuamente excluyente con `radius` y `height`.

A diferencia de `radius`, esto puede ser relativo al ancho del contenedor padre.

##### /2.3 `height` `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

La altura del círculo. Es mutuamente excluyente con `radius` y `width`.

A diferencia de `radius`, esto puede ser relativo a la altura del contenedor padre.

##### /2.4 `fill` `none` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `none`

Cómo rellenar el círculo. Mirá la documentación del rectángulo[3.11.10/2.3] para más detalles.

##### /2.5 `stroke` `none` or `auto` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `auto`

Cómo trazar el círculo. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

##### /2.6 `inset` `relative` or `dictionary` *Configurable con set* Por defecto: `0% + 5pt`

Cuánto relleno se agrega al contenido del círculo. Mirá la documentación de box[3.10.5/2.7] para más detalles.

/2.7 **outset** `relative` or `dictionary` *Configurable con set* Por defecto: (:)

Cuánto se expande el tamaño del círculo sin afectar el diseño. Mirá la documentación de box[3.10.5/2.8] para más detalles.

/2.8 **body** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido que se coloca dentro del círculo. El círculo se expande para ajustarse a este contenido, manteniendo la relación de aspecto 1-1.

### 3.11.3 color Desde: siempre

Un color en un espacio de color específico.

Typst admite:
- sRGB mediante la función `rgb`[3.11.3/4.5]
- CMYK de dispositivo mediante la función `cmyk`[3.11.3/4.6]
- Gris D65 mediante la función `luma`[3.11.3/4.1]
- Oklab mediante la función `oklab`[3.11.3/4.2]
- Oklch mediante la función `oklch`[3.11.3/4.3]
- RGB lineal mediante la función `color.linear-rgb`[3.11.3/4.4]
- HSL mediante la función `color.hsl`[3.11.3/4.7]
- HSV mediante la función `color.hsv`[3.11.3/4.8]
- Espacios de color descritos por colorantes planos (spot) mediante el tipo `color.spot`[3.11.3/5.1]

Todos los espacios de color, excepto CMYK y los colorantes planos, tienen canales alfa.

En toda la documentación usamos el término *color de proceso* para los colores que se pueden mezclar entre sí (actualmente todos los colores que no son colores planos). En este término, *proceso* significa que el tono se crea durante el proceso de impresión y no de antemano.

#### /1 Ejemplo

```
#rect(fill: aqua)
```

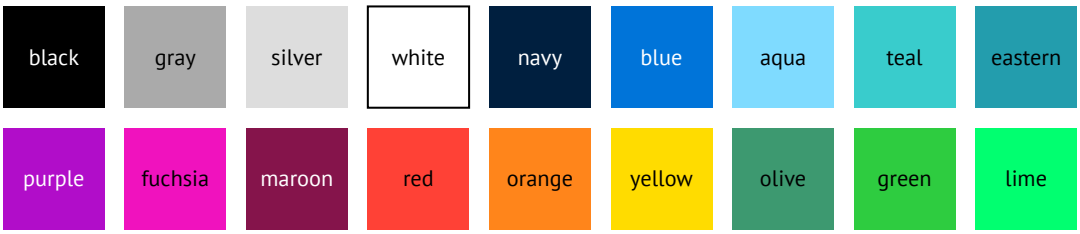


#### /2 Colores predefinidos

Typst define los siguientes colores incorporados:

| Color | Definición |
|---|---|
| black | `luma(0)` |
| gray | `luma(170)` |
| silver | `luma(221)` |
| white | `luma(255)` |
| navy | `rgb("#001f3f")` |
| blue | `rgb("#0074d9")` |
| aqua | `rgb("#7fdbff")` |
| teal | `rgb("#39cccc")` |
| eastern | `rgb("#239dad")` |
| purple | `rgb("#b10dc9")` |
| fuchsia | `rgb("#f012be")` |
| maroon | `rgb("#85144b")` |
| red | `rgb("#ff4136")` |
| orange | `rgb("#ff851b")` |
| yellow | `rgb("#ffdc00")` |
| olive | `rgb("#3d9970")` |
| green | `rgb("#2ecc40")` |
| lime | `rgb("#01ff70")` |

Los colores predefinidos y los constructores de color más importantes están disponibles globalmente y también en el ámbito del tipo color, así que podés escribir `color.red` o simplemente `red`.

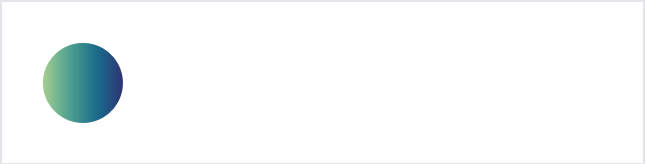



## /3 Constantes

### /3.1 map `module` Desde: 0.9.0

Una colección de mapas de color predefinidos que se pueden usar para los degradados[3.11.6/2]. Son simplemente arrays de colores.

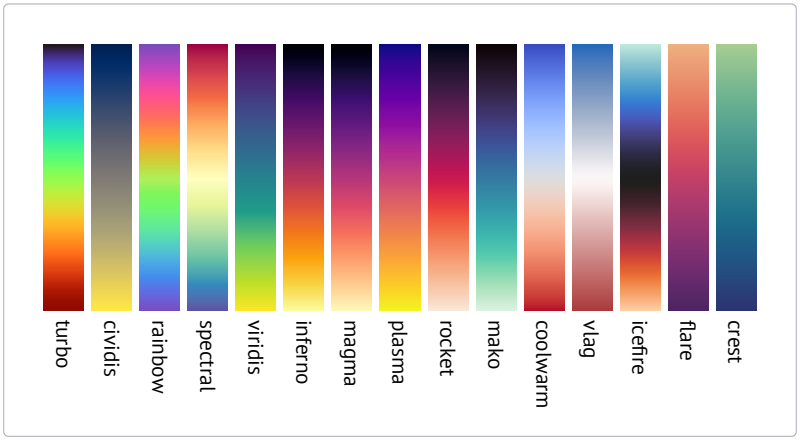
```
#circle(fill: gradient.linear(..color.map.crest))
```



| Mapa | Detalles |
|---|---|
| `turbo` | Un mapa de color tipo arcoíris perceptualmente uniforme. Leé esta entrada de blog[A] para más detalles. |
| `cividis` | Un mapa de color de azul a gris a amarillo. Mirá esta entrada de blog[B] para más detalles. |
| `rainbow` | Recorre todo el espectro de colores. Este mapa de color se usa mejor estableciendo el espacio de color de interpolación en HSL[3.11.3/4.7]. El degradado arcoíris **no es adecuado** para la visualización de datos porque no es perceptualmente uniforme, por lo que las diferencias entre valores se vuelven poco claras para tus lectores. Solo debe usarse con fines decorativos. |
| `spectral` | Mapa de color de rojo a amarillo a azul. |
| `viridis` | Un mapa de color de violeta a verde azulado a amarillo. |
| `inferno` | Un mapa de color de negro a rojo a amarillo. |
| `magma` | Un mapa de color de negro a violeta a amarillo. |
| `plasma` | Un mapa de color de violeta a rosa a amarillo. |
| `rocket` | Un mapa de color de negro a rojo a blanco. |
| `mako` | Un mapa de color de negro a verde azulado a blanco. |
| `coolwarm` | Un mapa de color de azul a blanco a rojo con transiciones suaves. |
| `vlag` | Un mapa de color de azul claro a blanco a rojo. |
| `icefire` | Un mapa de color de verde azulado claro a negro a naranja. |
| `flare` | Un mapa de color de naranja a violeta que es perceptualmente uniforme. |
| `crest` | Un mapa de color de verde claro a azul. |

Algunos preajustes populares no se incluyen porque no están disponibles bajo una licencia libre. Otros, como Jet[C], no se incluyen porque no son amigables para personas con daltonismo. ¡Sentite libre de usar o crear un paquete con otros preajustes que te resulten útiles!

[A] https://ai.googleblog.com/2019/08/turbo-improved-rainbow-colormap-for.html
[B] https://bids.github.io/colormap/
[C] https://jakevdp.github.io/blog/2014/10/16/how-bad-is-your-colormap/




## /4 Funciones

### /4.1 luma Desde: siempre

Crea un color en escala de grises.

Un color en escala de grises se representa con los componentes `lightness` (ratio[3.10.20]) y `alpha` (ratio[3.10.20]).

Estos componentes también están disponibles con el método `components`[3.11.3/4.9].

```
#for x in range(250, step: 50) {
  box(square(fill: luma(x)))
}
```



```
color.luma(
  int ratio,
  int ratio,
  color,
) → color
```

#### /4.1.1 lightness int or ratio *Obligatorio* *Posicional*

El componente de luminosidad.

#### /4.1.2 alpha int or ratio *Obligatorio* *Posicional*

El componente alfa.

#### /4.1.3 color color *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a escala de grises.

Si se da, no se deben dar los componentes individuales.

### /4.2 oklab Desde: 0.9.0

Crea un color Oklab[A].

Este espacio de color es muy adecuado para los siguientes casos de uso:
- Manipulación de colores, como saturar manteniendo el tono percibido
- Crear imágenes en escala de grises con luminosidad percibida uniforme
- Crear transiciones de color y degradados suaves y uniformes

Un color Oklab lineal se representa internamente con un array de cuatro componentes:
- luminosidad (ratio[3.10.20])
- a (float[3.5.15] o ratio[3.10.20]. Las proporciones son relativas a `0.4`; es decir, `50%` es igual a `0.2`)

[A] https://bottosson.github.io/posts/oklab/

– b (float[3.5.15] o ratio[3.10.20]. Las proporciones son relativas a 0.4; es decir, 50% es igual a 0.2)
– alfa (ratio[3.10.20])

Estos componentes también están disponibles con el método components[3.11.3/4.9].

```
#square(
  fill: oklab(27%, 20%, -3%, 50%)
)
```



```
color.oklab(
  ratio,
  float ratio,
  float ratio,
  ratio,
  color,
) → color
```

/4.2.1 **lightness** ratio *Obligatorio* *Posicional*

El componente de luminosidad.

/4.2.2 **a** float or ratio *Obligatorio* *Posicional*

El componente a ("verde/rojo").

/4.2.3 **b** float or ratio *Obligatorio* *Posicional*

El componente b ("azul/amarillo").

/4.2.4 **alpha** ratio *Obligatorio* *Posicional*

El componente alfa.

/4.2.5 **color** color *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a Oklab.

Si se da, no se deben dar los componentes individuales.

## /4.3 oklch Desde: 0.10.0

Crea un color Oklch[A].

Este espacio de color es muy adecuado para los siguientes casos de uso:
– Manipulación de colores que involucra luminosidad, croma y tono
– Crear imágenes en escala de grises con luminosidad percibida uniforme
– Crear transiciones de color y degradados suaves y uniformes

Un color Oklch lineal se representa internamente con un array de cuatro componentes:
– luminosidad (ratio[3.10.20])
– croma (float[3.5.15] o ratio[3.10.20]. Las proporciones son relativas a 0.4; es decir, 50% es igual a 0.2)
– tono (angle[3.10.3])
– alfa (ratio[3.10.20])

Estos componentes también están disponibles con el método components[3.11.3/4.9].

```
#square(
  fill: oklch(40%, 0.2, 160deg, 50%)
)
```



[A] https://bottosson.github.io/posts/oklab/

```
color.oklch(
  ratio,
  float ratio,
  angle,
  ratio,
  color,
) → color
```

/4.3.1 **lightness** ratio *Obligatorio* *Posicional*

El componente de luminosidad.

/4.3.2 **chroma** float or ratio *Obligatorio* *Posicional*

El componente de croma.

/4.3.3 **hue** angle *Obligatorio* *Posicional*

El componente de tono.

/4.3.4 **alpha** ratio *Obligatorio* *Posicional*

El componente alfa.

/4.3.5 **color** color *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a Oklch.

Si se da, no se deben dar los componentes individuales.

## /4.4 linear-rgb Desde: 0.9.0

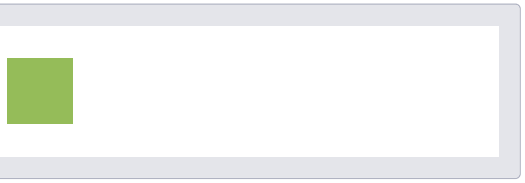
Crea un color RGB(A) con luma lineal.

Este espacio de color es similar a sRGB, pero con la diferencia de que los componentes de color no tienen corrección gamma. Esto facilita realizar operaciones de color como la mezcla y la interpolación. Aunque, para eso deberías preferir la función `oklab`[3.11.3/4.2].

Un color RGB(A) lineal se representa internamente con un array de cuatro componentes:
- rojo (`ratio`[3.10.20])
- verde (`ratio`[3.10.20])
- azul (`ratio`[3.10.20])
- alfa (`ratio`[3.10.20])

Estos componentes también están disponibles con el método `components`[3.11.3/4.9].

```
#square(fill: color.linear-rgb(
  30%, 50%, 10%,
))
```



```
color.linear-rgb(
  int ratio,
  int ratio,
  int ratio,
  int ratio,
  color,
) → color
```

/4.4.1 **red** int or ratio *Obligatorio* *Posicional*

El componente rojo.

/4.4.2 **green** int or ratio *Obligatorio* *Posicional*

El componente verde.

/4.4.3 **blue** int or ratio *Obligatorio* *Posicional*

El componente azul.

/4.4.4 **alpha** int or ratio *Obligatorio* *Posicional*

El componente alfa.

/4.4.5 **color** color *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a RGB(A) lineal.

Si se da, no se deben dar los componentes individuales.

/4.5 **rgb** Desde: siempre

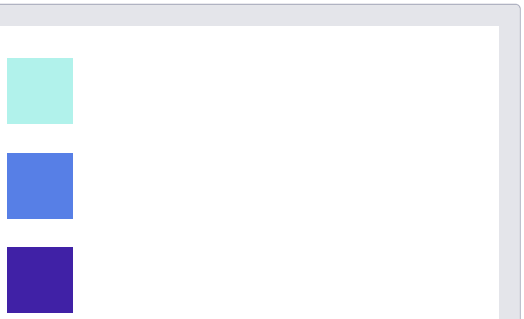
Crea un color RGB(A).

El color se especifica en el espacio de color sRGB.

Un color RGB(A) se representa internamente con un array de cuatro componentes:
- rojo (ratio[3.10.20])
- verde (ratio[3.10.20])
- azul (ratio[3.10.20])
- alfa (ratio[3.10.20])

Estos componentes también están disponibles con el método components[3.11.3/4.9].

```
#square(fill: rgb("#b1f2eb"))
#square(fill: rgb(87, 127, 230))
#square(fill: rgb(25%, 13%, 65%))
```



```
color.rgb(
  int ratio,
  int ratio,
  int ratio,
  int ratio,
  str,
  color,
) → color
```

/4.5.1 **red** int or ratio *Obligatorio* *Posicional*

El componente rojo.

/4.5.2 **green** int or ratio *Obligatorio* *Posicional*

El componente verde.

/4.5.3 **blue** int or ratio *Obligatorio* *Posicional*

El componente azul.

/4.5.4 **alpha** int or ratio *Obligatorio* *Posicional*

El componente alfa.

/4.5.5 **hex** str *Obligatorio* *Posicional*

Alternativamente: el color en notación hexadecimal.

Acepta tres, cuatro, seis u ocho dígitos hexadecimales y, opcionalmente, un numeral inicial.

Si se da, no se deben dar los componentes individuales.

```
#text(16pt, rgb("#239dad"))[
  *Typst*
]
```

Typst

/4.5.6 **color** color *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a RGB(a).

Si se da, no se deben dar los componentes individuales.

/4.6 **cmyk** Desde: siempre

Crea un color CMYK.

Es útil si querés apuntar a una impresora específica. La conversión a RGB para la vista previa en pantalla puede diferir de cómo tu impresora reproduce el color.

Un color CMYK se representa internamente con un array de cuatro componentes:
- cian (ratio[3.10.20])
- magenta (ratio[3.10.20])
- amarillo (ratio[3.10.20])
- clave / negro (key) (ratio[3.10.20])

Estos componentes también están disponibles con el método components[3.11.3/4.9].

Tené en cuenta que los colores CMYK actualmente no se admiten cuando la salida PDF/A está habilitada.

```
#square(
  fill: cmyk(27%, 0%, 3%, 5%)
)
```



```
color.cmyk(
  ratio,
  ratio,
  ratio,
  ratio,
  color,
) → color
```

/4.6.1 **cyan** ratio *Obligatorio* *Posicional*

El componente cian.

/4.6.2 **magenta** ratio *Obligatorio* *Posicional*

El componente magenta.

/4.6.3 **yellow** ratio *Obligatorio* *Posicional*

El componente amarillo.

/4.6.4 **key** ratio *Obligatorio* *Posicional*

El componente clave (negro).

/4.6.5 **color** color *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a CMYK.

Si se da, no se deben dar los componentes individuales.

### /4.7 hsl Desde: 0.9.0

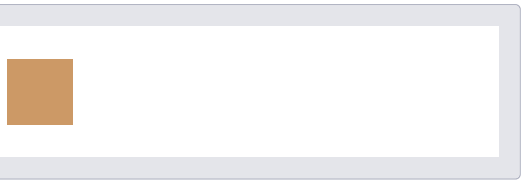
Crea un color HSL.

Este espacio de color es útil para especificar colores por tono, saturación y luminosidad. También es útil para la manipulación de colores, como saturar manteniendo el tono percibido.

Un color HSL se representa internamente con un array de cuatro componentes:
- tono (angle[3.10.3])
- saturación (ratio[3.10.20])
- luminosidad (ratio[3.10.20])
- alfa (ratio[3.10.20])

Estos componentes también están disponibles con el método components[3.11.3/4.9].

```
#square(
  fill: color.hsl(30deg, 50%, 60%)
)
```



```
color.hsl(
  angle,
  int ratio,
  int ratio,
  int ratio,
  color,
) → color
```

#### /4.7.1 hue angle *Obligatorio* *Posicional*

El ángulo de tono.

#### /4.7.2 saturation int or ratio *Obligatorio* *Posicional*

El componente de saturación.

#### /4.7.3 lightness int or ratio *Obligatorio* *Posicional*

El componente de luminosidad.

#### /4.7.4 alpha int or ratio *Obligatorio* *Posicional*

El componente alfa.

#### /4.7.5 color color *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a HSL.

Si se da, no se deben dar los componentes individuales.

### /4.8 hsv Desde: 0.9.0

Crea un color HSV.

Este espacio de color es útil para especificar colores por tono, saturación y valor. También es útil para la manipulación de colores, como saturar manteniendo el tono percibido.

Un color HSV se representa internamente con un array de cuatro componentes:
- tono (angle[3.10.3])
- saturación (ratio[3.10.20])
- valor (ratio[3.10.20])
- alfa (ratio[3.10.20])

Estos componentes también están disponibles con el método components[3.11.3/4.9].

```
#square(
  fill: color.hsv(30deg, 50%, 60%)
)
```



```
color.hsv(
  angle,
  int ratio,
  int ratio,
  int ratio,
  color,
) → color
```

/4.8.1 **hue** `angle` *Obligatorio* *Posicional*

El ángulo de tono.

/4.8.2 **saturation** `int` or `ratio` *Obligatorio* *Posicional*

El componente de saturación.

/4.8.3 **value** `int` or `ratio` *Obligatorio* *Posicional*

El componente de valor.

/4.8.4 **alpha** `int` or `ratio` *Obligatorio* *Posicional*

El componente alfa.

/4.8.5 **color** `color` *Obligatorio* *Posicional*

Alternativamente: el color que se convierte a HSL.

Si se da, no se deben dar los componentes individuales.

## /4.9 components Desde: 0.9.0

Extrae los componentes de este color.

El tamaño y los valores de este array dependen del espacio de color. Podés obtener el espacio de color con space[3.11.3/4.10]. A continuación hay una tabla de los espacios de color y sus componentes:

| Espacio de color | C1 | C2 | C3 | C4 |
|---|---|---|---|---|
| luma[3.11.3/4.1] | Luminosidad | | | |
| oklab[3.11.3/4.2] | Luminosidad | a | b | Alfa |
| oklch[3.11.3/4.3] | Luminosidad | Croma | Tono | Alfa |
| linear-rgb[3.11.3/4.4] | Rojo | Verde | Azul | Alfa |
| rgb[3.11.3/4.5] | Rojo | Verde | Azul | Alfa |
| cmyk[3.11.3/4.6] | Cian | Magenta | Amarillo | Clave |
| hsl[3.11.3/4.7] | Tono | Saturación | Luminosidad | Alfa |
| hsv[3.11.3/4.8] | Tono | Saturación | Valor | Alfa |
| spot[3.11.3/5.1] | Tinte | | | |

Para el significado y el tipo de cada valor individual, mirá la documentación del espacio de color correspondiente. El componente alfa es opcional y solo se incluye si el argumento `alpha` es `true`. La longitud del array devuelto depende de la cantidad de componentes y de si se incluye el componente alfa.

```
// note that the alpha component is included by default
#rgb(40%, 60%, 80%).components()
```

```
(40%, 60%, 80%, 100%)
```

```
self.components(alpha: bool) → array
```

/4.9.1 **alpha** `bool` Por defecto: `true`

Si se incluye el componente alfa.

## /4.10 space Desde: 0.9.0

Devuelve la función constructora del espacio de este color.

Devuelve una de las siguientes:
- luma[3.11.3/4.1]
- oklab[3.11.3/4.2]
- oklch[3.11.3/4.3]
- linear-rgb[3.11.3/4.4]
- rgb[3.11.3/4.5]
- cmyk[3.11.3/4.6]
- hsl[3.11.3/4.7]
- hsv[3.11.3/4.8]

```
#let color = cmyk(1%, 2%, 3%, 4%)
#(color.space() == cmyk)
```

```
true
```

```
self.space() → spot any
```

## /4.11 to-hex Desde: 0.9.0

Devuelve la representación hexadecimal RGB(A) del color (como `#ffaa32` o `#020304fe`). El componente alfa (los dos últimos dígitos en `#020304fe`) se omite si es igual a `ff` (255 / 100%).

```
self.to-hex() → str
```

## /4.12 lighten Desde: siempre

Aclara un color según un factor dado.

```
self.lighten(ratio) → color
```

/4.12.1 **factor** `ratio` *Obligatorio* *Posicional*

El factor por el que se aclara el color.

## /4.13 darken Desde: siempre

Oscurece un color según un factor dado.

```
self.darken(ratio) → color
```

/4.13.1 **factor** `ratio` *Obligatorio* *Posicional*

El factor por el que se oscurece el color.

### /4.14 saturate Desde: 0.9.0

Aumenta la saturación de un color según un factor dado.

Solo se pueden saturar los colores de proceso. Si querés saturar un color plano, convertilo primero en un color de proceso.

```
self.saturate( ratio ) → color
```

#### /4.14.1 factor ratio *Obligatorio* *Posicional*

El factor por el que se satura el color.

### /4.15 desaturate Desde: 0.9.0

Disminuye la saturación de un color según un factor dado.

Solo se pueden desaturar los colores de proceso. Si querés desaturar un color plano, convertilo primero en un color de proceso.

```
self.desaturate( ratio ) → color
```

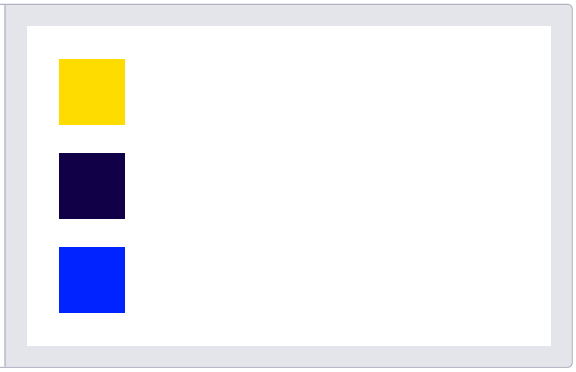
#### /4.15.1 factor ratio *Obligatorio* *Posicional*

El factor por el que se desatura el color.

### /4.16 negate Desde: siempre

Produce el color complementario usando un espacio de color dado. Podés pensarlo como el lado opuesto en una rueda de colores.

```
#square(fill: yellow)
#square(fill: yellow.negate())
#square(fill: yellow.negate(space: rgb))
```



```
self.negate(space: auto spot any ) → color
```

#### /4.16.1 space auto or spot or any Por defecto: auto

El espacio de color que se usa para la transformación. Por defecto se usa un espacio de color perceptual.

### /4.17 rotate Desde: 0.9.0

Rota el tono del color según un ángulo dado.

Esta función solo funciona en modelos de color con un componente de tono bien definido, es decir, Oklch, HSL y HSV.

```
self.rotate(
  angle ,
  space: any ,
) → color
```

#### /4.17.1 angle angle *Obligatorio* *Posicional*

El ángulo por el que se rota el tono.

/4.17.2 **space** any Por defecto: oklch

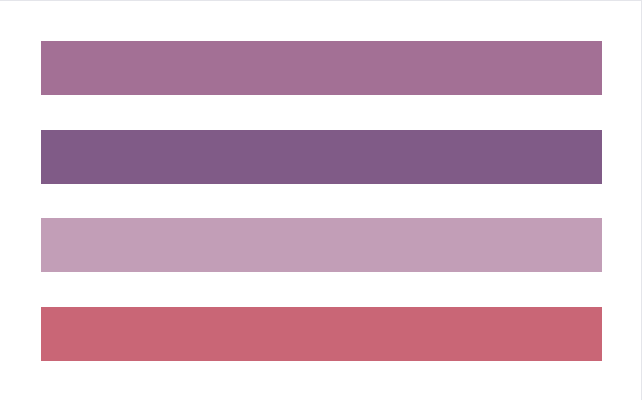
El espacio de color que se usa para rotar. Por defecto, esto ocurre en un espacio de color perceptual (oklch[3.11.3/4.3]).

## /4.18 mix Desde: 0.7.0

Crea un color mezclando dos o más colores.

En los espacios de color con un componente de tono (HSL, HSV, Oklch), solo se pueden mezclar dos colores a la vez. ¡Mezclar más de dos colores en un espacio así producirá un error!

```
#set block(height: 20pt, width: 100%)
#block(fill: red.mix(blue))
#block(fill: red.mix(blue, space: rgb))
#block(fill: color.mix(red, blue, white))
#block(fill: color.mix((red, 70%), (blue, 30%)))
```



```
color.mix(
  ..color array ,
  space: auto spot any ,
) → color
```

/4.18.1 **colors** color or array *Obligatorio* *Posicional* *Variádico*

Los colores, opcionalmente con pesos, especificados como un par (array de longitud dos) de color y peso (flotante o proporción).

Los pesos no necesitan sumar 100%, son relativos a la suma de todos los pesos.

/4.18.2 **space** auto or spot or any Por defecto: auto

El espacio de color en el que se mezcla. Por defecto, esto ocurre en un espacio de color perceptual (oklab[3.11.3/4.2]) o, si todos los colores usan el mismo colorante plano, usando ese colorante.

Todos los colores se convertirán a este espacio de color.

## /4.19 transparentize Desde: 0.11.0

Vuelve un color más transparente según un factor dado.

Este método es relativo al valor alfa existente. Si la escala es positiva, calcula `alpha - alpha * scale`. Las escalas negativas se comportan como `color.opacify(-scale)`.

```
#block(fill: red)[opaque]
#block(fill: red.transparentize(50%))[half red]
#block(fill: red.transparentize(75%))[quarter red]
```




```
self.transparentize( ratio ) → color
```

/4.19.1 **scale** `ratio` *Obligatorio* *Posicional*

El factor por el que se cambia el valor alfa.

/4.20 **opacify** Desde: 0.11.0

Vuelve un color más opaco según una escala dada.

Este método es relativo al valor alfa existente. Si la escala es positiva, calcula `alpha + scale - alpha * scale`. Las escalas negativas se comportan como `color.transparentize(-scale)`.

```
#let half-red = red.transparentize(50%)
#block(fill: half-red.opacify(100%))[opaque]
#block(fill: half-red.opacify(50%))[three quarters red]
#block(fill: half-red.opacify(-50%))[one quarter red]
```




```
self.opacify( ratio ) → color
```

/4.20.1 **scale** `ratio` *Obligatorio* *Posicional*

La escala por la que se cambia el valor alfa.

## /5 Tipos

/5.1 spot Desde: 0.15.0

Un colorante plano (spot) a partir del cual se pueden crear colores planos.

Usá colores planos para solicitar un pigmento preciso en un entorno de impresión profesional. Una vez que creaste un colorante plano, podés crear colores con su método `tint`[3.11.3/5.1.2.1].

/5.1.1 **Constructor**

Crea un colorante plano nuevo.

```
color.spot(
  none  str ,
  color ,
) → spot
```

/5.1.1.1 **name** `none` or `str` *Obligatorio* *Posicional*

Nombre del colorante plano que se usa.

En producción, este nombre se verificará manualmente y se asociará a un colorante, por lo que este valor debe ser inequívoco. Lo mejor es hacer referencia a un color de un registro como PANTONE, HKS, RAL, Toyo & DIC, etcétera.

Los valores acá pueden tratarse distinguiendo mayúsculas de minúsculas durante la producción: `"PANTONE 2221 C"` y `"PANTONE 2221 c"` pueden tratarse como colores distintos. Asegurate de usar una convención de nombres consistente, ya sea haciendo referencia a un registro o coordinando con tus expertos de impresión de producción.

Si este valor es `"all"` y tu impresión involucrará varias planchas de color[A], el uso de este colorante dará como resultado que el tinte especificado se aplique por igual a todas las planchas. Si elegís none, no se aplicará ningún colorante al usar este color. Este valor especial suele usarse para indicar cortes o barnices. ¡Asegurate de consultarlo con tu imprenta de producción!

No recomendamos usar los nombres `"Cyan"`, `"Magenta"`, `"Yellow"`, `"Key"`, `"Black"` ni sus traducciones a tu idioma local. Según tu impresora, pueden o no interpretarse como colores de proceso CMYK.

| Variante | Detalles |
|---|---|
| `"all"` | Usa todas las placas de color disponibles en lugar de un solo colorante. |

#### /5.1.1.2 **fallback** color *Obligatorio* *Posicional*

Cómo renderizar este color si el colorante especificado no está disponible.

Muchos medios, como la vista previa en pantalla y las impresoras domésticas, no tendrán disponible este colorante plano específico. Para mostrar una aproximación de la impresión prevista, se usa en su lugar otro color disponible.

### /5.1.2 Funciones on spot

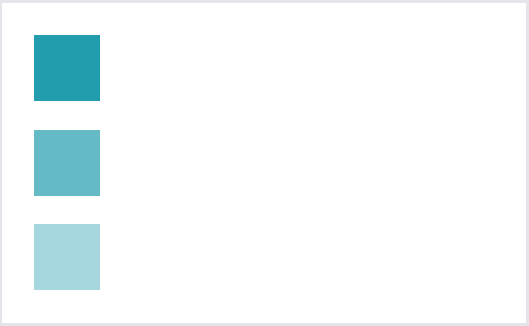
#### /5.1.2.1 **tint** Desde: 0.15.0

Crea un color plano en un tinte específico de este colorante.

El tinte representa qué porcentaje del colorante se aplica. Un tinte de `100%` significa que el colorante se aplica a plena intensidad, mientras que `0%` significa que no se aplica colorante.

```
#let pantone = color.spot(
  "PANTONE 2221 C",
  rgb("#239dad")
)

#square(fill: pantone.tint(100%))
#square(fill: pantone.tint(70%))
#square(fill: pantone.tint(40%))
```



```
self.tint( ratio ) → color
```

#### /5.1.2.1.1 **value** ratio *Obligatorio* *Posicional*

El porcentaje de tinte, entre `0%` y `100%`.

[A] https://en.wikipedia.org/wiki/Offset_printing#Plates

### 3.11.4 `curve` Elemento Desde: 0.13.0

Una curva formada por movimientos, líneas y segmentos de Bézier.

En todo momento hay una pluma o cursor conceptual.
- Los elementos de movimiento (move) mueven el cursor sin dibujar.
- Los elementos línea/cuadrática/cúbica (Line/Quadratic/Cubic) dibujan un segmento desde el cursor hasta una nueva posición, potencialmente con un punto de control para una curva de Bézier.
- Los elementos de cierre (close) dibujan una línea recta o suave de regreso al inicio de la curva o al último segmento de movimiento anterior.

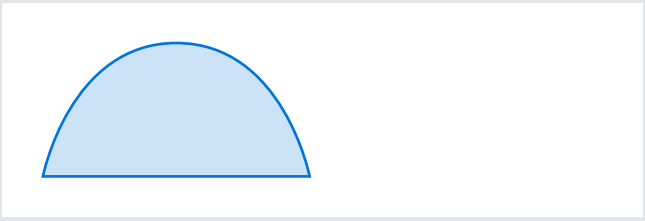
A los fines del diseño, el cuadro delimitador de la curva es un rectángulo ajustado que contiene todos los segmentos, así como el punto `(0pt, 0pt)`.

Las posiciones se pueden especificar de forma absoluta (es decir, relativas a `(0pt, 0pt)`) o relativas a la posición actual de la pluma/cursor, es decir, la posición donde terminó el segmento anterior.

Los puntos de control de las curvas de Bézier se pueden omitir pasando `none` o reflejarse automáticamente del segmento anterior pasando `auto`.

#### /1 Ejemplo

```
#curve(
  fill: blue.lighten(80%),
  stroke: blue,
  curve.move((0pt, 50pt)),
  curve.line((100pt, 50pt)),
  curve.cubic(none, (90pt, 0pt), (50pt, 0pt)),
  curve.close(),
)
```



#### /2 Parámetros

```
curve(
  fill: none color gradient tiling,
  fill-rule: str,
  stroke: none auto length color gradient stroke tiling dictionary,
  ..content,
) → content
```

##### /2.1 `fill` `none` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `none`

Cómo rellenar la curva.

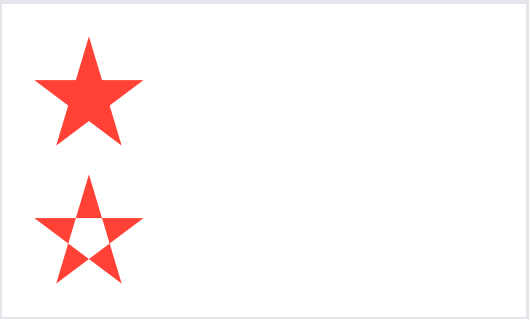
Al establecer un relleno, el trazo por defecto desaparece. Para crear una curva con relleno y trazo, tenés que configurar ambos.

##### /2.2 `fill-rule` `str` *Configurable con set* Por defecto: `"non-zero"`

La regla de dibujo que se usa para rellenar la curva.

```
// We use `.with` to get a new
// function that has the common
// arguments pre-applied.
#let star = curve.with(
  fill: red,
  curve.move((25pt, 0pt)),
  curve.line((10pt, 50pt)),
  curve.line((50pt, 20pt)),
  curve.line((0pt, 20pt)),
  curve.line((40pt, 50pt)),
  curve.close(),
)

#star(fill-rule: "non-zero")
#star(fill-rule: "even-odd")
```



| Variante | Detalles |
|---|---|
| `"non-zero"` | Especifica que el "interior" se calcula con una suma distinta de cero de cruces de bordes con signo. |
| `"even-odd"` | Especifica que el "interior" se calcula con una cantidad impar de cruces de bordes. |

/2.3 **stroke** `none` or `auto` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `auto`

Cómo trazar[3.11.12] la curva.

Se puede establecer en `none` para desactivar el trazo o en `auto` para un trazo negro de `1pt` si y solo si no se da un relleno.

```
#let down = curve.line((40pt, 40pt), relative: true)
#let up = curve.line((40pt, -40pt), relative: true)

#curve(
  stroke: 4pt + gradient.linear(red, blue),
  down, up, down, up, down,
)
```



/2.4 **components** `content` *Obligatorio* *Posicional* *Variádico*

Los componentes de la curva, en forma de movimientos, segmentos de línea y de Bézier, y cierres.

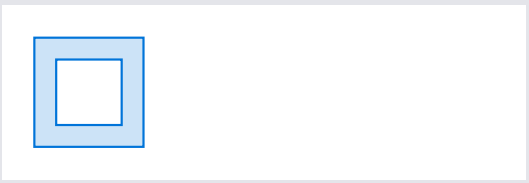
## /3 Funciones

/3.1 **move** Elemento Desde: 0.13.0

Inicia un componente de curva nuevo.

Si no se pasa ningún elemento `curve.move`, la curva empezará en `(0pt, 0pt)`.

```
#curve(
  fill: blue.lighten(80%),
  fill-rule: "even-odd",
  stroke: blue,
  curve.line((50pt, 0pt)),
  curve.line((50pt, 50pt)),
  curve.line((0pt, 50pt)),
  curve.close(),
  curve.move((10pt, 10pt)),
  curve.line((40pt, 10pt)),
  curve.line((40pt, 40pt)),
  curve.line((10pt, 40pt)),
  curve.close(),
)
```



```
curve.move(
  array,
  relative: bool,
) → content
```

/3.1.1 **start** `array` *Obligatorio* *Posicional*

El punto de inicio del componente nuevo.

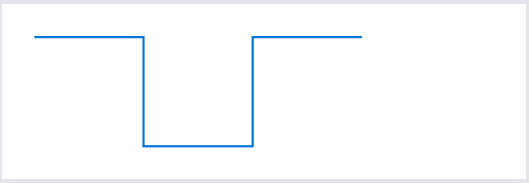
/3.1.2 **relative** `bool` *Configurable con set* Por defecto: `false`

Si las coordenadas son relativas al punto anterior.

## /3.2 line Elemento Desde: 0.13.0

Agrega una línea recta desde el punto actual hasta el siguiente.

```
#curve(
  stroke: blue,
  curve.line((50pt, 0pt)),
  curve.line((50pt, 50pt)),
  curve.line((100pt, 50pt)),
  curve.line((100pt, 0pt)),
  curve.line((150pt, 0pt)),
)
```



```
curve.line(
  array,
  relative: bool,
) → content
```

/3.2.1 **end** `array` *Obligatorio* *Posicional*

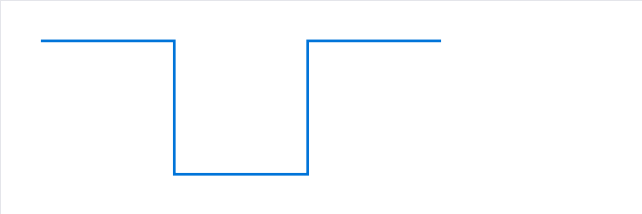
El punto en el que debe terminar la línea.

/3.2.2 **relative** `bool` *Configurable con set* Por defecto: `false`

Si las coordenadas son relativas al punto anterior.

```
#curve(
  stroke: blue,
  curve.line((50pt, 0pt), relative: true),
  curve.line((0pt, 50pt), relative: true),
  curve.line((50pt, 0pt), relative: true),
  curve.line((0pt, -50pt), relative: true),
```

```
  curve.line((50pt, 0pt), relative: true),
)
```



### /3.3 quad Elemento Desde: 0.13.0

Agrega un segmento de curva de Bézier cuadrática desde el último punto hasta `end`, usando `control` como punto de control.

```
// Function to illustrate where the control point is.
#let mark((x, y)) = place(
  dx: x - 1pt, dy: y - 1pt,
  circle(fill: aqua, radius: 2pt),
)

#mark((20pt, 20pt))

#curve(
  stroke: blue,
  curve.move((0pt, 100pt)),
  curve.quad((20pt, 20pt), (100pt, 0pt)),
)
```



```
curve.quad(
  none auto array,
  array,
  relative: bool,
) → content
```

#### /3.3.1 control none or auto or array *Obligatorio* *Posicional*

El punto de control de la curva de Bézier cuadrática.

- Si es `auto` y este segmento sigue a otra curva de Bézier cuadrática, se reflejará el punto de control anterior.
- Si es `none`, el punto de control por defecto es `end` y la curva será una línea recta.

```
#curve(
  stroke: 2pt,
```

```
  curve.quad((20pt, 40pt), (40pt, 40pt), relative: true),
  curve.quad(auto, (40pt, -40pt), relative: true),
)
```



/3.3.2 **end** `array` *Obligatorio* *Posicional*

El punto en el que debe terminar el segmento.

/3.3.3 **relative** `bool` *Configurable con set* Por defecto: `false`

Si las coordenadas `control` y `end` son relativas al punto anterior.

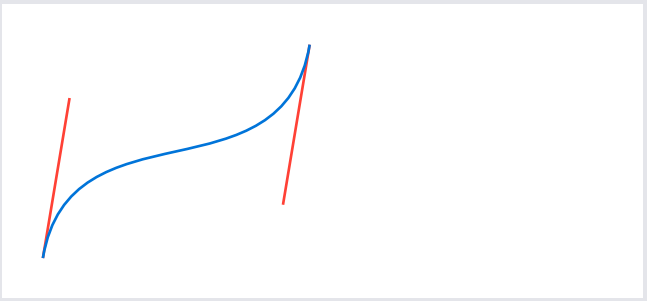
## /3.4 cubic Elemento Desde: 0.13.0

Agrega un segmento de curva de Bézier cúbica desde el último punto hasta `end`, usando `control-start` y `control-end` como puntos de control.

```
// Function to illustrate where the control points are.
#let handle(start, end) = place(
  line(stroke: red, start: start, end: end)
)

#handle((0pt, 80pt), (10pt, 20pt))
#handle((90pt, 60pt), (100pt, 0pt))

#curve(
  stroke: blue,
  curve.move((0pt, 80pt)),
  curve.cubic((10pt, 20pt), (90pt, 60pt), (100pt, 0pt)),
)
```



```
curve.cubic(
  none auto array,
  none array,
  array,
  relative: bool,
) → content
```

/3.4.1 **control-start** `none` or `auto` or `array` *Obligatorio* *Posicional*

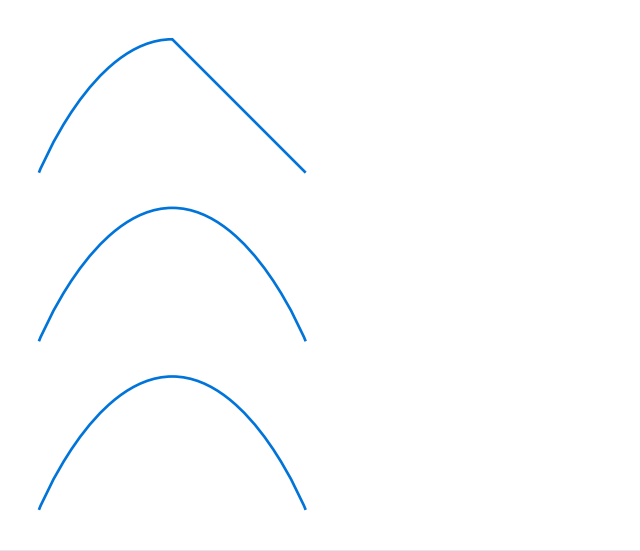
El punto de control que sale del inicio del segmento de curva.

- Si es auto y este elemento sigue a otro elemento curve.cubic, se reflejará el último punto de control. En términos de SVG, esto hace que curve.cubic se comporte como el operador S en lugar del operador C.
- Si es none, la curva no tiene primer punto de control o, de manera equivalente, el punto de control por defecto es el punto de inicio de la curva.

```
#curve(
  stroke: blue,
  curve.move((0pt, 50pt)),
  // - No start control point
  // - End control point at `(20pt, 0pt)`
  // - End point at `(50pt, 0pt)`
  curve.cubic(none, (20pt, 0pt), (50pt, 0pt)),
  // - No start control point
  // - No end control point
  // - End point at `(50pt, 0pt)`
  curve.cubic(none, none, (100pt, 50pt)),
)

#curve(
  stroke: blue,
  curve.move((0pt, 50pt)),
  curve.cubic(none, (20pt, 0pt), (50pt, 0pt)),
  // Passing `auto` instead of `none` means the start control point
  // mirrors the end control point of the previous curve. Mirror of
  // `(20pt, 0pt)` w.r.t `(50pt, 0pt)` is `(80pt, 0pt)`.
  curve.cubic(auto, none, (100pt, 50pt)),
)

#curve(
  stroke: blue,
  curve.move((0pt, 50pt)),
  curve.cubic(none, (20pt, 0pt), (50pt, 0pt)),
  // `(80pt, 0pt)` is the same as `auto` in this case.
  curve.cubic((80pt, 0pt), none, (100pt, 50pt)),
)
```



/3.4.2 **control-end** none or array *Obligatorio* *Posicional*

El punto de control que llega al punto final del segmento de curva.

Si se establece en none, la curva no tiene punto de control final o, de manera equivalente, el punto de control por defecto es el punto final de la curva.

/3.4.3 **end** `array` *Obligatorio* *Posicional*

El punto en el que debe terminar el segmento de curva.

/3.4.4 **relative** `bool` *Configurable con set* Por defecto: `false`

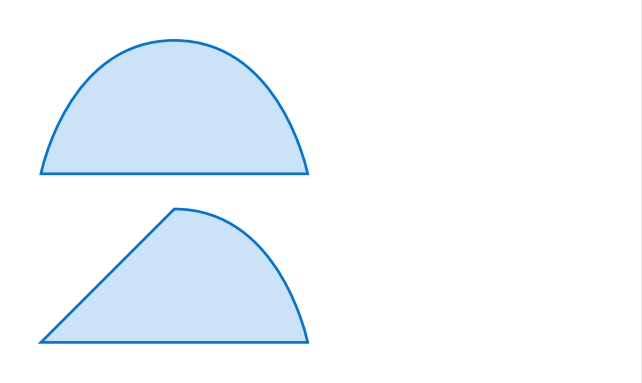
Si las coordenadas `control-start`, `control-end` y `end` son relativas al punto anterior.

/3.5 **close** Elemento Desde: 0.13.0

Cierra la curva agregando un segmento desde el último punto hasta el inicio de la curva (o hasta el último punto `curve.move` anterior).

```
// We define a function to show the same shape with
// both closing modes.
#let shape(mode: "smooth") = curve(
  fill: blue.lighten(80%),
  stroke: blue,
  curve.move((0pt, 50pt)),
  curve.line((100pt, 50pt)),
  curve.cubic(auto, (90pt, 0pt), (50pt, 0pt)),
  curve.close(mode: mode),
)

#shape(mode: "smooth")
#shape(mode: "straight")
```



```
curve.close(mode: str) → content
```

/3.5.1 **mode** `str` *Configurable con set* Por defecto: `"smooth"`

Cómo cerrar la curva.

| Variante | Detalles |
|---|---|
| `"smooth"` | Cierra la curva con un segmento suave que tiene en cuenta el punto de control opuesto al punto de inicio. |
| `"straight"` | Cierra la curva con una línea recta. |

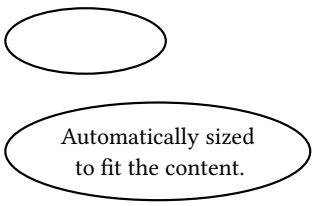
### 3.11.5 `ellipse` Elemento Desde: siempre

Una elipse con contenido opcional.

#### /1 Ejemplo

```
// Without content.
#ellipse(width: 35%, height: 30pt)

// With content.
#ellipse[
  #set align(center)
  Automatically sized \
  to fit the content.
]
```




#### /2 Parámetros

```
ellipse(
  width: auto relative,
  height: auto relative fraction,
  fill: none color gradient tiling,
  stroke: none auto length color gradient stroke tiling dictionary,
  inset: relative dictionary,
  outset: relative dictionary,
  none content,
) → content
```

/2.1 **width** `auto` or `relative` *Configurable con set* Por defecto: `auto`

El ancho de la elipse, relativo a su contenedor padre.

/2.2 **height** `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

La altura de la elipse, relativa a su contenedor padre.

/2.3 **fill** `none` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `none`

Cómo rellenar la elipse. Mirá la documentación del rectángulo[3.11.10/2.3] para más detalles.

/2.4 **stroke** `none` or `auto` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `auto`

Cómo trazar la elipse. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

/2.5 **inset** `relative` or `dictionary` *Configurable con set* Por defecto: `0% + 5pt`

Cuánto relleno se agrega al contenido de la elipse. Mirá la documentación de box[3.10.5/2.7] para más detalles.

/2.6 **outset** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se expande el tamaño de la elipse sin afectar el diseño. Mirá la documentación de box[3.10.5/2.8] para más detalles.

/2.7 **body** `none` or `content` *Posicional* *Configurable con set* Por defecto: `none`

El contenido que se coloca dentro de la elipse.

Cuando se omite, la elipse toma un tamaño por defecto de como máximo `45pt` por `30pt`.

### 3.11.6 gradient Desde: 0.9.0

Un degradado de color.

Typst admite degradados lineales mediante la función `gradient.linear`[3.11.6/7.1], degradados radiales mediante la función `gradient.radial`[3.11.6/7.2] y degradados cónicos mediante la función `gradient.conic`[3.11.6/7.3].

Un degradado se puede usar con los siguientes fines:
- Como relleno para pintar el interior de una forma: `rect(fill: gradient.linear(..))`
- Como trazo para pintar el contorno de una forma: `rect(stroke: 1pt + gradient.linear(..))`
- Como relleno de texto: `set text(fill: gradient.linear(..))`
- Como mapa de color del que podés muestrear[3.11.6/7.15]: `gradient.linear(..).sample(50%)`

#### /1 Ejemplos

```
#stack(
  dir: ltr,
  spacing: 1fr,
  square(fill: gradient.linear(..color.map.rainbow)),
  square(fill: gradient.radial(..color.map.rainbow)),
  square(fill: gradient.conic(..color.map.rainbow)),
)
```



Los degradados también se admiten en texto, pero solo cuando se establece la relatividad[3.11.6/7.9] en `auto` (el valor por defecto) o en `"parent"`. Para crear degradados palabra por palabra o glifo por glifo, podés envolver manualmente las palabras o los caracteres de tu texto en boxes[3.10.5] o mediante una regla show[3.2.2].

```
#set text(fill: gradient.linear(red, blue))
#let rainbow(content) = {
  set text(fill: gradient.linear(..color.map.rainbow))
  box(content)
}

This is a gradient on text, but with a #rainbow[twist]!
```

This is a gradient on text, but with a twist!

#### /2 Paradas

Un degradado se compone de una serie de paradas (stops). Cada una de estas paradas tiene un color y un desplazamiento. El desplazamiento es una proporción[3.10.20] entre `0%` y `100%` o un ángulo entre `0deg` y `360deg`. El desplazamiento es una posición relativa que determina qué tan lejos a lo largo del degradado está ubicada la parada. El color de la parada define el color que debe tener el degradado en esa posición.

Cada parada se pasa como argumento posicional y puede tener una de dos formas:
- Solo un color, como `red`, para que Typst ubique la parada automáticamente.
- Un array con un color y su desplazamiento, como `(red, 30%)`, para ubicar la parada vos mismo.

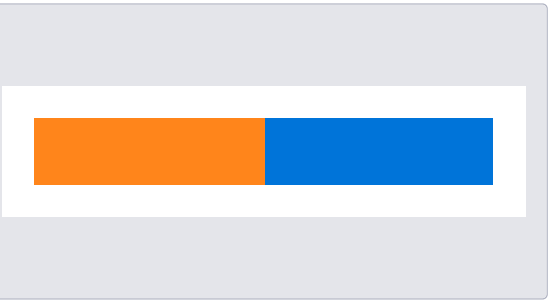
Podés elegir omitir los desplazamientos al definir un degradado. En ese caso, Typst espaciará todas las paradas de manera uniforme. De lo contrario, se deben proporcionar desplazamientos para *todas* las paradas. No deben disminuir de una parada a la siguiente, la primera debe ser `0%` y la última, `100%`.

```
#rect(
  width: 100%,
  fill: gradient.linear(
    (green, 0%),
    (red, 30%),
    (blue, 100%),
  ),
)
```



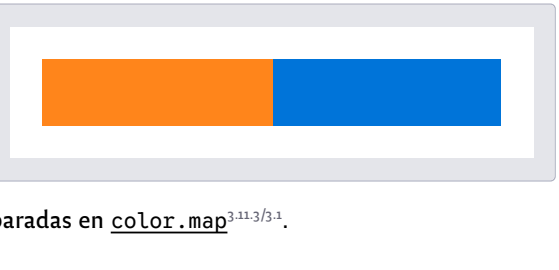
Darle el mismo desplazamiento a dos paradas crea un borde duro en lugar de una transición suave:

```
#rect(
  width: 100%,
  fill: gradient.linear(
    (orange, 0%),
    (orange, 50%),
    (blue, 50%),
    (blue, 100%),
  ),
)
```



Typst ofrece el método sharp[3.11.6/7.4] en los degradados para convertir automáticamente un degradado suave en uno con bordes duros. El ejemplo anterior se puede escribir de manera equivalente así:

```
#rect(
  width: 100%,
  fill: gradient.linear(orange, blue)
    .sharp(2),
)
```



Typst predefine mapas de color que podés usar como paradas en color.map[3.11.3/3.1].

## /3 Relatividad

La ubicación de las paradas `0%` y `100%` depende de las dimensiones de un contenedor. Este contenedor puede ser la forma sobre la que se está pintando o el contenedor circundante más cercano. Esto se controla con el argumento `relative` de un constructor de degradado. Por defecto, los degradados son relativos a la forma sobre la que se pintan, salvo que el degradado se aplique sobre texto, en cuyo caso son relativos al contenedor ancestro más cercano.

Typst determina el contenedor ancestro de la siguiente manera:
- Para las formas que se ubican en la raíz / nivel superior del documento, el ancestro más cercano es la propia página.
- Para las otras formas, el ancestro es el block[3.10.4] o box[3.10.5] más interno que contiene la forma.

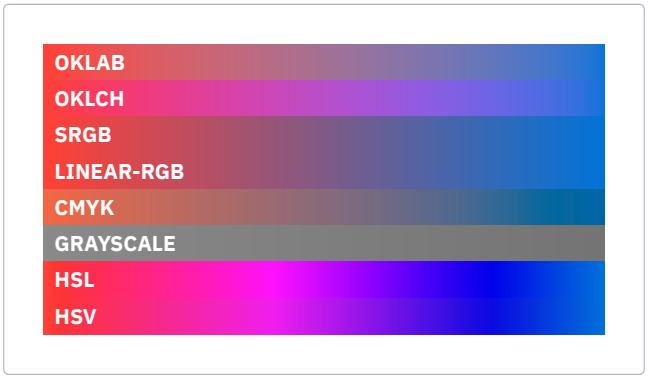
## /4 Espacios de color e interpolación

Los degradados se pueden interpolar en cualquier espacio de color. Por defecto, los degradados se interpolan en el espacio de color Oklab[3.11.3/4.2], que es un espacio de color perceptualmente uniforme[A]. Esto significa que el degradado se percibirá con una progresión suave de colores. Esto es particularmente útil para la visualización de datos.

Sin embargo, podés elegir interpolar el degradado en cualquier espacio de color admitido que quieras, pero tené en cuenta que algunos espacios de color no son adecuados para interpolar perceptualmente entre colores. Consultá la tabla de abajo al elegir un espacio de interpolación.

[A] https://programmingdesignsystems.com/color/perceptually-uniform-color-spaces/index.html

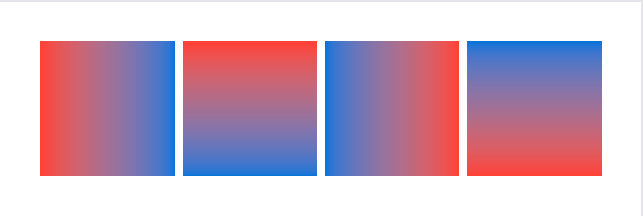
| Espacio de color | ¿Perceptualmente uniforme? |
|---|---|
| Oklab[3.11.3/4.2] | *Sí* |
| Oklch[3.11.3/4.3] | *Sí* |
| sRGB[3.11.3/4.5] | *No* |
| RGB lineal[3.11.3/4.4] | *Sí* |
| CMYK[3.11.3/4.6] | *No* |
| Escala de grises[3.11.3/4.1] | *Sí* |
| HSL[3.11.3/4.7] | *No* |
| HSV[3.11.3/4.8] | *No* |




## /5 Dirección

Algunos degradados son sensibles a la dirección. Por ejemplo, un degradado lineal tiene un ángulo que determina su dirección. Typst usa un ángulo en sentido horario, con 0° de izquierda a derecha, 90° de arriba hacia abajo, 180° de derecha a izquierda y 270° de abajo hacia arriba.

```
#stack(
  dir: ltr,
  spacing: 1fr,
  square(fill: gradient.linear(red, blue, angle: 0deg)),
  square(fill: gradient.linear(red, blue, angle: 90deg)),
  square(fill: gradient.linear(red, blue, angle: 180deg)),
  square(fill: gradient.linear(red, blue, angle: 270deg)),
)
```



## /6 Nota sobre los tamaños de archivo

Los degradados pueden ser bastante grandes, especialmente si tienen muchas paradas. Esto se debe a que los degradados se almacenan como una lista de colores y desplazamientos, que puede ocupar mucho espacio. En la exportación a SVG, los degradados se almacenan como una lista de colores `color.rgb`[3.11.3/4.5] que replican el espacio de color original con una cantidad optimizada de paradas adicionales en el medio. En la exportación a PDF, lo mismo se aplica a los degradados en los espacios de color `color.oklab`[3.11.3/4.2], `color.oklch`[3.11.3/4.3], `color.hsv`[3.11.3/4.8], `color.hsl`[3.11.3/4.7] y `color.linear-rgb`[3.11.3/4.4]. Esto evita tener que codificar estos espacios de color en tu archivo PDF, pero sí agrega paradas adicionales a tu degradado, lo que puede aumentar el tamaño del archivo.

## /7 Funciones

### /7.1 linear Desde: 0.9.0

Crea un degradado lineal nuevo, en el que los colores hacen la transición a lo largo de una línea recta.

```
#rect(
  width: 100%,
  height: 20pt,
  fill: gradient.linear(
    ..color.map.viridis,
  ),
)
```



```
gradient.linear(
  ..color array ,
  space: auto spot any ,
  relative: auto str ,
  direction ,
  angle ,
) → gradient
```

#### /7.1.1 stops color or array *Obligatorio* *Posicional* *Variádico*

Las paradas[3.11.6/2] de color del degradado.

Cada parada se da como un argumento posicional separado. Puede ser un color, como `red`, o un array con un color y su desplazamiento, como `(red, 30%)`. Si se dan desplazamientos, deben proporcionarse para todas las paradas, no deben disminuir de una parada a la siguiente, deben empezar en `0%` y terminar en `100%`.

Al usar un mapa de color predefinido del módulo `color.map`, se usa el operador de expansión[3.5.1/2] `..` para pasar cada parada como un argumento separado.

#### /7.1.2 space auto or spot or any Por defecto: auto

El espacio de color en el que se interpola el degradado.

Por defecto es un espacio de color perceptualmente uniforme llamado Oklab[3.11.3/4.2].

#### /7.1.3 relative auto or str Por defecto: auto

La ubicación relativa[3.11.6/3] del degradado.

El padre de un elemento es el box[3.10.5] o block[3.10.4] más interno que contiene el elemento o, si no hay ninguno, la propia página.

| Variante | Detalles |
|---|---|
| `"self"` | Relativo a sí mismo (su propio cuadro delimitador). |
| `"parent"` | Relativo a su elemento padre (el cuadro delimitador del padre). |

#### /7.1.4 dir direction *Posicional* Por defecto: ltr

La dirección del degradado.

#### /7.1.5 angle angle *Obligatorio* *Posicional*

El ángulo del degradado.

### /7.2 radial Desde: 0.9.0

Crea un degradado radial nuevo, en el que los colores irradian desde un origen.

El degradado se define con dos círculos: el círculo focal y el círculo final. El círculo focal es un círculo con centro `focal-center` y radio `focal-radius`, que define los puntos en los que el degradado empieza y tiene el color de la primera parada. El círculo final es un círculo con centro `center` y radio `radius`, que define los

puntos en los que el degradado termina y tiene el color de la última parada. Luego el degradado se interpola entre estos dos círculos.

Con estos cuatro valores, también llamados punto focal para el círculo inicial y centro y radio para el círculo final, podemos definir un degradado con propiedades más interesantes que un degradado radial básico.

```
#stack(
  dir: ltr,
  spacing: 1fr,
  circle(fill: gradient.radial(
    ..color.map.viridis,
  )),
  circle(fill: gradient.radial(
    ..color.map.viridis,
    focal-center: (10%, 40%),
    focal-radius: 5%,
  )),
)
```



```
gradient.radial(
  ..color array ,
  space: auto spot any ,
  relative: auto str ,
  center: array ,
  radius: ratio ,
  focal-center: auto array ,
  focal-radius: ratio ,
) → gradient
```

/7.2.1 **stops** color or array *Obligatorio* *Posicional* *Variádico*

Las paradas[3.11.6/2] de color del degradado.

Mirá también la documentación del degradado lineal[3.11.6/7.1.1] para más detalles.

/7.2.2 **space** auto or spot or any Por defecto: auto

El espacio de color en el que se interpola el degradado.

Por defecto es un espacio de color perceptualmente uniforme llamado Oklab[3.11.3/4.2].

/7.2.3 **relative** auto or str Por defecto: auto

La ubicación relativa[3.11.6/3] del degradado.

El padre de un elemento es el box[3.10.5] o block[3.10.4] más interno que contiene el elemento o, si no hay ninguno, la propia página.

| Variante | Detalles |
|---|---|
| "self" | Relativo a sí mismo (su propio cuadro delimitador). |
| "parent" | Relativo a su elemento padre (el cuadro delimitador del padre). |

/7.2.4 **center** array Por defecto: (50%, 50%)

El centro del círculo final del degradado.

Un valor de (50%, 50%) significa que el círculo final está centrado dentro de su contenedor.

/7.2.5 **radius** ratio Por defecto: 50%

El radio del círculo final del degradado.

Por defecto, se establece en 50%. El radio final debe ser mayor que el radio focal.

/7.2.6 **focal-center** auto or array — Por defecto: auto

El centro del círculo focal del degradado.

El centro focal debe estar dentro del círculo final.

Un valor de (50%, 50%) significa que el círculo focal está centrado dentro de su contenedor.

Por defecto, se establece igual que el centro del último círculo.

/7.2.7 **focal-radius** ratio — Por defecto: 0%

El radio del círculo focal del degradado.

El centro focal debe estar dentro del círculo final.

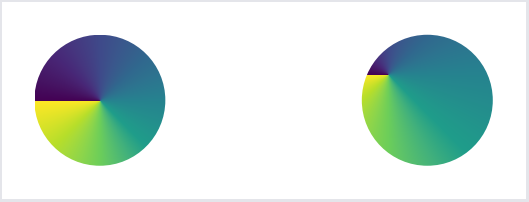
Por defecto, se establece en 0%. El radio focal debe ser menor que el radio final.

## /7.3 conic Desde: 0.9.0

Crea un degradado cónico nuevo, en el que los colores cambian radialmente alrededor de un punto central.

Podés controlar el punto central del degradado con el argumento `center`. Por defecto, el punto central es el centro de la forma.

```
#stack(
  dir: ltr,
  spacing: 1fr,
  circle(fill: gradient.conic(
    ..color.map.viridis,
  )),
  circle(fill: gradient.conic(
    ..color.map.viridis,
    center: (20%, 30%),
  )),
)
```



```
gradient.conic(
  ..color array ,
  angle: angle ,
  space: auto spot any ,
  relative: auto str ,
  center: array ,
) → gradient
```

/7.3.1 **stops** color or array *Obligatorio* *Posicional* *Variádico*

Las paradas[3.11.6/2] de color del degradado.

Mirá también la documentación del degradado lineal[3.11.6/7.1.1] para más detalles.

/7.3.2 **angle** angle — Por defecto: 0deg

El ángulo del degradado.

/7.3.3 **space** auto or spot or any — Por defecto: auto

El espacio de color en el que se interpola el degradado.

Por defecto es un espacio de color perceptualmente uniforme llamado Oklab[3.11.3/4.2].

/7.3.4 **relative** auto or str — Por defecto: auto

La ubicación relativa[3.11.6/3] del degradado.

El padre de un elemento es el box[3.10.5] o block[3.10.4] más interno que contiene el elemento o, si no hay ninguno, la propia página.

| Variante | Detalles |
|---|---|
| `"self"` | Relativo a sí mismo (su propio cuadro delimitador). |
| `"parent"` | Relativo a su elemento padre (el cuadro delimitador del padre). |

/7.3.5 **center** `array` Por defecto: `(50%, 50%)`

El centro del círculo del degradado.

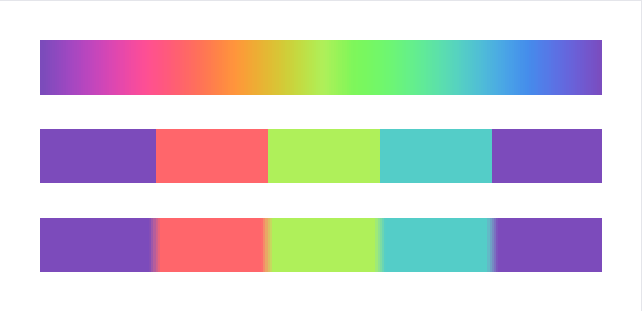
Un valor de `(50%, 50%)` significa que el círculo está centrado dentro de su contenedor.

## /7.4 sharp Desde: 0.9.0

Crea una versión nítida de este degradado.

Los degradados nítidos tienen saltos discretos entre colores, en lugar de una transición suave. Son particularmente útiles para crear listas de colores para un degradado predefinido.

```
#set rect(width: 100%, height: 20pt)
#let grad = gradient.linear(..color.map.rainbow)
#rect(fill: grad)
#rect(fill: grad.sharp(5))
#rect(fill: grad.sharp(5, smoothness: 20%))
```



```
self.sharp(
  int,
  smoothness: ratio,
) → gradient
```

/7.4.1 **steps** `int` *Obligatorio* *Posicional*

La cantidad de paradas del degradado.

/7.4.2 **smoothness** `ratio` Por defecto: `0%`

Cuánto se suaviza el degradado.

## /7.5 repeat Desde: 0.9.0

Repite este degradado una cantidad dada de veces, opcionalmente reflejándolo en cada segunda repetición.

```
#circle(
  radius: 40pt,
  fill: gradient
    .radial(aqua, white)
    .repeat(4),
)
```



```
self.repeat(
  int,
  mirror: bool,
) → gradient
```

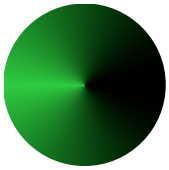
/7.5.1 **repetitions** `int` *Obligatorio* *Posicional*

La cantidad de veces que se repite el degradado.

/7.5.2 **mirror** `bool` Por defecto: `false`

Si se refleja el degradado en cada segunda repetición, es decir, la primera instancia (y todas las impares) permanece sin cambios.

```
#circle(
  radius: 40pt,
  fill: gradient
    .conic(green, black)
    .repeat(2, mirror: true)
)
```



/7.6 **kind** Desde: 0.9.0

Devuelve el tipo de este degradado.

```
self.kind() → function
```

/7.7 **stops** Desde: 0.9.0

Devuelve las paradas de este degradado.

```
self.stops() → array
```

/7.8 **space** Desde: 0.9.0

Devuelve el espacio de mezcla de este degradado.

```
self.space() → spot any
```

/7.9 **relative** Desde: 0.9.0

Devuelve la ubicación relativa de este degradado.

```
self.relative() → auto str
```

/7.10 **angle** Desde: 0.9.0

Devuelve el ángulo de este degradado.

Devuelve none si el degradado no es ni lineal ni cónico.

```
self.angle() → none angle
```

### /7.11 center Desde: 0.13.0

Devuelve el centro de este degradado.

Devuelve none si el degradado no es ni radial ni cónico.

```
self.center() → none array
```

### /7.12 radius Desde: 0.13.0

Devuelve el radio de este degradado.

Devuelve none si el degradado no es radial.

```
self.radius() → none ratio
```

### /7.13 focal-center Desde: 0.13.0

Devuelve el centro focal de este degradado.

Devuelve none si el degradado no es radial.

```
self.focal-center() → none array
```

### /7.14 focal-radius Desde: 0.13.0

Devuelve el radio focal de este degradado.

Devuelve none si el degradado no es radial.

```
self.focal-radius() → none ratio
```

### /7.15 sample Desde: 0.9.0

Muestrea el degradado en una posición dada.

La posición es una posición a lo largo del degradado (una proporción[3.10.20] entre 0% y 100%) o un ángulo[3.10.3]. Cualquier valor fuera de este rango se acotará.

```
self.sample( angle ratio ) → color
```

#### /7.15.1 t angle or ratio *Obligatorio* *Posicional*

La posición en la que se muestrea el degradado.

### /7.16 samples Desde: 0.9.0

Muestrea el degradado en varias posiciones a la vez y devuelve los resultados como un array.

```
self.samples(.. angle ratio ) → array
```

#### /7.16.1 ts angle or ratio *Obligatorio* *Posicional* *Variádico*

Las posiciones en las que se muestrea el degradado.

### 3.11.7 `image` Elemento Desde: siempre

Un gráfico ráster o vectorial.

Podés envolver la imagen en una `figure`[3.6.7] para darle un número y un epígrafe.

Como la mayoría de los elementos, las imágenes son *de bloque* por defecto y, por lo tanto, no se integran en los párrafos adyacentes. Para forzar que una imagen sea en línea, ponela en un `box`[3.10.5].

#### /1 Ejemplo

```
#figure(
  image("molecular.jpg", width: 80%),
  caption: [
    A step in the molecular testing
    pipeline of our lab.
  ],
)
```

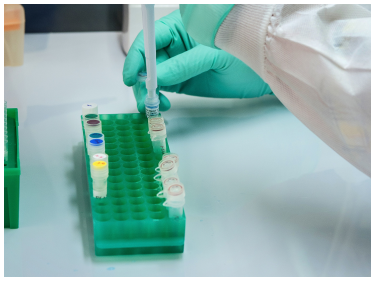


Figura 1: A step in the molecular testing pipeline of our lab.

#### /2 Recorte

Podés envolver una imagen en un `block`[3.10.4] o un `box`[3.10.5][A] con un `inset`[3.10.4/2.7] negativo y `clip: true` para recortarla. Tené en cuenta que las partes recortadas solo quedan ocultas visualmente. Todos los datos de la imagen siguen incrustados en la salida (excepto al exportar a PNG). Por lo tanto, este enfoque no es adecuado para censurar partes de una imagen.

```
#let lynx = image("lynx.jpg", height: 150pt, fit: "cover")
#grid(
  columns: 2,
  column-gutter: 1fr,
  // The full image on the left.
  lynx,
  // Two cropped parts on the right: One cropped by 80pt
  // from the bottom and one cropped by 75pt from the top.
  stack(
    spacing: 5pt,
    block(lynx, clip: true, inset: (bottom: -80pt)),
    block(lynx, clip: true, inset: (top: -75pt)),
  )
)
```

[A] Un box solo debería usarse si la imagen se va a mostrar en línea como parte de un párrafo. De lo contrario, es preferible un bloque.



## /3 Parámetros

```
image(
  str path bytes,
  format: auto str dictionary,
  width: auto relative,
  height: auto relative fraction,
  alt: none str,
  page: int,
  fit: str,
  scaling: auto str,
  icc: auto str path bytes,
) → content
```

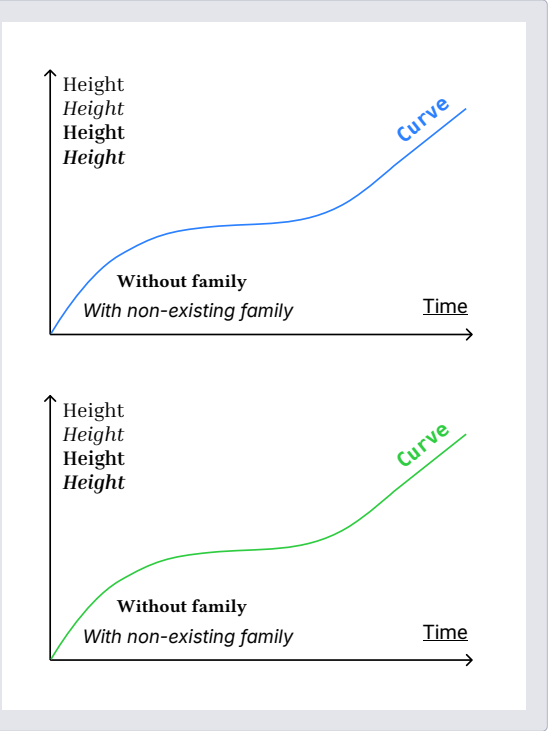
### /3.1 source `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo de imagen o bytes sin procesar que componen una imagen en uno de los formatos[3.11.7/3.2] admitidos.

Se pueden usar bytes para especificar datos de píxeles sin procesar en formato de filas, de izquierda a derecha y de arriba hacia abajo.

```
#let original = read("diagram.svg")
#let changed = original.replace(
  "#2B80FF", // blue
  green.to-hex(),
)

#image(bytes(original))
#image(bytes(changed))
```




/3.2 **format** `auto` or `str` or `dictionary` *Configurable con set* Por defecto: `auto`

El formato de la imagen.

Por defecto, el formato se detecta automáticamente. Por lo tanto, normalmente solo necesitás especificarlo al proporcionar bytes sin procesar como `source`[3.11.7/3.1] (incluso entonces, Typst intentará averiguar el formato automáticamente, pero no siempre es posible).

Los formatos admitidos son `"png"`, `"jpg"`, `"gif"`, `"svg"`, `"pdf"` y `"webp"`, así como datos de píxeles sin procesar.

Tené en cuenta que se aplican varias restricciones al usar archivos PDF como imágenes:

- Al exportar a PDF, cualquier archivo de imagen PDF que se use debe tener una versión igual o inferior a la versión de PDF del destino de exportación[3.14.4/3.2.1].
- Actualmente no se admiten archivos PDF como imágenes al exportar con un estándar de PDF específico, como PDF/A-3 o PDF/UA-1. En esos casos, podés usar SVG para incrustar imágenes vectoriales.
- El archivo de imagen no debe estar protegido con contraseña.
- Las etiquetas (tags) de tu imagen PDF no se conservarán. En cambio, tenés que proporcionar una descripción alternativa[3.11.7/3.5] para que la imagen sea accesible.

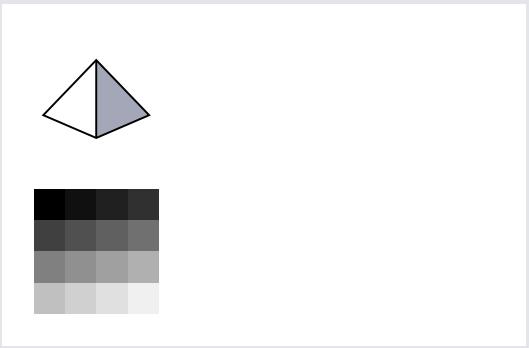
Al proporcionar datos de píxeles sin procesar como `source`, tenés que especificar como `format` un diccionario con las siguientes claves:

- `encoding` (str[3.5.28]): La codificación de los datos de píxeles. Una de:
  - `"rgb8"` (tres canales de 8 bits: rojo, verde, azul)
  - `"rgba8"` (cuatro canales de 8 bits: rojo, verde, azul, alfa)
  - `"luma8"` (un canal de 8 bits)
  - `"lumaa8"` (dos canales de 8 bits: luma y alfa)
- `width` (int[3.5.17]): El ancho de la imagen en píxeles.
- `height` (int[3.5.17]): El alto de la imagen en píxeles.

El ancho en píxeles multiplicado por el alto multiplicado por la cantidad de canales de la codificación especificada debe coincidir entonces con los datos de `source`.

```
#image(
  read(
    "tetrahedron.svg",
    encoding: none,
  ),
  format: "svg",
  width: 2cm,
)

#image(
  bytes(range(16).map(x => x * 16)),
  format: (
    encoding: "luma8",
    width: 4,
    height: 4,
  ),
  width: 2cm,
)
```



| Variante | Detalles |
|---|---|
| `"png"` | Formato de trama para ilustraciones y gráficos con transparencia. |
| `"jpg"` | Formato de trama con pérdida, adecuado para fotos. |
| `"gif"` | Formato de trama que suele usarse para clips animados cortos. Typst puede cargar GIF, pero quedan estáticos. |
| `"webp"` | Formato de trama que admite compresión con y sin pérdida. |
| `"svg"` | El formato de gráficos vectoriales de la web. |
| `"pdf"` | Formato de documentos y gráficos de alta fidelidad, centrado en la reproducción exacta para impresión. |

/3.3 **`width`** `auto` or `relative` *Configurable con set* Por defecto: `auto`

El ancho de la imagen.

/3.4 **`height`** `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

El alto de la imagen.

/3.5 **`alt`** `none` or `str` *Configurable con set* Por defecto: `none`

Una descripción alternativa de la imagen.

Las tecnologías de asistencia (TA), como los lectores de pantalla, usan este texto para describirles la imagen a los usuarios con discapacidad visual.

Cuando la imagen está envuelta en una `figure`[3.6.7], usá este parámetro en lugar del parámetro `alt` de la figura[3.6.7/5.2] para describir la imagen. La única excepción a esta regla es cuando la imagen y el resto del contenido de la figura forman una única unidad semántica. En ese caso, usá el parámetro `alt` de la figura para describir toda la composición y no uses este parámetro.

Podés aprender a escribir buenas descripciones alternativas en la guía de accesibilidad[4.4.7].

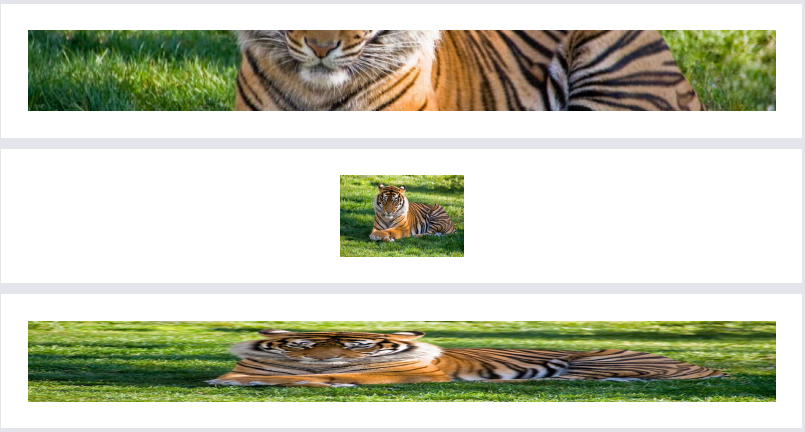
/3.6 **`page`** `int` *Configurable con set* Por defecto: `1`

El número de página que se debe incrustar como imagen. Este atributo solo tiene efecto para archivos PDF.

/3.7 **`fit`** `str` *Configurable con set* Por defecto: `"cover"`

Cómo debe ajustarse la imagen a un área dada (el área está definida por los campos `width` y `height`). Tené en cuenta que `fit` no cambia nada visualmente si la relación de aspecto del área es la misma que la de la imagen.

```
#set page(width: 300pt, height: 50pt, margin: 10pt)
#image("tiger.jpg", width: 100%, fit: "cover")
#image("tiger.jpg", width: 100%, fit: "contain")
#image("tiger.jpg", width: 100%, fit: "stretch")
```



| Variante | Detalles |
|---|---|
| `"cover"` | La imagen debe cubrir por completo el área (conserva la relación de aspecto recortando la imagen solo en horizontal o en vertical). Este es el valor predeterminado. |
| `"contain"` | La imagen debe quedar totalmente contenida en el área (conserva la relación de aspecto; no recorta la imagen; una de las dimensiones puede ser menor que la especificada). |
| `"stretch"` | La imagen debe estirarse para llenar exactamente el área, aunque eso signifique que se deforme (no conserva la relación de aspecto y no recorta la imagen). |

/3.8 **`scaling`** `auto` or `str` *Configurable con set* Por defecto: `auto`

Una indicación a los visores sobre cómo deben escalar la imagen.

Cuando se establece en `auto`, el valor por defecto queda a criterio del visor. En la exportación a PNG, Typst usará por defecto un escalado suave, como la mayoría de los visores de PDF y SVG.

*Nota:* El aspecto exacto puede diferir entre los visores de PDF.

| Variante | Detalles |
|---|---|
| `"smooth"` | Escala con un algoritmo de suavizado, como la interpolación bilineal. |
| `"pixelated"` | Escala con vecino más cercano o un algoritmo similar para conservar el aspecto pixelado de la imagen. |

/3.9 **`icc`** `auto` or `str` or `path` or `bytes` *Configurable con set* Por defecto: `auto`

Un perfil ICC para la imagen.

Los perfiles ICC definen cómo interpretar los colores de una imagen. Cuando se establece en `auto`, Typst intentará extraer un perfil ICC de la imagen.

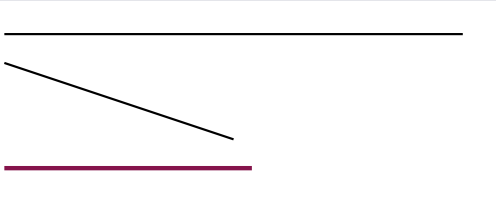
## 3.11.8 `line` Elemento Desde: siempre

Una línea de un punto a otro.

### /1 Ejemplo

```typ
#set page(height: 100pt)

#line(length: 100%)
#line(end: (50%, 50%))
#line(
  length: 4cm,
  stroke: 2pt + maroon,
)
```



### /2 Parámetros

```
line(
  start: array,
  end: none array,
  length: relative,
  angle: angle,
  stroke: length color gradient stroke tiling dictionary,
) → content
```

/2.1 **start** `array` *Configurable con set* Por defecto: `(0% + 0pt, 0% + 0pt)`

El punto de inicio de la línea.

Debe ser un array de exactamente dos longitudes relativas.

/2.2 **end** `none` or `array` *Configurable con set* Por defecto: `none`

El punto donde termina la línea.

/2.3 **length** `relative` *Configurable con set* Por defecto: `0% + 30pt`

La longitud de la línea. Solo se respeta si `end` es `none`.

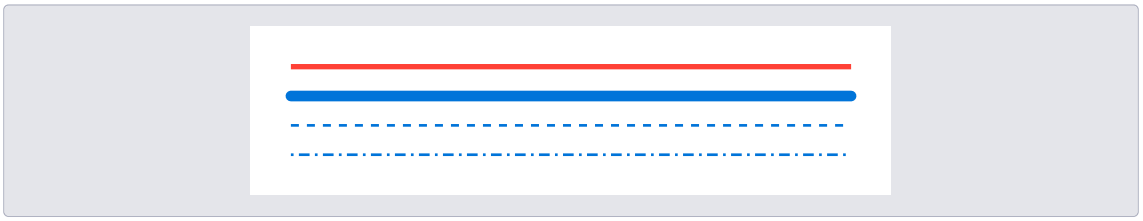
/2.4 **angle** `angle` *Configurable con set* Por defecto: `0deg`

El ángulo con el que la línea se aleja del origen. Solo se respeta si `end` es `none`.

/2.5 **stroke** `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `stroke()`

Cómo trazar[3.11.12] la línea.

```typ
#set line(length: 100%)
#stack(
  spacing: 1em,
  line(stroke: 2pt + red),
  line(stroke: (paint: blue, thickness: 4pt, cap: "round")),
  line(stroke: (paint: blue, thickness: 1pt, dash: "dashed")),
  line(stroke: (paint: blue, thickness: 1pt, dash: ("dot", 2pt, 4pt, 2pt))),
)
```



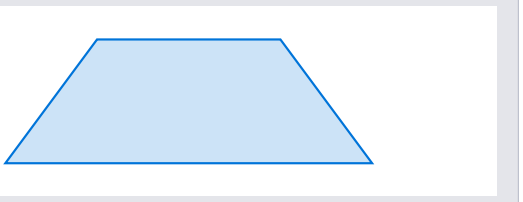
## 3.11.9 polygon Elemento Desde: 0.1.0

Un polígono cerrado.

El polígono se define por sus puntos de esquina y se cierra automáticamente.

### /1 Ejemplo

```
#polygon(
  fill: blue.lighten(80%),
  stroke: blue,
  (20%, 0pt),
  (60%, 0pt),
  (80%, 2cm),
  (0%,  2cm),
)
```



### /2 Parámetros

```
polygon(
  fill: none color gradient tiling,
  fill-rule: str,
  stroke: none auto length color gradient stroke tiling dictionary,
  ..array,
) → content
```

#### /2.1 fill none or color or gradient or tiling *Configurable con set* Por defecto: none

Cómo rellenar el polígono.

Al establecer un relleno, el trazo por defecto desaparece. Para crear un rectángulo con relleno y trazo, tenés que configurar ambos.

#### /2.2 fill-rule str *Configurable con set* Por defecto: "non-zero"

La regla de dibujo que se usa para rellenar el polígono.

Mirá la documentación de curve[3.11.4/2.2] para ver un ejemplo.

| Variante | Detalles |
|---|---|
| "non-zero" | Especifica que el "interior" se calcula con una suma distinta de cero de cruces de bordes con signo. |
| "even-odd" | Especifica que el "interior" se calcula con una cantidad impar de cruces de bordes. |

#### /2.3 stroke none or auto or length or color or gradient or stroke or tiling or dictionary *Configurable con set* Por defecto: auto

Cómo trazar[3.11.12] el polígono.

Se puede establecer en none para desactivar el trazo o en auto para un trazo negro de 1pt si y solo si no se da un relleno.

#### /2.4 vertices array *Obligatorio* *Posicional* *Variádico*

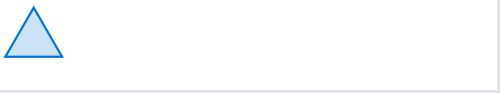
Los vértices del polígono. Cada punto se especifica como un array de dos longitudes relativas[3.10.21].

### /3 Funciones

#### /3.1 regular Desde: 0.8.0

Un polígono regular, definido por su tamaño y su cantidad de vértices.

```
#polygon.regular(
  fill: blue.lighten(80%),
  stroke: blue,
  size: 30pt,
  vertices: 3,
)
```



```
polygon.regular(
  fill: none color gradient tiling,
  stroke: none auto length color gradient stroke tiling dictionary,
  size: length,
  vertices: int,
) → content
```

/3.1.1 **fill** none or color or gradient or tiling

Cómo rellenar el polígono. Mirá la documentación general de polygon[3.11.9/2.1] para más detalles.

/3.1.2 **stroke** none or auto or length or color or gradient or stroke or tiling or dictionary

Cómo trazar el polígono. Mirá la documentación general de polygon[3.11.9/2.3] para más detalles.

/3.1.3 **size** length — Por defecto: 1em

El diámetro de la circunferencia circunscrita[A] del polígono regular.

/3.1.4 **vertices** int — Por defecto: 3

La cantidad de vértices del polígono.

[A] https://en.wikipedia.org/wiki/Circumcircle

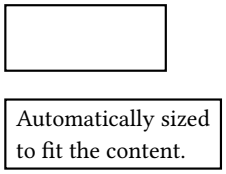
## 3.11.10 `rect` Elemento Desde: siempre

Un rectángulo con contenido opcional.

### /1 Ejemplo

```
// Without content.
#rect(width: 35%, height: 30pt)

// With content.
#rect[
  Automatically sized \
  to fit the content.
]
```



Automatically sized
to fit the content.

### /2 Parámetros

```
rect(
  width: auto relative,
  height: auto relative fraction,
  fill: none color gradient tiling,
  stroke: none auto length color gradient stroke tiling dictionary,
  radius: relative dictionary,
  inset: relative dictionary,
  outset: relative dictionary,
  none content,
) → content
```

#### /2.1 `width` `auto` or `relative` *Configurable con set* Por defecto: `auto`

El ancho del rectángulo, relativo a su contenedor padre.

#### /2.2 `height` `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

La altura del rectángulo, relativa a su contenedor padre.

#### /2.3 `fill` `none` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `none`

Cómo rellenar el rectángulo.

Al establecer un relleno, el trazo por defecto desaparece. Para crear un rectángulo con relleno y trazo, tenés que configurar ambos.

```
#rect(fill: blue)
```



#### /2.4 `stroke` `none` or `auto` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `auto`

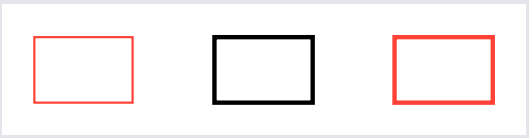
Cómo trazar el rectángulo. Puede ser:

- `none` para desactivar el trazo
- `auto` para un trazo de `1pt + black` si y solo si no se da un relleno.
- Cualquier tipo de trazo[3.11.12]
- Un diccionario que describe el trazo de cada lado individualmente. El diccionario puede contener las siguientes claves, en orden de precedencia:

- `top`: El trazo superior.
- `right`: El trazo derecho.
- `bottom`: El trazo inferior.
- `left`: El trazo izquierdo.
- `x`: El trazo izquierdo y derecho.
- `y`: El trazo superior e inferior.
- `rest`: El trazo de todos los lados excepto aquellos para los que el diccionario establece explícitamente un tamaño.

Todas las claves son opcionales; las claves omitidas usarán su valor establecido previamente o el trazo por defecto si nunca se establecieron.

```
#stack(
  dir: ltr,
  spacing: 1fr,
  rect(stroke: red),
  rect(stroke: 2pt),
  rect(stroke: 2pt + red),
)
```

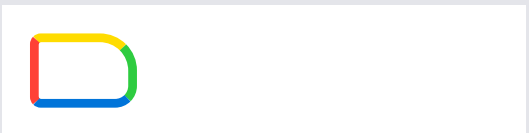


/2.5 **radius** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se redondean las esquinas del rectángulo, relativo al mínimo entre el ancho y el alto dividido por dos. Puede ser:

- Una longitud relativa para un radio de esquina uniforme.
- Un diccionario: con un diccionario, el radio de cada esquina se puede establecer individualmente. El diccionario puede contener las siguientes claves, en orden de precedencia:
  - `top-left`: El radio de la esquina superior izquierda.
  - `top-right`: El radio de la esquina superior derecha.
  - `bottom-right`: El radio de la esquina inferior derecha.
  - `bottom-left`: El radio de la esquina inferior izquierda.
  - `left`: Los radios de las esquinas superior izquierda e inferior izquierda.
  - `top`: Los radios de las esquinas superior izquierda y superior derecha.
  - `right`: Los radios de las esquinas superior derecha e inferior derecha.
  - `bottom`: Los radios de las esquinas inferior izquierda e inferior derecha.
  - `rest`: Los radios de todas las esquinas excepto aquellas para las que el diccionario establece explícitamente un tamaño.

```
#set rect(stroke: 4pt)
#rect(
  radius: (
    left: 5pt,
    top-right: 20pt,
    bottom-right: 10pt,
  ),
  stroke: (
    left: red,
    top: yellow,
    right: green,
    bottom: blue,
  ),
)
```



/2.6 **inset** `relative` or `dictionary` *Configurable con set* Por defecto: `0% + 5pt`

Cuánto relleno se agrega al contenido del rectángulo. Mirá la documentación de box[3.10.5/2.7] para más detalles.

/2.7 **outset** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se expande el tamaño del rectángulo sin afectar el diseño. Mirá la documentación de box[3.10.5/2.8] para más detalles.

/2.8 **body** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido que se coloca dentro del rectángulo.

Cuando se omite, el rectángulo toma un tamaño por defecto de como máximo `45pt` por `30pt`.

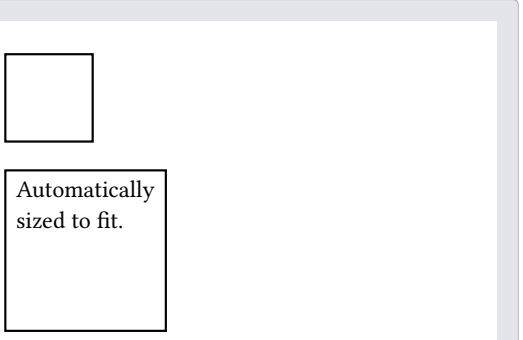
## 3.11.11 `square` Elemento Desde: siempre

Un cuadrado con contenido opcional.

### /1 Ejemplo

```
// Without content.
#square(size: 40pt)

// With content.
#square[
  Automatically \
  sized to fit.
]
```




### /2 Parámetros

```
square(
  size: auto length,
  width: auto relative,
  height: auto relative fraction,
  fill: none color gradient tiling,
  stroke: none auto length color gradient stroke tiling dictionary,
  radius: relative dictionary,
  inset: relative dictionary,
  outset: relative dictionary,
  none content,
) → content
```

#### /2.1 `size` `auto` or `length` Por defecto: `auto`

La longitud del lado del cuadrado. Es mutuamente excluyente con `width` y `height`.

#### /2.2 `width` `auto` or `relative` *Configurable con set* Por defecto: `auto`

El ancho del cuadrado. Es mutuamente excluyente con `size` y `height`.

A diferencia de `size`, esto puede ser relativo al ancho del contenedor padre.

#### /2.3 `height` `auto` or `relative` or `fraction` *Configurable con set* Por defecto: `auto`

La altura del cuadrado. Es mutuamente excluyente con `size` y `width`.

A diferencia de `size`, esto puede ser relativo a la altura del contenedor padre.

#### /2.4 `fill` `none` or `color` or `gradient` or `tiling` *Configurable con set* Por defecto: `none`

Cómo rellenar el cuadrado. Mirá la documentación del rectángulo[3.11.10/2.3] para más detalles.

#### /2.5 `stroke` `none` or `auto` or `length` or `color` or `gradient` or `stroke` or `tiling` or `dictionary` *Configurable con set* Por defecto: `auto`

Cómo trazar el cuadrado. Mirá la documentación del rectángulo[3.11.10/2.4] para más detalles.

#### /2.6 `radius` `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se redondean las esquinas del cuadrado. Mirá la documentación del rectángulo[3.11.10/2.5] para más detalles.

/2.7 **inset** `relative` or `dictionary` *Configurable con set* Por defecto: `0% + 5pt`

Cuánto relleno se agrega al contenido del cuadrado. Mirá la documentación de box[3.10.5/2.7] para más detalles.

/2.8 **outset** `relative` or `dictionary` *Configurable con set* Por defecto: `(:)`

Cuánto se expande el tamaño del cuadrado sin afectar el diseño. Mirá la documentación de box[3.10.5/2.8] para más detalles.

/2.9 **body** `none` or `content` *Posicional* *Configurable con set* Por defecto: none

El contenido que se coloca dentro del cuadrado. El cuadrado se expande para ajustarse a este contenido, manteniendo la relación de aspecto 1-1.

Cuando se omite, el cuadrado toma un tamaño por defecto de como máximo `30pt`.

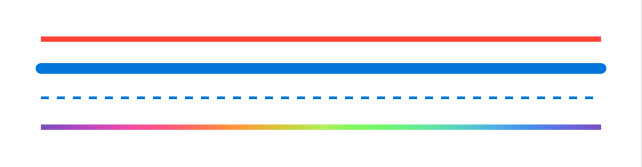
### 3.11.12 stroke Desde: siempre

Define cómo dibujar una línea.

Un trazo tiene una *pintura* (un color sólido o un degradado), un *grosor*, un *remate* de línea (cap), una *unión* de línea (join), un *límite de inglete* (miter limit) y un patrón de *guiones* (dash). Todos estos valores son opcionales y tienen valores por defecto razonables.

#### /1 Ejemplo

```
#set line(length: 100%)
#stack(
  spacing: 1em,
  line(stroke: 2pt + red),
  line(stroke: (paint: blue, thickness: 4pt, cap: "round")),
  line(stroke: (paint: blue, thickness: 1pt, dash: "dashed")),
  line(stroke: 2pt + gradient.linear(..color.map.rainbow)),
)
```



#### /2 Trazos simples

Podés crear un trazo sólido simple a partir de un color, un grosor o una combinación de ambos. Específicamente, dondequiera que se espere un trazo podés pasar cualquiera de los siguientes valores:

- Una longitud que especifica el grosor del trazo. El color se hereda, con negro por defecto.
- Un color para usar en el trazo. El grosor se hereda, con `1pt` por defecto.
- Un trazo combinado de color y grosor con el operador +, como en `2pt + red`.

Para tener control total, también podés proporcionar un diccionario[3.5.12] o un objeto `stroke` a cualquier función que espere un trazo. Las claves del diccionario pueden incluir cualquiera de los parámetros de la función constructora, que se muestran abajo.

#### /3 Campos

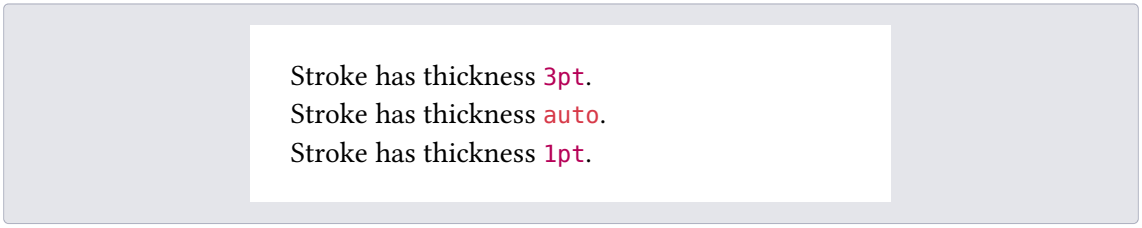
En un objeto trazo, podés acceder a cualquiera de los campos enumerados en la función constructora. Por ejemplo, `(2pt + blue).thickness` es `2pt`. Mientras tanto, `stroke(red).cap` es `auto` porque no está especificado. Los campos establecidos en `auto` se heredan.

#### /4 Constructor

Convierte un valor en un trazo o construye un trazo con los parámetros dados.

Tené en cuenta que, en la mayoría de los casos, no necesitás convertir valores en trazos para usarlos, ya que se convertirán automáticamente. Sin embargo, este constructor puede ser útil para asegurarte de que un valor tenga todos los campos de un trazo.

```
#let my-func(x) = {
    x = stroke(x) // Convert to a stroke
    [Stroke has thickness #x.thickness.]
}
#my-func(3pt) \
#my-func(red) \
#my-func(stroke(cap: "round", thickness: 1pt))
```




```
stroke(
  auto color gradient tiling,
  auto length,
  auto str,
  auto str,
  none auto str array dictionary,
  auto float,
) → stroke
```

/4.1 **paint** auto or color or gradient or tiling *Obligatorio* *Posicional*

El color o degradado que se usa para el trazo.

Si se establece en auto, el valor se hereda, con black por defecto.

/4.2 **thickness** auto or length *Obligatorio* *Posicional*

El grosor del trazo.

Si se establece en auto, el valor se hereda, con 1pt por defecto.

/4.3 **cap** auto or str *Obligatorio* *Posicional*

Cómo se renderizan los extremos del trazo.

Si se establece en auto, el valor se hereda, con "butt" por defecto.

| Variante | Detalles |
|---|---|
| "butt" | Terminación de trazo cuadrada con el borde en el punto final del trazo. |
| "round" | Terminación de trazo circular centrada en el punto final del trazo. |
| "square" | Terminación de trazo cuadrada centrada en el punto final del trazo. |

/4.4 **join** auto or str *Obligatorio* *Posicional*

Cómo se renderizan los giros bruscos.

Si se establece en auto, el valor se hereda, con "miter" por defecto.

| Variante | Detalles |
|---|---|
| "miter" | Los segmentos se unen con bordes filosos. Los quiebres filosos que superan el límite de inglete se biselan en su lugar. |
| "round" | Los segmentos se unen con esquinas redondeadas. |
| "bevel" | Los segmentos se unen con un bisel (un borde recto que conecta los extremos de los segmentos unidos). |

/4.5 **dash** none or auto or str or array or dictionary *Obligatorio* *Posicional*

El patrón de guiones que se usa. Puede ser:

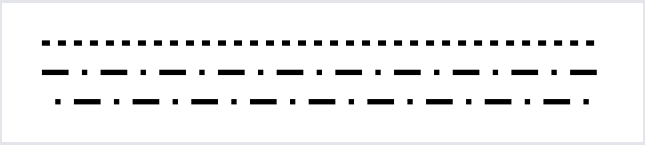
- Uno de los patrones predefinidos que se enumeran abajo.
- none, equivalente al patrón predefinido "solid".
- Un array[3.5.2] con longitudes alternadas de guiones y espacios. También podés usar la cadena "dot" para una longitud igual al grosor de la línea.

– Un diccionario[3.5.12] con las claves `array` (igual que el array de arriba) y `phase` (de tipo longitud[3.10.13]), que define en qué parte del patrón se empieza a dibujar.

Si se establece en `auto`, el valor se hereda, con `none` por defecto.

Example · Uso

```
#set line(length: 100%, stroke: 2pt)
#stack(
  spacing: 1em,
  line(stroke: (dash: "dashed")),
  line(stroke: (dash: (10pt, 5pt, "dot", 5pt))),
  line(stroke: (dash: (array: (10pt, 5pt, "dot", 5pt), phase: 10pt))),
)
```



Example · Patrones predefinidos

```
#set page(width: auto, height: auto)

#let patterns = (
  ("dotted", "dashed", "dash-dotted").map(d => (
    "loosely-" + d,
    d,
    "densely-" + d,
  ))
).flatten()


#let display(dash) = {
  set par(spacing: 0.3em)

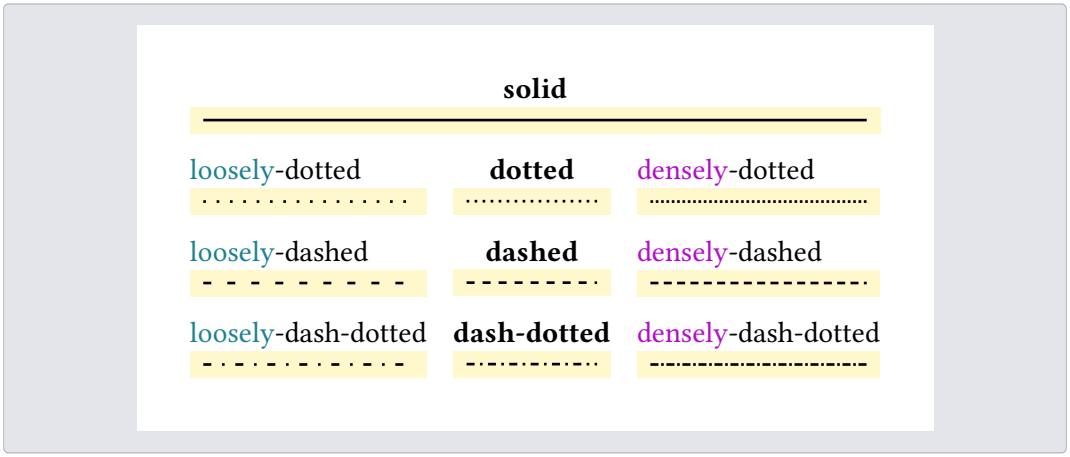
  show "loosely": set text(eastern.darken(17%))
  show "densely": set text(purple)

  let derived = dash.starts-with("loosely-") or dash.starts-with("densely-")
  if derived { dash } else { strong(dash) }

  rect(
    line(stroke: (dash: dash), length: 100%),
    stroke: none,
    fill: yellow.transparentize(80%),
  )
}

#table(
  columns: 3,
  stroke: none,
  align: (left, center, left),

  table.cell(colspan: 3, align: center, display("solid")),
  ..patterns.map(display),
)
```




| Variante | Detalles |
|---|---|
| `"solid"` | `()` |
| `"dotted"` | `("dot", 2pt)` |
| `"densely-dotted"` | `("dot", 1pt)` |
| `"loosely-dotted"` | `("dot", 4pt)` |
| `"dashed"` | `(3pt, 3pt)` |
| `"densely-dashed"` | `(3pt, 2pt)` |
| `"loosely-dashed"` | `(3pt, 6pt)` |
| `"dash-dotted"` | `(3pt, 2pt, "dot", 2pt)` |
| `"densely-dash-dotted"` | `(3pt, 1pt, "dot", 1pt)` |
| `"loosely-dash-dotted"` | `(3pt, 4pt, "dot", 4pt)` |

### /4.6 `miter-limit` `auto` or `float` *Obligatorio* *Posicional*

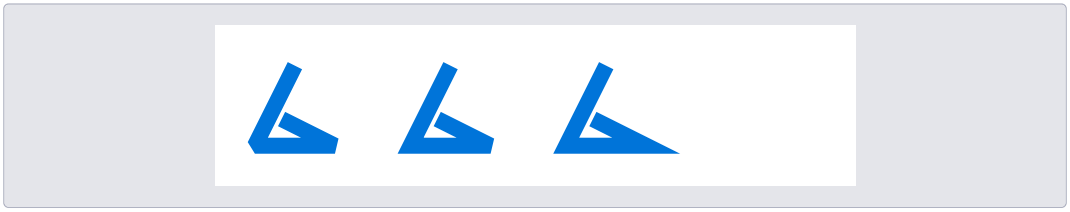
Número a partir del cual las curvas bruscas que sobresalen se renderizan con un bisel en lugar de una unión en inglete. Cuanto mayor sea el número, más agudo puede ser un ángulo antes de que se bisele. Solo es aplicable si `join` es `"miter"`.

Específicamente, el límite de inglete es la relación máxima entre la longitud de la protrusión de la esquina y el grosor del trazo.

Si se establece en `auto`, el valor se hereda, con `4.0` por defecto.

```
#let items = (
  curve.move((15pt, 0pt)),
  curve.line((0pt, 30pt)),
  curve.line((30pt, 30pt)),
  curve.line((10pt, 20pt)),
)

#set curve(stroke: 6pt + blue)
#stack(
  dir: ltr,
  spacing: 1cm,
  curve(stroke: (miter-limit: 1), ..items),
  curve(stroke: (miter-limit: 4), ..items),
  curve(stroke: (miter-limit: 5), ..items),
)
```



### 3.11.13 `tiling` Desde: 0.13.0

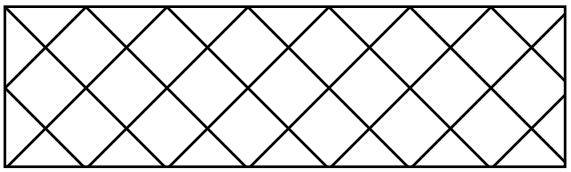
Un relleno de teselado repetido.

Typst admite el tipo de teselado más común, en el que un patrón se repite en forma de grilla, cubriendo toda el área de un elemento que se rellena o traza. El patrón se define por un `size`[3.11.13/3.1] de tesela y un cuerpo que define el contenido de cada celda. También podés agregar `spacing`[3.11.13/3.2] horizontal o vertical entre las celdas del teselado. El `offset`[3.11.13/3.3] y el `angle`[3.11.13/3.4] determinan la ubicación del teselado.

#### /1 Ejemplo

```
#let pat = tiling(size: (30pt, 30pt), {
  place(line(start: (0%, 0%), end: (100%, 100%)))
  place(line(start: (0%, 100%), end: (100%, 0%)))
})

#rect(fill: pat, width: 100%, height: 60pt, stroke: 1pt)
```



#### /2 Teselados en texto

Los teselados también se admiten en texto, pero solo cuando se establece `relative`[3.11.13/3.5] en `auto` (el valor por defecto) o en `"parent"`. Para crear teselados palabra por palabra o glifo por glifo, podés envolver manualmente las palabras o los caracteres de tu texto en `boxes`[3.10.5] o mediante una `regla show`[3.2.2].

```
#let pat = tiling(
  size: (30pt, 30pt),
  relative: "parent",
  square(
    size: 30pt,
    fill: gradient
      .conic(..color.map.rainbow),
  )
)

#set text(fill: pat)
#lorem(10)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

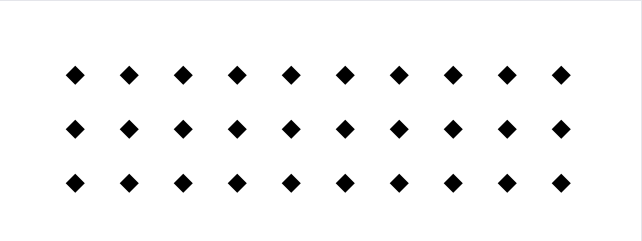
#### /3 Constructor

Construye un teselado nuevo.

```
#let pat = tiling(
  size: (20pt, 20pt),
  relative: "parent",
  place(
    dx: 5pt,
    dy: 5pt,
    rotate(45deg, square(
      size: 5pt,
      fill: black,
```

```
    )),
  ),
)

#rect(width: 100%, height: 60pt, fill: pat)
```



```
tiling(
  size: auto array,
  spacing: array,
  offset: array,
  angle: angle,
  relative: auto str,
  content,
) → tiling
```

/3.1 **size** auto or array — Por defecto: auto

El cuadro delimitador de cada celda del teselado, especificado como un par (x, y).

Si se establece en auto, el teselado toma el tamaño del contenido dispuesto.

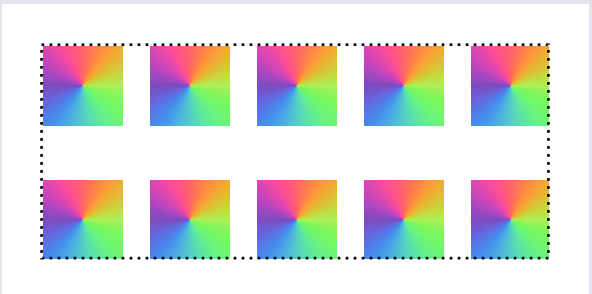
/3.2 **spacing** array — Por defecto: (0pt, 0pt)

El espaciado entre las celdas del teselado, especificado como un par (x, y).

Si el espaciado es menor que el tamaño del teselado, el teselado se superpondrá consigo mismo. Si es mayor, el teselado tendrá huecos.

```
#let pat = tiling(
  size: (30pt, 30pt),
  spacing: (10pt, 20pt),
  square(size: 30pt, fill: gradient.conic(..color.map.rainbow)),
)

#rect(
  width: 100%,
  height: 80pt,
  fill: pat,
  stroke: (thickness: 1pt, dash: "dotted"),
)
```



/3.3 **offset** array — Por defecto: (0% + 0pt, 0% + 0pt)

Desplaza toda la grilla de teselas sin afectar el tamaño ni el espaciado de las teselas.

El desplazamiento se especifica como un par (x, y). Los valores positivos de x mueven el patrón hacia la derecha y los valores positivos de y lo mueven hacia abajo. Los valores relativos se resuelven respecto del tamaño de la tesela más el espaciado.

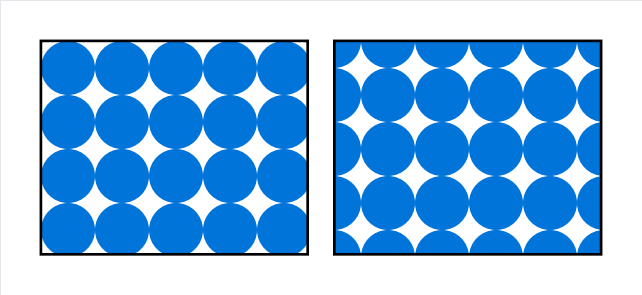
Tené en cuenta que el desplazamiento causado por el offset afecta a las teselas en sí, mientras que el desplazamiento del contenido interior (por ejemplo, mediante place(dx: .., dy: ..)) puede causar recortes cuando el contenido se mueve fuera del cuadro delimitador de la tesela.

```
#set rect(width: 100%, height: 80pt, stroke: 1pt)

#let pat = tiling(
  size: (20pt, 20pt),
  circle(radius: 10pt, fill: blue),
)

#let pat-with-offset = tiling(
  size: (20pt, 20pt),
  offset: (50%, 50%),
  circle(radius: 10pt, fill: blue),
)

#grid(
  columns: 2,
  column-gutter: 10pt,
  rect(fill: pat),
  rect(fill: pat-with-offset),
)
```

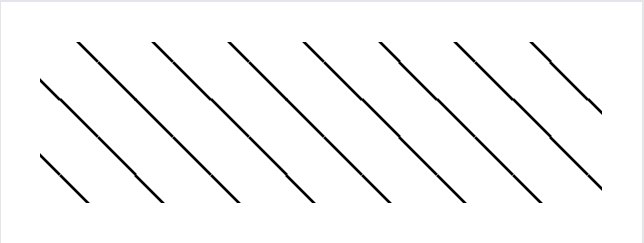


/3.4 **angle** angle — Por defecto: 0deg

Rota las teselas y la grilla en sentido horario alrededor de las coordenadas del desplazamiento según el ángulo especificado. La rotación se aplica después del desplazamiento.

```
#let pat = tiling(
  size: (20pt, 20pt),
  angle: 45deg,
  line(start: (0%, 50%), end: (100%, 50%)),
)

#rect(width: 100%, height: 60pt, fill: pat)
```



/3.5 **relative** auto or str — Por defecto: auto

Determina respecto del cuadro delimitador de qué elemento se dibuja el teselado.

Por defecto, los teselados se dibujan relativos a la forma sobre la que se pintan ("self"), salvo que el teselado se aplique sobre texto, en cuyo caso son relativos al contenedor ancestro más cercano ("parent").

El padre de un elemento es el box[3.10.5] o block[3.10.4] más interno que contiene el elemento o, si no hay ninguno, la propia página.

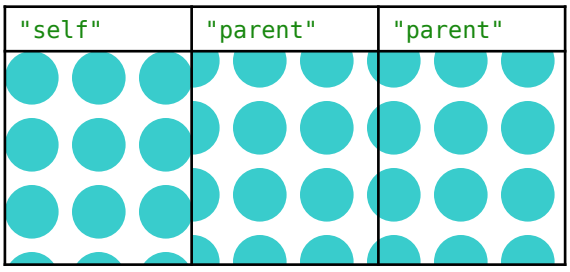
```
#let pat = tiling(
  size: (20pt, 20pt),
  spacing: (5pt, 5pt),
  relative: "self",
  circle(radius: 10pt, fill: teal),
)

#let pat-with-parent = tiling(
  size: (20pt, 20pt),
  spacing: (5pt, 5pt),
  relative: "parent",
  circle(radius: 10pt, fill: teal),
)

#set raw(lang: "typc")
#table(
  columns: (1fr, 1fr, 1fr),
  rows: (auto, 80pt),
  table.header(`"self"`, `"parent"`, `"parent"`),

  // This one is local to the cell itself.
  table.cell(fill: pat, none),

  // These two are both page-relative, so the
  // pattern is continuous.
  table.cell(fill: pat-with-parent, none),
  table.cell(fill: pat-with-parent, none),
)
```



| Variante | Detalles |
| --- | --- |
| `"self"` | Relativo a sí mismo (su propio cuadro delimitador). |
| `"parent"` | Relativo a su elemento padre (el cuadro delimitador del padre). |

/3.6 **body** `content` *Obligatorio* *Posicional*

El contenido de cada celda del teselado.

# 3.12 Introspección

Interacciones entre partes del documento.

Esta categoría es el hogar de las capacidades de introspección de Typst: con la función `counter`, podés acceder a los contadores de páginas, secciones, figuras y ecuaciones y manipularlos, o crear contadores personalizados. Por su parte, la función `query` te permite buscar elementos en el documento para construir cosas como una lista de figuras o encabezados que muestran el título del capítulo actual.

La mayoría de las funciones son *contextuales*. Se recomienda leer el capítulo sobre contexto[3.4] antes de continuar acá.

## Definiciones


3.12.1 Counter Counts through pages, elements, and more. . . . . 484
3.12.2 Here Provides the current location in the document. . . . . 489
3.12.3 Locate Determines the location of an element in the document. . . . . 490
3.12.4 Location Identifies an element in the document. . . . . 491
3.12.5 Metadata Exposes a value to the query system without producing visible content. . . . . 493
3.12.6 Query Finds elements in the document. . . . . 494
3.12.7 State Manages stateful parts of your document. . . . . 498

### 3.12.1 `counter` Desde: siempre

Cuenta a través de páginas, elementos y más.

Con la función counter, podés acceder a los contadores de páginas, títulos, figuras y más, y modificarlos. Además, podés definir contadores personalizados para otras cosas que quieras contar.

Como los contadores cambian a lo largo del documento, su valor actual es *contextual*. Se recomienda leer el capítulo sobre contexto[3.4] antes de continuar acá.

#### /1 Acceder a un contador

Para acceder al valor sin procesar de un contador, podemos usar la función `get`[3.12.1/11.1]. Esta función devuelve un array[3.5.2]: los contadores pueden tener varios niveles (en el caso de los títulos, para secciones, subsecciones, etcétera) y cada elemento del array corresponde a un nivel.

```
#set heading(numbering: "1.")

= Introduction
Raw value of heading counter is
#context counter(heading).get()
```

**1. Introduction**
Raw value of heading counter is `(1,)`

#### /2 Mostrar un contador

A menudo queremos mostrar el valor de un contador de una manera más legible para las personas. Para eso, podemos llamar a la función `display`[3.12.1/11.2] sobre el contador. Esta función obtiene el valor actual del contador y lo formatea con una numeración[3.6.12] proporcionada o inferida automáticamente.

```
#set heading(numbering: "1.")

= Introduction
Some text here.

= Background
The current value is: #context {
  counter(heading).display()
}

Or in roman numerals: #context {
  counter(heading).display("I")
}
```

**1. Introduction**
Some text here.

**2. Background**
The current value is: 2.

Or in roman numerals: II

#### /3 Modificar un contador

Para modificar un contador, podés usar los métodos `step` y `update`:

- El método `step` aumenta en uno el valor del contador. Como los contadores pueden tener varios niveles, opcionalmente toma un argumento `level`. Si se indica, el contador avanza en la profundidad dada.
- El método `update` te permite modificar el contador de forma arbitraria. En su forma básica, le das un entero (o un array para varios niveles). Para tener más flexibilidad, también podés darle una función que reciba el valor actual y devuelva un valor nuevo.

El contador de títulos avanza antes de que se muestre el título, así que `Analysis` recibe el número siete aunque el contador esté en seis después de la segunda actualización.

```
#set heading(numbering: "1.")

= Introduction
#counter(heading).step()

= Background
#counter(heading).update(3)
#counter(heading).update(n => n * 2)

= Analysis
Let's skip 7.1.
#counter(heading).step(level: 2)

== Analysis
Still at #context {
  counter(heading).display()
}
```

1. Introduction

3. Background

7. Analysis

Let's skip 7.1.

7.2. Analysis

Still at 7.2.

/4 ## Contadores de elementos

Arriba hay varios ejemplos del uso del contador de heading[3.6.9]. Los títulos son solo un tipo de elemento que se puede contar. En general, los contadores pueden contar cualquier tipo de elemento *localizable*[3.12.4/1].

Además, un contador también puede contar solo los elementos que coinciden con un selector[3.5.26] específico. Por ejemplo, `counter(figure.where(kind: image))` cuenta las figuras que contienen imágenes, pero ignora otros tipos de figuras.

/5 ## Contador de páginas

El contador de páginas es especial. Avanza automáticamente en cada salto de página. Pero, como los demás contadores, también podés hacerlo avanzar manualmente. Por ejemplo, podrías tener números de página romanos para tu prefacio, luego pasar a números de página arábigos para tu contenido principal y reiniciar el contador de páginas en uno.

```
#set page(numbering: "(i)")

= Preface
The preface is numbered with
roman numerals.

#set page(numbering: "1 / 1")
#counter(page).update(1)

= Main text
Here, the counter is reset to one.
We also display both the current
page and total number of pages in
Arabic numbers.
```

Preface

The preface is numbered with roman numerals.

(i)

Main text

Here, the counter is reset to one. We also display both the current page and total number of pages in Arabic numbers.

1 / 1

/6 ## Contadores personalizados

Para definir tu propio contador, llamá a la función `counter` con una cadena de texto como clave. Esta clave identifica el contador de forma global.

```
#let mine = counter("mycounter")
#context mine.display() \
#mine.step()
#context mine.display() \
#mine.update(c => c * 3)
#context mine.display()
```

0
1
3

## /7 Cómo avanzar

Cuando definís y usás un contador personalizado, en general, primero tenés que avanzar el contador y después mostrarlo. De esta manera, el comportamiento de avance de un contador puede depender del elemento para el que se avanza. Si estuvieras escribiendo un contador para, digamos, teoremas, la definición de tu teorema incluiría primero el avance del contador y solo después mostraría el contador y el contenido del teorema.

```
#let c = counter("theorem")
#let theorem(it) = block[
  #c.step()
  *Theorem #context c.display():*
  #it
]

#theorem[$1 = 1$]
#theorem[$2 < 3$]
```

**Theorem 1:** $1 = 1$

**Theorem 2:** $2 < 3$

La lógica detrás de esto se explica mejor con el ejemplo del contador de títulos: una actualización del contador de títulos depende del nivel del título. Al avanzar justo antes del título, podemos pasar correctamente de `1` a `1.1` al encontrar un título de nivel 2. Si avanzáramos después del título, no sabríamos a qué valor avanzar.

Como los contadores siempre deben avanzar antes de los elementos que cuentan, siempre empiezan en cero. De esta manera, están en uno en la primera visualización (que ocurre después del primer avance).

## /8 Viaje en el tiempo

¡Los contadores pueden viajar en el tiempo! Podés averiguar el valor final del contador antes de que se alcance e incluso determinar cuál era el valor en cualquier ubicación particular del documento.

```
#let mine = counter("mycounter")

= Values
#context [
  Value here: #mine.get() \
  At intro: #mine.at(<intro>) \
  Final value: #mine.final()
]

#mine.update(n => n + 3)

= Introduction <intro>
#lorem(10)

#mine.step()
#mine.step()
```

### Values

Value here: `(0,)`
At intro: `(3,)`
Final value: `(5,)`

### Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

## /9 Otros tipos de estado

El tipo `counter` está estrechamente relacionado con el tipo state[3.12.7]. Leé su documentación para más detalles sobre la gestión de estado en Typst y por qué no usa simplemente variables normales para los contadores.

## /10 Constructor

Crea un contador nuevo identificado por una clave.

```
counter( str  label  selector  location  function ) → counter
```

### /10.1 key  str or label or selector or location or function  *Obligatorio*  *Posicional*

La clave que identifica este contador de forma global.

- Si es una cadena de texto, crea un contador personalizado que solo se ve afectado por actualizaciones manuales,
- Si es la función page[3.10.17], cuenta a través de las páginas,
- Si es un selector[3.5.26], cuenta a través de los elementos que coinciden con el selector. Por ejemplo, podés
  - proporcionar una función de elemento para contar elementos de ese tipo,
  - proporcionar un selector where[3.5.16/8.2] para contar un tipo de elemento con campos específicos,
  - proporcionar una <label>[3.5.18] para contar los elementos con esa etiqueta.

  Las claves de contador de tipo elemento y selector where[3.5.16/8.2] solo se admiten para elementos localizables[3.12.4/1].

## /11 Funciones

### /11.1 get  Contextual  Desde: 0.11.0

Obtiene el valor del contador en la ubicación actual. Siempre devuelve un array de enteros, incluso si el contador tiene un solo número.

Esto equivale a `counter.at(here())`.

```
self.get() → int  array
```

### /11.2 display  Contextual  Desde: siempre

Muestra el valor del contador.

Podés proporcionar tanto una numeración personalizada como una ubicación personalizada. Ambas son auto por defecto, lo que selecciona valores por defecto razonables (la numeración del elemento contado y la ubicación actual, respectivamente).

Devuelve la salida formateada.

```
self.display(
  auto  str  function ,
  at:  auto  label  selector  location  function ,
  both:  bool ,
) → any
```

#### /11.2.1 numbering  auto or str or function  *Posicional*  Por defecto: auto

Una función o patrón de numeración[3.6.12] que especifica cómo mostrar el contador. Si se da una función, esa función recibe cada número del contador como un argumento separado. Si la cantidad de números varía, por ejemplo para el argumento heading, podés usar un sumidero de argumentos[3.5.1].

Si se omite o se establece en auto, muestra el contador con el estilo de numeración del elemento contado o con el patrón "1.1" si no existe tal estilo.

#### /11.2.2 at  auto or label or selector or location or function  Por defecto: auto

El lugar en el que se debe mostrar el contador.

Si se usa un selector, debe coincidir exactamente con un elemento del documento. Los tipos de selectores más útiles para esto son las etiquetas[3.5.18] y las ubicaciones[3.12.4].

Si se omite o se establece en auto, muestra el contador en la ubicación actual. Esto equivale a usar here()[3.12.2].

La numeración se ejecutará con un contexto en el que here() se resuelve en la ubicación indicada, de modo que las numeraciones que involucran otros contadores se resuelvan correctamente.

/11.2.3 **both** bool — Por defecto: false

Si está activado, muestra juntos el conteo actual y el final del nivel superior. Ambos se pueden estilizar con un único patrón de numeración. La propiedad de numeración de páginas lo usa para mostrar el número actual y el total de páginas cuando se da un patrón como "1 / 1".

/11.3 **at** Contextual Desde: siempre

Obtiene el valor del contador en la ubicación dada. Siempre devuelve un array de enteros, incluso si el contador tiene un solo número.

El selector debe coincidir exactamente con un elemento del documento. Los tipos de selectores más útiles para esto son las etiquetas[3.5.18] y las ubicaciones[3.12.4].

```
self.at(label selector location function) → int array
```

/11.3.1 **selector** label or selector or location or function *Obligatorio* *Posicional*

El lugar en el que se debe obtener el valor del contador.

/11.4 **final** Contextual Desde: siempre

Obtiene el valor del contador al final del documento. Siempre devuelve un array de enteros, incluso si el contador tiene un solo número.

```
self.final() → int array
```

/11.5 **step** Desde: siempre

Aumenta en uno el valor del contador.

La actualización tendrá efecto en la posición donde se inserte en el documento el contenido devuelto. ¡Si no ponés la salida en el documento, no pasa nada! Este sería el caso, por ejemplo, si escribís let _ = counter(page).step(). Las actualizaciones de contadores siempre se aplican en el orden del diseño y, en ese caso, Typst no sabría cuándo avanzar el contador.

```
self.step(level: int) → content
```

/11.5.1 **level** int — Por defecto: 1

La profundidad a la que avanza el contador. Por defecto es 1.

/11.6 **update** Desde: siempre

Actualiza el valor del contador.

Igual que con step, la actualización solo ocurre si ponés el contenido resultante en el documento.

```
self.update(int array function) → content
```

/11.6.1 **update** int or array or function *Obligatorio* *Posicional*

Si se da un entero o un array de enteros, establece el contador en ese valor. Si se da una función, esa función recibe el valor anterior del contador (con cada número como un argumento separado) y tiene que devolver el valor nuevo (entero o array).

### 3.12.2 `here` Contextual Desde: 0.11.0

Proporciona la ubicación actual en el documento.

Podés pensar en `here` como una pieza fundamental de bajo nivel que extrae directamente la ubicación actual del contexto[3.4] activo. Algunas otras funciones la usan internamente: por ejemplo, `counter.get()` equivale a `counter.at(here())`.

Dentro de reglas show sobre elementos localizables[3.12.4/1], `here()` coincidirá con la ubicación del elemento mostrado.

Si querés mostrar el número de la página actual, consultá la documentación del tipo `counter`[3.12.1]. Si bien `here` se puede usar para determinar el número de página físico, normalmente querés el número de página lógico, que puede, por ejemplo, haberse reiniciado después de un prefacio.

#### /1 Ejemplos

Determinar la posición actual en el documento en combinación con el método `position`[3.12.4/2.3]:

```
#context [
  I am located at
  #here().position()
]
```

I am located at (page: 1, x: 15pt, y: 22.24pt)

Ejecutar una consulta[3.12.6] de elementos anteriores a la posición actual:

```
= Introduction
= Background

There are
#context query(
  selector(heading).before(here())
).len()
headings before me.

= Conclusion
```

**Introduction**

**Background**

There are 2 headings before me.

**Conclusion**

Consultá el tipo `selector`[3.5.26] para más detalles sobre los selectores before/after.

#### /2 Parámetros

```
here() → location
```

### 3.12.3 `locate` Contextual Desde: siempre

Determina la ubicación de un elemento en el documento.

Toma un selector que debe coincidir exactamente con un elemento y devuelve la `location`[3.12.4] de ese elemento. Esta ubicación se puede usar, en particular, para obtener el número de `page`[3.12.4/2.1] físico y la `position`[3.12.4/2.3] (página, x, y) de ese elemento.

#### /1 Ejemplos

Ubicar un elemento específico:

```
#context [
  Introduction is at: \
  #locate(<intro>).position()
]

= Introduction <intro>
```

Introduction is at:
(page: 1, x: 15pt, y: 55.87pt)

**Introduction**

#### /2 Parámetros

```
locate( label  selector  location  function ) → location
```

##### /2.1 `selector` `label` or `selector` or `location` or `function` *Obligatorio* *Posicional*

Un selector que debe coincidir exactamente con un elemento. Se ubicará ese elemento.

Es especialmente útil en combinación con
- `here`[3.12.2], para ubicar el contexto actual,
- una `location`[3.12.4] obtenida de algún elemento consultado mediante el método `location()`[3.5.8/3.5] del contenido.

### 3.12.4 `location` Desde: siempre

Identifica un elemento en el documento.

Una ubicación identifica de forma única un elemento en el documento y te permite acceder a su posición absoluta en las páginas. Podés obtener la ubicación actual con la función here[3.12.2] y la ubicación de un elemento consultado o mostrado con el método `location()`[3.5.8/3.5] del contenido.

## /1 Elementos localizables

Los elementos a los que se les asigna una ubicación automáticamente se llaman *localizables* y se pueden encontrar con consultas[3.12.6]:

- En la categoría Modelo[3.6], los siguientes elementos son localizables: `asset`[3.14.2/5], `bibliography`[3.6.1], `cite`[3.6.3], `document`[3.6.5], `emph`[3.6.6], `enum`[3.6.11], `figure`[3.6.7], `figure.caption`[3.6.7/6.1], `footnote`[3.6.8], `footnote.entry`[3.6.8/4.1], `heading`[3.6.9], `link`[3.6.10], `list`[3.6.2], `outline`[3.6.13], `outline.entry`[3.6.13/5.1], `par`[3.6.14], `quote`[3.6.16], `ref`[3.6.17], `strong`[3.6.18], `table`[3.6.19], `terms`[3.6.20] y `title`[3.6.21]. La mayoría de los elementos de la categoría *Modelo* son localizables porque los elementos semánticos, como los títulos y las figuras, se suelen usar con introspección.
- En la categoría Texto[3.7], el elemento `raw`[3.7.6] y los elementos de decoración `underline`[3.7.13], `overline`[3.7.5], `strike`[3.7.9] y `highlight`[3.7.1] son localizables, ya que también son bastante semánticos por naturaleza.
- En la categoría Introspección[3.12], el elemento `metadata`[3.12.5] es localizable, ya que su propósito principal es ser consultado.
- En las demás categorías, la mayoría de los elementos no son localizables. Las excepciones son `math.equation`[3.8.16], `image`[3.11.7] y `pdf.attach`[3.14.4/5].

Para averiguar si un elemento específico es localizable, podés intentar consultarlo con `query`[3.12.6].

Tené en cuenta que igualmente podés observar en las consultas elementos que no son localizables por otros medios, por ejemplo, cuando tienen una etiqueta adjunta.

## /2 Funciones

### /2.1 `page` Desde: siempre

Devuelve el número de página de esta ubicación.

Tené en cuenta que esto no devuelve el valor del contador de páginas[3.12.1] en esta ubicación, sino el número de página real (empezando desde uno). Si querés saber el valor del contador de páginas, usá `counter(page).at(loc)` en su lugar.

En un documento HTML, Typst no puede saber dónde terminará el contenido, por lo que esta función devuelve none.

Este método se puede usar en combinación con here[3.12.2] para obtener la posición física de página del contexto actual:

```
#context [
  I am located on
  page #here().page()
]
```

I am located on page 1

```
self.page() → none int
```

### /2.2 `page-numbering` Desde: siempre

Devuelve el patrón de numeración de la página de esta ubicación. Se puede usar al mostrar el contador de páginas para obtener la numeración local. Es útil si estás construyendo índices o tablas de contenidos personalizados.

Si la numeración de páginas está establecida en none en esa ubicación, esta función devuelve none.

```
self.page-numbering() → none str function
```

### /2.3 position Desde: siempre

Devuelve un diccionario con el número de página y la posición x, y de esta ubicación. El número de página empieza en uno y las coordenadas se miden desde la esquina superior izquierda de la página.

Si solo necesitás el número de página, usá page()[3.12.4/2.1], ya que le permite a Typst omitir trabajo innecesario.

En un documento HTML, Typst no puede saber dónde terminará el contenido, por lo que esta función devuelve none. El contenido de un html.frame[3.14.3/5] es una excepción, ya que se dispone con el motor de diseño de Typst. Para las ubicaciones dentro de esos marcos, esta función devuelve un diccionario solo con las coordenadas x e y dentro del marco, pero sin la clave page.

```
self.position() → none dictionary
```

### 3.12.5 `metadata` Elemento Desde: 0.7.0

Expone un valor al sistema de consultas sin producir contenido visible.

Este elemento se puede obtener con la función `query`[3.12.6] y, desde la línea de comandos, con `typst query`[3.12.6/3]. Su propósito es exponer un valor arbitrario al sistema de introspección. Para identificar un valor de metadatos entre otros, podés adjuntarle una `label`[3.5.18] y consultar por esa etiqueta.

El elemento `metadata` es especialmente útil para las consultas desde la línea de comandos, porque te permite exponer valores arbitrarios al mundo exterior.

```
// Put metadata somewhere.
#metadata("This is a note") <note>

// And find it from anywhere else.
#context {
  query(<note>).first().value
}
```

This is a note

#### /1 Parámetros

```
metadata( any ) → content
```

##### /1.1 `value` any *Obligatorio* *Posicional*

El valor que se incrusta en el documento.

### 3.12.6 `query` Contextual Desde: siempre

Encuentra elementos en el documento.

La función `query` te permite buscar en tu documento elementos de un tipo particular o con una etiqueta particular. Para usarla, primero tenés que asegurarte de que haya contexto[3.4] disponible.

#### /1 Encontrar elementos

En el siguiente ejemplo, creamos manualmente un índice de contenidos en lugar de usar la función `outline`[3.6.13].

Para hacerlo, primero consultamos todos los títulos del documento de nivel 1 y donde `outlined` sea verdadero. Consultar solo los títulos de nivel 1 asegura que, a los fines de este ejemplo, los subtítulos no se incluyan en el índice de contenidos. El campo `outlined` se usa para excluir el propio título "Table of Contents".

Fijate que abrimos un `context` para poder usar la función `query`.

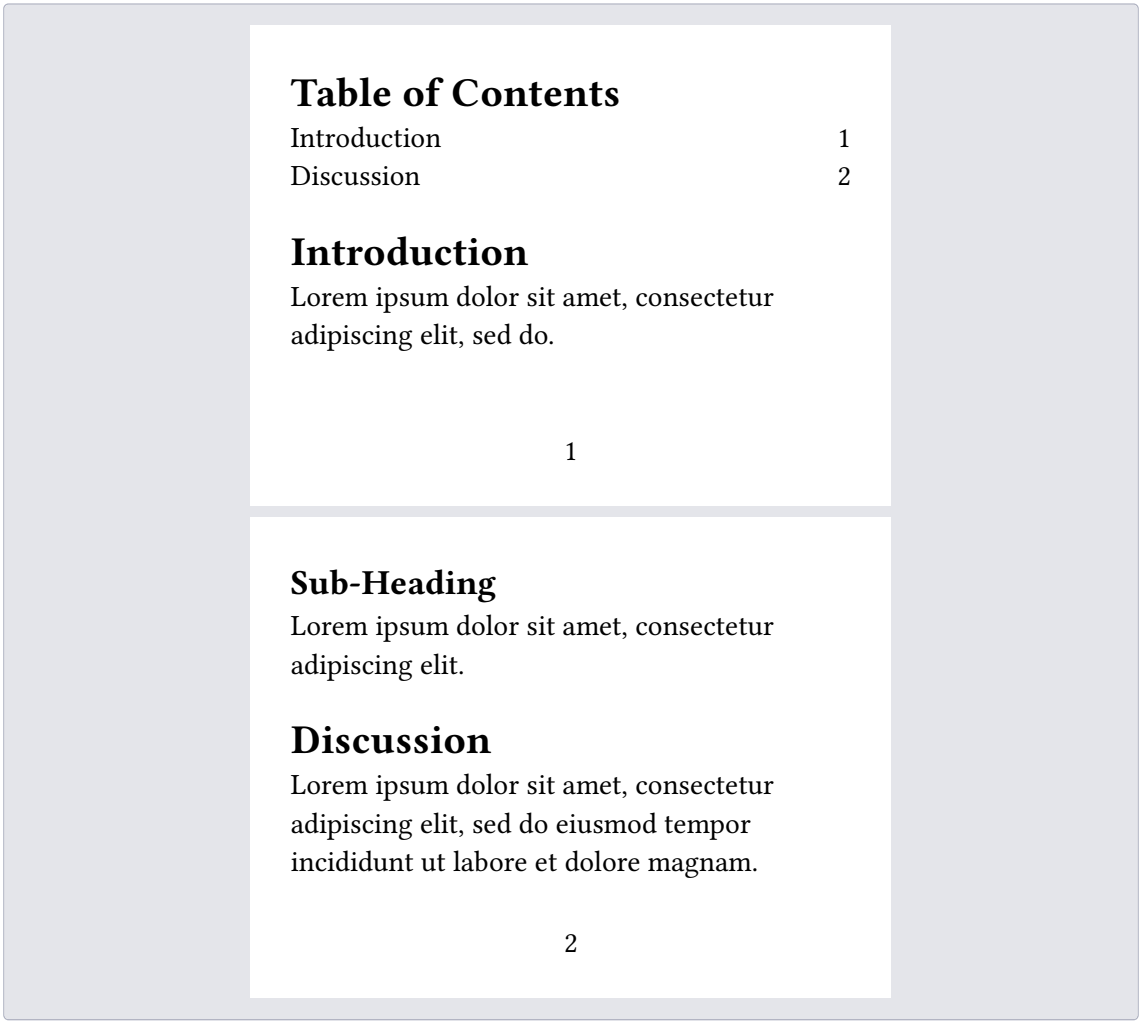
```
#set page(numbering: "1")

#heading(outlined: false)[
  Table of Contents
]
#context {
  let chapters = query(
    heading.where(
      level: 1,
      outlined: true,
    )
  )
  for chapter in chapters {
    let loc = chapter.location()
    let nr = counter(page).display(at: loc)
    [#chapter.body #h(1fr) #nr \ ]
  }
}

= Introduction
#lorem(10)
#pagebreak()

== Sub-Heading
#lorem(8)

= Discussion
#lorem(18)
```




Para obtener los números de página, primero obtenemos la ubicación de los elementos devueltos por `query` con `location`[3.5.8/3.5]. Luego también obtenemos la numeración de páginas[3.12.4/2.2] y el contador de páginas[3.12.1/5] en esa ubicación y aplicamos la numeración al contador.

/2 ## Una advertencia

Para resolver todas tus consultas, Typst evalúa y dispone partes del documento varias veces. Sin embargo, no hay garantía de que tus consultas se puedan resolver por completo. Si no tenés cuidado, una consulta puede afectarse a sí misma, lo que lleva a un resultado que nunca se estabiliza.

En el siguiente ejemplo, consultamos todos los títulos del documento. Luego generamos esa misma cantidad de títulos. Al principio, hay un solo título, llamado `Real`. Por lo tanto, `count` es `1` y se genera un título `Fake`. Typst ve que el resultado de la consulta cambió y la procesa de nuevo. Esta vez, `count` es `2` y se generan dos títulos `Fake`. Esto sigue y sigue. Como podemos ver, la salida tiene una cantidad finita de títulos. Esto se debe a que Typst simplemente se rinde después de unos cuantos intentos.

En general, deberías intentar no escribir consultas que se afecten a sí mismas. Las mismas advertencias también se aplican a otras funciones de introspección como los contadores[3.12.1] y el estado[3.12.7].

```
= Real
#context {
  let elems = query(heading)
  let count = elems.len()
  count * [= Fake]
}
```

**Real**

**Fake**

**Fake**

**Fake**

**Fake**

## /3 Consultas desde la línea de comandos

También podés realizar consultas desde la línea de comandos con el comando `typst eval`. Este comando evalúa código Typst, potencialmente en el contexto de un documento, y muestra el valor resultante en forma serializada. Toma como primer argumento el código que se evalúa y, opcionalmente, la ruta a un documento mediante `--in`.

Considerá el siguiente archivo `example.typ`, que contiene algunos metadatos[3.12.5] invisibles:

```
#metadata("This is a note") <note>
```

Podés ejecutar una consulta sobre él de la siguiente manera con la CLI de Typst.

```
$ typst eval 'query(<note>)' --in example.typ
[
  {
    "func": "metadata",
    "value": "This is a note",
    "label": "<note>"
  }
]
```

Este comando le indica a Typst que compile `example.typ` y luego ejecute el código `query(<note>)` con acceso al documento resultante.

**Nota:** El código está rodeado de comillas para evitar que el shell interprete caracteres especiales. La forma de entrecomillar cadenas depende de tu plataforma/shell.

### /3.1 Obtener un campo específico

Con frecuencia, te interesa solo un campo específico de los elementos resultantes. En el caso del elemento `metadata`, el campo interesante es `value`. Podés extraer solo este campo ajustando el código.

```
$ typst eval 'query(<note>).map(it => it.value)' --in example.typ
["This is a note"]
```

Si te interesa un único elemento, también podés usar el método `first()`[3.5.2/3.2] para extraer solo ese.

```
$ typst eval 'query(<note>).first().value' --in example.typ
"This is a note"
```

### /3.2 Consultar para un destino de exportación específico

Si necesitás consultar un documento al exportar para un destino específico, podés usar el argumento `--target`. Los valores válidos son `paged` y `html` (si la característica `html`[3.14.3] está habilitada).

## /4 Parámetros

```
query(label selector location function) → array
```

### /4.1 target `label` or `selector` or `location` or `function` *Obligatorio* *Posicional*

Puede ser
- una función de elemento como `heading` o `figure`,
- una `<label>`,
- un selector más complejo como `heading.where(level: 1)`,
- o `selector(heading).before(here())`.

Solo se admiten funciones de elemento localizables[3.12.4/1].

### 3.12.7 `state` Desde: siempre

Administra las partes con estado de tu documento.

Supongamos que tenés algunos cálculos en tu documento y querés recordar el resultado de tu último cálculo para usarlo en el siguiente. Podrías probar algo similar al código de abajo y esperar que muestre 10, 13, 26 y 21. Sin embargo, esto **no funciona** en Typst. Si probás este código, vas a ver que Typst se queja con el siguiente mensaje de error: *Variables from outside the function are read-only and cannot be modified.*

```
// This doesn't work!
#let star = 0
#let compute(expr) = {
  star = eval(
    expr.replace("⭐", str(star))
  )
  [New value is #star.]
}

#compute("10") \
#compute("⭐ + 3") \
#compute("⭐ * 2") \
#compute("⭐ - 5")
```

#### /1 Estado y marcado del documento

¿Por qué hace eso? Porque, en general, este tipo de cálculo con efectos secundarios es problemático en el marcado de un documento y Typst es claro al respecto. Para que los resultados tengan sentido, el cálculo debe avanzar en el mismo orden en que se van a disponer los resultados en el documento. En nuestro ejemplo simple, ese es el caso, pero en general podría no serlo.

Veamos un tipo de estado ligeramente distinto, pero similar: la numeración de títulos. Queremos aumentar el contador de títulos en cada título. Bastante fácil, ¿no? Solo hay que sumar uno. Bueno, no es tan simple. Consideremos el siguiente ejemplo:

```
#set heading(numbering: "1.")
#let template(body) = [
  = Outline
  ...
  #body
]

#show: template

= Introduction
...
```

1. Outline

...

2. Introduction

...

Acá, Typst primero procesa el cuerpo del documento después de la regla show, ve el título `Introduction`, luego le pasa el contenido resultante a la función `template` y recién entonces ve `Outline`. Simplemente contar hacia arriba numeraría `Introduction` con 1 y `Outline` con 2.

#### /2 Gestionar estado en Typst

Entonces, ¿qué hacemos en cambio? Usamos el sistema de gestión de estado de Typst. Llamar a la función `state` con una clave de tipo cadena de texto que lo identifique y un valor inicial opcional te da un valor de estado que expone algunas funciones. Las dos más importantes son `get` y `update`:

– La función get[3.12.7/6.1] obtiene el valor actual del estado. Como el valor puede variar a lo largo del documento, es una función *contextual* que solo se puede usar cuando hay contexto[3.4] disponible.

– La función update[3.12.7/6.4] modifica el estado. Podés darle cualquier valor. Si se le da un valor que no es una función, establece el estado en ese valor. Si se le da una función, esa función recibe el estado anterior y tiene que devolver el estado nuevo.

Nuestro ejemplo inicial ahora se vería así:

```
#let star = state("star", 0)
#let compute(expr) = {
  star.update(old =>
    eval(expr.replace("⭐", str(old)))
  )
  [New value is #context star.get().]
}

#compute("10") \
#compute("⭐ + 3") \
#compute("⭐ * 2") \
#compute("⭐ - 5")
```

New value is 10.
New value is 13.
New value is 26.
New value is 21.

El estado administrado por Typst siempre se actualiza en el orden del diseño, no en el orden de evaluación. El método `update` devuelve contenido y su efecto ocurre en la posición donde el contenido devuelto se inserta en el documento.

Como resultado, ahora también podemos guardar algunos de los cálculos en variables y aun así mostrar los resultados correctos:

```
...

#let more = [
  #compute("⭐ * 2") \
  #compute("⭐ - 5")
]

#compute("10") \
#compute("⭐ + 3") \
#more
```

New value is 10.
New value is 13.
New value is 26.
New value is 21.

Por supuesto, este ejemplo es un poco tonto, pero en la práctica ¡a menudo es exactamente lo que querés! Un buen ejemplo son los contadores de títulos, y por eso el sistema de conteo[3.12.1] de Typst es muy similar a su sistema de estado.

## /3 Viaje en el tiempo

¡Al usar el sistema de gestión de estado de Typst también obtenés capacidades de viaje en el tiempo! Podemos averiguar cuál será el valor del estado en cualquier posición del documento desde cualquier otro lugar. En particular, el método `at` nos da el valor del estado en cualquier ubicación particular y el método `final` nos da el valor del estado al final del documento.

```
...

Value at `<here>` is
#context star.at(<here>)

#compute("10") \
#compute("⭐ + 3") \
*Here.* <here> \
#compute("⭐ * 2") \
#compute("⭐ - 5")
```

Value at <here> is 13

New value is 10.
New value is 13.
**Here.**
New value is 26.
New value is 21.

## /4 Una advertencia

Para resolver los valores de todos los estados, Typst evalúa partes de tu código varias veces. Sin embargo, no hay garantía de que tu manipulación de estado se pueda resolver por completo.

Por ejemplo, si generás actualizaciones de estado que dependen del valor final de un estado, los resultados podrían no converger nunca. El siguiente ejemplo lo ilustra. Inicializamos nuestro estado con 1 y luego lo actualizamos a su propio valor final más 1. Entonces debería ser 2, pero entonces su valor final es 2, así que debería ser 3, y así sucesivamente. Este ejemplo muestra un valor finito porque Typst simplemente se rinde después de unos cuantos intentos.

```
// This is bad!
#let x = state("key", 1)
#context x.update(x.final() + 1)
#context x.get()
```

5

En general, deberías intentar no generar actualizaciones de estado dentro de expresiones de contexto. Si es posible, intentá expresar tus actualizaciones como valores no contextuales o como funciones que calculan el valor nuevo a partir del valor anterior. A veces no hay otra opción, pero en esos casos te toca a vos asegurarte de que el resultado converja.

## /5 Constructor

Crea un estado nuevo identificado por una clave.

```
state(
  str,
  any,
) → state
```

### /5.1 key `str` *Obligatorio* *Posicional*

La clave que identifica este estado.

Cualquier actualización[3.12.7/6.4] del estado se identificará con la clave de tipo cadena de texto. Si construís varios estados con la misma `key`, actualizar uno cualquiera afectará a todos.

### /5.2 init `any` *Posicional* Por defecto: none

El valor inicial del estado.

Si construís varios estados con la misma `key` pero distintos valores `init`, cada uno usará su propio valor inicial pero compartirán las actualizaciones. Específicamente, el valor de un estado en cierta ubicación del documento se calculará a partir del valor inicial de ese estado y de todas las actualizaciones anteriores de la clave del estado.

```
#let banana = state("key", "🍌")
#let broccoli = state("key", "🥦")

#banana.update(it => it + "😋")

#context [
  - #state("key", "🍎").get()
  - #banana.get()
  - #broccoli.get()
]
```

- 🍎😋
- 🍌😋
- 🥦😋

## /6 Funciones

### /6.1 get Contextual Desde: 0.11.0

Obtiene el valor del estado en la ubicación actual.

Esto equivale a `state.at(here())`.

```
self.get() → any
```

### /6.2 at Contextual Desde: siempre

Obtiene el valor del estado en la coincidencia única del selector dado.

El `selector` debe coincidir exactamente con un elemento del documento. Los tipos de selectores más útiles para esto son las etiquetas[3.5.18] y las ubicaciones[3.12.4].

```
self.at(label selector location function) → any
```

#### /6.2.1 selector label or selector or location or function *Obligatorio* *Posicional*

El lugar en el que se debe obtener el valor del estado.

### /6.3 final Contextual Desde: siempre

Obtiene el valor del estado al final del documento.

```
self.final() → any
```

### /6.4 update Desde: siempre

Actualiza el valor del estado.

Devuelve un fragmento invisible de contenido[3.5.8] que hay que insertar en el documento para que tenga efecto. Este contenido invisible le indica a Typst que la actualización especificada debe ocurrir donde el contenido se inserte en el documento.

El estado es parte de tu documento y corre como un hilo incrustado en el contenido del documento. El valor de un estado es el resultado de todas las actualizaciones de estado que ocurrieron en el documento hasta ese punto.

Por eso `state.update` devuelve un fragmento invisible de contenido que tenés que devolver e incluir en el documento: una actualización de estado que no está "ubicada" en el documento no ocurre, y "cuándo" ocurre lo determina dónde la ubicás. Por eso también necesitás contexto[3.4] para leer el estado: necesitás usar la posición actual del documento para saber en qué lugar del "hilo" del estado estás.

Guardar una actualización de estado en una variable (por ejemplo, `let my-update = state("key").update(c => c * 2)`) no tendrá ningún efecto por sí solo. Solo cuando insertás la variable `#my-update` en algún lugar del contenido del documento, la actualización tendrá efecto, en la posición donde se insertó. También podés usar `#my-update` varias veces en distintas posiciones. Entonces, la actualización también tendrá efecto varias veces.

A diferencia de `get`[3.12.7/6.1], `at`[3.12.7/6.2] y `final`[3.12.7/6.3], esta función no requiere contexto[3.4]. Esto se debe a que, para crear la actualización de estado, no necesitamos saber en qué parte del documento estamos. Solo necesitamos esa información para resolver el valor del estado.

```
self.update(function any) → content
```

#### /6.4.1 update function or any *Obligatorio* *Posicional*

Un valor al que actualizar o una función con la que actualizar.

- Si se le da un valor que no es una función, establece el estado en ese valor.
- Si se le da una función, esa función recibe el valor anterior del estado y tiene que devolver el valor nuevo.

Al actualizar el estado a partir de su valor anterior, conviene preferir la forma con función en lugar de obtener el valor anterior del contexto[3.4]. Esto permite que el compilador resuelva el estado final de manera eficiente y minimice la cantidad de iteraciones de diseño[3.4.4] necesarias.

En el siguiente ejemplo, `fill.update(f => not f)` pintará los elementos impares de la lista con viñetas[3.6.2/4.1] como se espera. Sin embargo, si se lo reemplaza por `context fill.update(not fill.get())`, el diseño no convergerá en 5 intentos, ya que cada actualización necesitará una iteración adicional para propagarse.

```
#let fill = state("fill", false)

#show list.item: it => {
  fill.update(f => not f)
  context {
    set text(fill: fuchsia) if fill.get()
    it
  }
}

#lorem(5).split().map(list.item).join()
```

- Lorem
- ipsum
- dolor
- sit
- amet.

## 3.13 Carga de datos

Carga de datos desde archivos externos.

Estas funciones te ayudan a cargar e incrustar datos, por ejemplo, los resultados de un experimento.

### 3.13.1 Codificación

Algunas de las funciones también pueden codificar, por ejemplo `cbor.encode`[3.13.2/3.1]. Facilitan pasar datos estructurados a plugins[3.5.23].

Sin embargo, cada formato de datos tiene sus propios tipos nativos. Por lo tanto, para un valor arbitrario de Typst, el viaje de ida y vuelta de codificar y decodificar puede perder información. En general, los números, las cadenas de texto y los arrays[3.5.2] o diccionarios[3.5.12] compuestos por ellos se pueden convertir de forma confiable, mientras que otros tipos pueden recurrir a cadenas de texto mediante `repr`[3.5.25], que es solo para depuración[3.5.25/2]. Consultá la página de cada formato de datos para más detalles.

## Definiciones


3.13.2 CBOR Reads structured data from a CBOR file. . . . . 504
3.13.3 CSV Reads structured data from a CSV file. . . . . 505
3.13.4 JSON Reads structured data from a JSON file. . . . . 506
3.13.5 Read Reads plain text or data from a file. . . . . 508
3.13.6 TOML Reads structured data from a TOML file. . . . . 509
3.13.7 XML Reads structured data from an XML file. . . . . 511
3.13.8 YAML Reads structured data from a YAML file. . . . . 513

### 3.13.2 `cbor` Desde: 0.8.0

Lee datos estructurados de un archivo CBOR.

El archivo debe contener una serialización CBOR válida. Los valores CBOR se convertirán en los valores de Typst correspondientes, según se enumera en la tabla de abajo[3.13.2/1].

La función devuelve un diccionario, un array u otro tipo de dato CBOR, según el archivo CBOR.

#### /1 Detalles de la conversión

| Valor CBOR | Convertido a Typst |
|---|---|
| entero | `int`[3.5.17] (o `float`[3.5.15]) |
| bytes | `bytes`[3.5.7] |
| flotante | `float`[3.5.15] |
| texto | `str`[3.5.28] |
| booleano | `bool`[3.5.6] |
| null | none |
| array | `array`[3.5.2] |
| mapa | `dictionary`[3.5.12] |

| Valor de Typst | Convertido a CBOR |
|---|---|
| tipos que se pueden convertir desde CBOR | valor CBOR correspondiente |
| `symbol`[3.5.29] | texto |
| `content`[3.5.8] | un mapa que describe el contenido |
| otros tipos (`length`[3.10.13], etc.) | texto mediante `repr`[3.5.25] |

#### /1.1 Notas

- Tené en cuenta que los enteros CBOR mayores que $2^{63}-1$ o menores que $-2^{63}$ se convertirán en números de punto flotante, lo que puede dar como resultado un valor aproximado.
- Las etiquetas CBOR (tags) no se admiten y se producirá un error.
- La función `repr` es solo para depuración[3.5.25/2] y no se garantiza que su salida sea estable entre versiones de Typst.

#### /2 Parámetros

```
cbor( str  path  bytes ) → any
```

/2.1 **source** `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo CBOR o bytes CBOR sin procesar.

#### /3 Funciones

/3.1 **encode** Desde: 0.8.0

Codifica datos estructurados en bytes CBOR.

```
cbor.encode( any ) → bytes
```

/3.1.1 **value** `any` *Obligatorio* *Posicional*

Valor que se codifica.

### 3.13.3 `csv` Desde: siempre

Lee datos estructurados de un archivo CSV.

El archivo CSV se leerá y se analizará como un array bidimensional de cadenas de texto: cada fila del archivo CSV se representará como un array de cadenas y todas las filas se reunirán en un único array. Las filas de encabezado no se eliminan.

#### /1 Ejemplo

```typst
#let results = csv("example.csv")

#table(
  columns: 2,
  [*Condition*], [*Result*],
  ..results.flatten(),
)
```

| Condition | Result |
|---|---|
| 0..2 | small |
| 3..5 | medium |
| 6.. | big |

#### /2 Parámetros

```
csv(
  str path bytes,
  delimiter: str,
  row-type: type,
) → array
```

/2.1 **`source`** `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo CSV o bytes CSV sin procesar.

/2.2 **`delimiter`** `str` Por defecto: `","`

El delimitador que separa las columnas del archivo CSV. Debe ser un único carácter ASCII.

/2.3 **`row-type`** `type` Por defecto: `array`

Cómo se representan las filas del archivo.

- Si se establece en `array`, cada fila se representa como un array simple de cadenas de texto.
- Si se establece en `dictionary`, cada fila se representa como un diccionario que asigna claves de encabezado a cadenas de texto. Esta opción solo tiene sentido cuando el archivo CSV tiene una fila de encabezado.

### 3.13.4 `json` Desde: siempre

Lee datos estructurados de un archivo JSON.

El archivo debe contener un valor JSON válido, como un objeto o un array. Los valores JSON se convertirán en los valores de Typst correspondientes, según se enumera en la tabla de abajo[3.13.4/2].

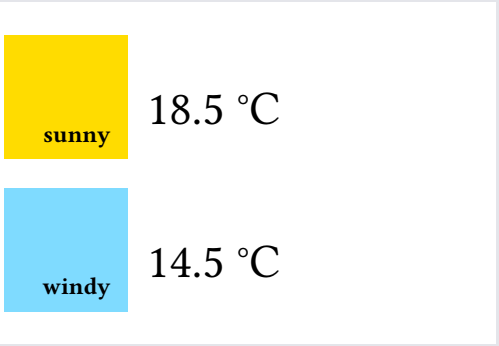
La función devuelve un diccionario, un array u otro tipo de dato JSON, según el archivo JSON.

Los archivos JSON del ejemplo contienen objetos con las claves `temperature`, `unit` y `weather`.

#### /1 Ejemplo

```
#let forecast(day) = block[
  #box(square(
    width: 2cm,
    inset: 8pt,
    fill: if day.weather == "sunny" {
      yellow
    } else {
      aqua
    },
    align(
      bottom + right,
      strong(day.weather),
    ),
  ))
  #h(6pt)
  #set text(22pt, baseline: -8pt)
  #day.temperature °#day.unit
]

#forecast(json("monday.json"))
#forecast(json("tuesday.json"))
```




#### /2 Detalles de la conversión

| Valor JSON | Convertido a Typst |
|---|---|
| null | none |
| booleano | bool[3.5.6] |
| número | float[3.5.15] o int[3.5.17] |
| cadena | str[3.5.28] |
| array | array[3.5.2] |
| objeto | dictionary[3.5.12] |

| Valor de Typst | Convertido a JSON |
|---|---|
| tipos que se pueden convertir desde JSON | valor JSON correspondiente |
| bytes[3.5.7] | cadena mediante repr[3.5.25] |
| symbol[3.5.29] | cadena |
| content[3.5.8] | un objeto que describe el contenido |
| otros tipos (length[3.10.13], etc.) | cadena mediante repr[3.5.25] |

##### /2.1 Notas

- En la mayoría de los casos, los números JSON se convertirán en flotantes o enteros según sean o no números enteros. Sin embargo, tené en cuenta que los enteros mayores que $2^{63}-1$ o menores que $-2^{63}$ se convertirán en números de punto flotante, lo que puede dar como resultado un valor aproximado.

- Los bytes no se codifican como arrays JSON por razones de rendimiento y legibilidad. Considerá usar `cbor.encode`[3.13.2/3.1] para datos binarios.
- La función `repr` es solo para depuración[3.5.25/2] y no se garantiza que su salida sea estable entre versiones de Typst.

## /3 Parámetros

```
json( str  path  bytes ) → any
```

### /3.1 **source** `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo JSON o bytes JSON sin procesar.

## /4 Funciones

### /4.1 **encode** Desde: 0.8.0

Codifica datos estructurados en una cadena JSON.

```
json.encode(
  any ,
  pretty: bool ,
) → str
```

#### /4.1.1 **value** `any` *Obligatorio* *Posicional*

Valor que se codifica.

#### /4.1.2 **pretty** `bool` Por defecto: `true`

Si se debe dar formato legible al JSON, con saltos de línea e indentación.

### 3.13.5 `read` Desde: siempre

Lee texto plano o datos de un archivo.

Por defecto, el archivo se leerá como UTF-8 y se devolverá como una cadena de texto[3.5.28].

Si especificás `encoding: none`, devuelve bytes[3.5.7] sin procesar.

#### /1 Ejemplo

```
An example for a HTML file: \
#let text = read("example.html")
#raw(text, block: true, lang: "html")

Raw bytes:
#read("tiger.jpg", encoding: none)
```

An example for a HTML file:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Example document</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

Raw bytes: `bytes(116679)`

#### /2 Parámetros

```
read(
  str path,
  encoding: none str,
) → str bytes
```

##### /2.1 **path** `str` or `path` *Obligatorio* *Posicional*

Ruta a un archivo.

##### /2.2 **encoding** `none` or `str` Por defecto: `"utf8"`

La codificación con la que se lee el archivo.

Si se establece en `none`, esta función devuelve bytes sin procesar.

| Variante | Detalles |
|---|---|
| `"utf8"` | La codificación Unicode UTF-8. |

### 3.13.6 `toml` Desde: 0.3.0

Lee datos estructurados de un archivo TOML.

El archivo debe contener una tabla TOML válida. Los valores TOML se convertirán en los valores de Typst correspondientes, según se enumera en la tabla de abajo[3.13.6/2].

La función devuelve un diccionario que representa la tabla TOML.

El archivo TOML del ejemplo consiste en una tabla con las claves `title`, `version` y `authors`.

#### /1 Ejemplo

```
#let details = toml("details.toml")

Title: #details.title \
Version: #details.version \
Authors: #(details.authors
  .join(", ", last: " and "))
```

Title: Secret project
Version: 2
Authors: Mr Robert and Miss Enola

#### /2 Detalles de la conversión

Antes que nada, los documentos TOML son tablas. Los demás valores deben ponerse en una tabla para codificarse o decodificarse.

| Valor TOML | Convertido a Typst |
|---|---|
| cadena | `str`[3.5.28] |
| entero | `int`[3.5.17] |
| flotante | `float`[3.5.15] |
| booleano | `bool`[3.5.6] |
| datetime | `datetime`[3.5.10] |
| array | `array`[3.5.2] |
| tabla | `dictionary`[3.5.12] |

| Valor de Typst | Convertido a TOML |
|---|---|
| tipos que se pueden convertir desde TOML | valor TOML correspondiente |
| none | se ignora |
| `bytes`[3.5.7] | cadena mediante `repr`[3.5.25] |
| `symbol`[3.5.29] | cadena |
| `content`[3.5.8] | una tabla que describe el contenido |
| otros tipos (`length`[3.10.13], etc.) | cadena mediante `repr`[3.5.25] |

#### /2.1 Notas

- Tené en cuenta que los enteros TOML mayores que $2^{63}-1$ o menores que $-2^{63}$ no se pueden representar sin pérdida en Typst y se producirá un error, según la especificación[A].
- Los bytes no se codifican como arrays TOML por razones de rendimiento y legibilidad. Considerá usar `cbor.encode`[3.13.2/3.1] para datos binarios.
- La función `repr` es solo para depuración[3.5.25/2] y no se garantiza que su salida sea estable entre versiones de Typst.

[A] https://toml.io/en/v1.0.0#integer

## /3 Parámetros

```
toml( str  path  bytes ) → dictionary
```

### /3.1 source `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo TOML o bytes TOML sin procesar.

## /4 Funciones

### /4.1 encode Desde: 0.8.0

Codifica datos estructurados en una cadena TOML.

```
toml.encode(
  dictionary ,
  pretty: bool ,
) → str
```

#### /4.1.1 value `dictionary` *Obligatorio* *Posicional*

Valor que se codifica.

Los documentos TOML son tablas. Por lo tanto, solo son adecuados los diccionarios.

#### /4.1.2 pretty `bool` Por defecto: `true`

Si se debe dar formato legible al TOML resultante.

### 3.13.7 `xml` Desde: siempre

Lee datos estructurados de un archivo XML.

El archivo XML se analiza como un array de diccionarios y cadenas de texto. Los nodos XML pueden ser elementos o cadenas de texto. Los elementos se representan como diccionarios con las siguientes claves:

- `tag`: El nombre del elemento como cadena de texto.
- `attrs`: Un diccionario con los atributos del elemento como cadenas de texto.
- `children`: Un array con los nodos hijos del elemento.

El archivo XML del ejemplo contiene una etiqueta raíz `news` con varias etiquetas `article`. Cada artículo tiene una etiqueta `title`, `author` y `content`. La etiqueta `content` contiene uno o más párrafos, que se representan como etiquetas `p`.

#### Ejemplo

```
#let find-child(elem, tag) = {
  elem.children
    .find(e => "tag" in e and e.tag == tag)
}

#let article(elem) = {
  let title = find-child(elem, "title")
  let author = find-child(elem, "author")
  let pars = find-child(elem, "content")

  [= #title.children.first()]
  text(10pt, weight: "medium")[
    Published by
    #author.children.first()
  ]

  for p in pars.children {
    if type(p) == dictionary {
      parbreak()
      p.children.first()
    }
  }
}

#let data = xml("example.xml")
#for elem in data.first().children {
  if type(elem) == dictionary {
    article(elem)
  }
}
```




## /2 Parámetros

```
xml(str path bytes) → any
```

### /2.1 source `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo XML o bytes XML sin procesar.

### 3.13.8 `yaml` Desde: 0.1.0

Lee datos estructurados de un archivo YAML.

El archivo debe contener un objeto o array YAML válido. Los valores YAML se convertirán en los valores de Typst correspondientes, según se enumera en la tabla de abajo[3.13.8/2].

La función devuelve un diccionario, un array u otro tipo de dato YAML, según el archivo YAML.

Los archivos YAML del ejemplo contienen objetos con autores como claves, cada uno con una secuencia de su propio submapeo con las claves "title" y "published".

#### /1 Ejemplo

```
#let bookshelf(contents) = {
  for (author, works) in contents {
    author
    for work in works [
      - #work.title (#work.published)
    ]
  }
}

#bookshelf(
  yaml("scifi-authors.yaml")
)
```

Arthur C. Clarke
- Against the Fall of Night (1978)
- The songs of distant earth (1986)

Isaac Asimov
- Quasar, Quasar, Burning Bright (1977)
- Far as Human Eye Could See (1987)

#### /2 Detalles de la conversión

| Valor YAML | Convertido a Typst |
|---|---|
| valores nulos (`null`, ~ o vacío ) | none |
| booleano | `bool`[3.5.6] |
| número | `float`[3.5.15] o `int`[3.5.17] |
| cadena | `str`[3.5.28] |
| secuencia | `array`[3.5.2] |
| mapeo | `dictionary`[3.5.12] |

| Valor de Typst | Convertido a YAML |
|---|---|
| tipos que se pueden convertir desde YAML | valor YAML correspondiente |
| `bytes`[3.5.7] | cadena mediante `repr`[3.5.25] |
| `symbol`[3.5.29] | cadena |
| `content`[3.5.8] | un mapeo que describe el contenido |
| otros tipos (`length`[3.10.13], etc.) | cadena mediante `repr`[3.5.25] |

##### /2.1 Notas

- En la mayoría de los casos, los números YAML se convertirán en flotantes o enteros según sean o no números enteros. Sin embargo, tené en cuenta que los enteros mayores que $2^{63}-1$ o menores que $-2^{63}$ se convertirán en números de punto flotante, lo que puede dar como resultado un valor aproximado.
- Las etiquetas YAML personalizadas se ignoran, aunque el valor cargado seguirá presente.
- Los bytes no se codifican como secuencias YAML por razones de rendimiento y legibilidad. Considerá usar `cbor.encode`[3.13.2/3.1] para datos binarios.
- La función `repr` es solo para depuración[3.5.25/2] y no se garantiza que su salida sea estable entre versiones de Typst.

## /3 Parámetros

```
yaml(str path bytes) → any
```

### /3.1 source `str` or `path` or `bytes` *Obligatorio* *Posicional*

Una ruta a un archivo YAML o bytes YAML sin procesar.

## /4 Funciones

### /4.1 encode Desde: 0.8.0

Codifica datos estructurados en una cadena YAML.

```
yaml.encode(any) → str
```

#### /4.1.1 value `any` *Obligatorio* *Posicional*

Valor que se codifica.

# 3.14 Formatos

Los documentos de Typst se pueden exportar a varios formatos de exportación distintos. Esta sección documenta los formatos de exportación disponibles y enumera las configuraciones y características específicas de cada formato.

Cada formato de exportación tiene un elemento asociado con el que se puede configurar. Estos elementos están definidos en el módulo global `format`. Los formatos principales `pdf` y `html` además están directamente disponibles en el ámbito global.

## 3.14.1 Configurar los ajustes de exportación

Hay dos maneras de especificar los ajustes de exportación:

- Al momento de exportar, mediante argumentos de línea de comandos o el panel "Export & Preview" de la app web
- En el documento, mediante una regla set[3.2.1] sobre el elemento del formato

A continuación hay un ejemplo que muestra cómo podrías cambiar el estándar de PDF[3.14.4/3.2] por defecto de un documento. A menos que se especifique otra cosa (por ejemplo, mediante un argumento de línea de comandos), un documento con esta regla set se exportará como PDF/UA-1.

```
// The document will now default to PDF/UA-1.
#set pdf(standard: "ua-1")
```

De manera similar, podemos escribir una regla set que haga que los PNG renderizados sean un poco más nítidos por defecto. Acá tenemos que escribir `format.png`[3.14.5] en lugar de solo `png`, ya que el elemento del formato PNG no está disponible globalmente (es un formato de exportación un poco más de nicho que PDF).

```
// This makes PNGs a bit higher-res.
#set format.png(ppi: 300)
```

## Definiciones


3.14.2 Bundle Typst's bundle export format (experimental) ..... 516
3.14.3 HTML Typst's HTML export format (experimental) ..... 519
3.14.4 PDF Typst's PDF export format ..... 572
3.14.5 PNG Typst's PNG export format ..... 587
3.14.6 SVG Typst's SVG export format ..... 589

### 3.14.2 `bundle` Elemento Desde: unreleased Requires the `bundle` feature

El formato de exportación a bundle de Typst (experimental).

> INFO
>
> La exportación a bundle solo está disponible para experimentar detrás de un indicador de característica. No uses esta característica en casos de uso de producción. En la CLI, podés experimentar con ella pasando `--features bundle` o estableciendo la variable de entorno `TYPST_FEATURES` en `bundle`. Para usar las características `bundle` y `html` al mismo tiempo, especificá ambas separadas por una coma (es decir, `bundle,html`). En la app web, la exportación a bundle no está disponible por el momento.

Con la exportación a bundle de Typst, podés emitir varios archivos de salida a partir de un único proyecto de Typst. La salida en bundle es útil para crear sitios web de varias páginas con exportación a HTML, pero no se limita a la exportación a HTML. Podés crear bundles que contengan cualquier combinación de páginas HTML[3.14.3], PDF[3.14.4], PNG[3.14.5], SVG[3.14.6] y recursos[3.14.2/5] arbitrarios.

## /1 Exportar como bundle

### /1.1 Línea de comandos

Pasá `--format bundle` al subcomando `compile` o `watch`. Tené en cuenta que también tenés que pasar `--features bundle` o establecer `TYPST_FEATURES=bundle` para habilitar este destino de exportación experimental.

Al usar `typst watch`, Typst lanzará un servidor HTTP con recarga en vivo que sirve tus archivos. Podés configurarlo así:

- Pasá `--port` para cambiar el puerto. (Por defecto, el primer puerto libre del rango 3000-3005.)
- Pasá `--no-reload` para desactivar la inyección de un script de recarga en vivo en las páginas HTML. (El HTML que se escribe en el disco no se ve afectado en ningún caso.) Los documentos que no son HTML no admiten la recarga en vivo.
- Pasá `--no-serve` para desactivar por completo el servidor.

### /1.2 App web

Actualmente no está disponible.

## /2 Crear archivos

Un bundle es una colección de archivos. Cada uno de estos archivos del bundle pertenece a una de dos categorías: documento o recurso. Un `document`[3.6.5] toma contenido[3.5.8] que se exporta con uno de los otros formatos de exportación de Typst. Por su parte, un `asset`[3.14.2/5] toma datos de bytes[3.5.7] sin procesar de tu elección que se escribirán en el disco tal cual. Ambos elementos toman como primer argumento la ruta de salida deseada.

El siguiente ejemplo muestra un caso básico de cómo se podría usar la exportación a bundle en la práctica:

```
#document("index.html", title: [Home])[
  #title()
  - #link(<blog>)[Go to blog]
]

#document("blog.html", title: [Blog])[
  #title()
  Welcome to my blog!

  ...

  This blog also exists as a
  #link(<blog-pdf>)[single PDF].
```

```
] <blog>

#document("blog.pdf", title: [Blog])[
  ...
] <blog-pdf>

#asset(
  "favicon.ico",
  read("images/favicon.ico", encoding: none),
)
```

En el ejemplo, creamos dos documentos HTML: una página de inicio y un blog. La página de inicio enlaza al blog mediante un enlace de etiqueta. El mecanismo de enlaces incorporado de Typst admite de forma nativa los enlaces entre documentos[3.6.10/6] y resuelve por vos las rutas relativas correctas. El bundle también contiene una versión en PDF del blog, que se enlaza desde la versión HTML. En la práctica, ahora podrías compartir el contenido entre la versión HTML y la PDF guardándolo en una variable y usándola en ambas. Esto se omite acá por brevedad. Por último, el bundle contiene un recurso de ícono para el sitio web. En este caso, proporcionamos los datos del recurso leyendo un archivo del disco. Como alternativa, también es posible generar los datos del recurso desde dentro de Typst (por ejemplo, con una función como `json.encode`[3.13.4/4.1]).

Los documentos y los recursos son elementos normales, así que podés usarlos con los mecanismos habituales de scripting, estilos e introspección de Typst. Para más detalles, consultá la documentación de `document`[3.6.5] y `asset`[3.14.2/5].

## /3 Introspección

Las introspecciones siempre observan el bundle completo y no los documentos individuales. Por ejemplo, consultar los títulos te dará todos los títulos de todos los documentos y no solo los del documento actual. De manera similar, las etiquetas son globales al bundle: podés ubicar y enlazar etiquetas de otros documentos[3.6.10/6]. Los contadores y el estado también son globales.[A] En particular, si habilitás cosas como la numeración de títulos, la numeración avanzará de forma consecutiva a lo largo de todo el bundle.

Si estás usando la exportación a bundle para construir una obra conceptual que se divide en varios archivos de salida, normalmente esto es lo que querés. Si, en cambio, estás usando la exportación a bundle para exportar varias obras conceptualmente separadas, quizás quieras que las introspecciones consideren cada documento de forma aislada. Actualmente, vas a tener que hacerlo manualmente (por ejemplo, reiniciando contadores, ajustando selectores, etcétera). Nosotros planeamos ofrecer más herramientas[B] para administrar el alcance preciso de la introspección en el futuro.

## /4 Parámetros

```
format.bundle() → content
```

## Definiciones

3.14.2/5 Asset Adds a custom file to a bundle. . . . . 518

[A] Una excepción a esto es el contador de páginas, que (naturalmente) es por documento.
[B] https://github.com/typst/typst/issues/7735#issuecomment-3908841853

## /5 asset Elemento Desde: 0.15.0 Requires the bundle feature

Agrega un archivo personalizado a un bundle.

Esta función crea un único archivo en un bundle[3.14.2], a partir de datos de bytes sin procesar[3.5.7]. A diferencia de los documentos[3.6.5], los recursos (assets) se emitirán tal cual, sin pasar por compilación.

La función `asset` se puede combinar con `read`[3.13.5] para copiar un archivo del proyecto al bundle de salida. El primer argumento de `asset` define la ruta de salida del recurso en el bundle, mientras que la ruta que se le pasa a `read` define desde dónde del proyecto se leen los datos.

```
// Copy the file `styles.css` into the bundle.
#asset("styles.css", read("styles.css"))
```

Dicho esto, `asset` no está atado a `read`. También podés generar bytes directamente o usar una función como `json.encode`[3.13.4/4.1] para emitir datos serializados.

```
// Emits a JSON file with the number
// of headings in the document.
#context {
  let headings = query(heading)
  let meta = (
    count: headings.len(),
  )
  asset("meta.json", json.encode(meta))
}

#document("doc.pdf")[
  = Introduction
  = Conclusion
]
```

Esto emitiría un archivo `meta.json` con el siguiente contenido en el bundle resultante:

```
{
  "count": 2
}
```

Esta función está definida en el elemento `format.bundle`[3.14.2], pero también está disponible directamente en el ámbito global. Solo se puede usar en el destino bundle[3.14.2].

### /5.1 Parámetros

```
asset(
  str,
  str bytes,
) → content
```

#### /5.1.1 path `str` *Obligatorio* *Posicional*

La ruta en el bundle en la que se ubicará el recurso.

Puede contener barras interiores, en cuyo caso los directorios intermedios se crearán automáticamente.

#### /5.1.2 data `str` or `bytes` *Obligatorio* *Posicional*

Los datos sin procesar que se escribirán en el archivo de la ruta especificada.

Si se da una cadena de texto, se codificará en UTF-8.

### 3.14.3 `html` Elemento Desde: 0.13.0 Requires the `html` feature

El formato de exportación a HTML de Typst (experimental).

> INFO
>
> La exportación a HTML de Typst está actualmente en desarrollo activo. La característica todavía es muy incompleta y solo está disponible para experimentar detrás de un indicador de característica. No uses esta característica en casos de uso de producción. En la CLI, podés experimentar con la exportación a HTML pasando `--features html` o estableciendo la variable de entorno `TYPST_FEATURES` en `html`. En la app web, podés habilitarla igualmente como característica experimental. Visitá el issue de seguimiento[A] para seguir el progreso de la exportación a HTML y conocer más sobre las características planeadas.

Los archivos HTML describen un documento de forma estructural. El objetivo de la exportación a HTML de Typst es capturar la estructura de un documento de entrada y producir HTML semánticamente rico que conserve esa estructura. El HTML resultante debería ser accesible, legible por personas y editable a mano y con herramientas posteriores.

La exportación a PDF, PNG y SVG, en cambio, producen todas representaciones *visuales* de un documento totalmente dispuesto. Esta divergencia en las intenciones de los formatos significa que Typst no puede simplemente producir HTML perfecto para tus documentos de Typst existentes. No siempre puede saber cuál es la mejor representación semántica en HTML de tu contenido.

En cambio, te da a *vos* el control total: podés comprobar el formato de exportación actual mediante la función `target`[3.5.3-0] y, cuando está establecido en HTML, generar elementos HTML sin procesar[3.14.3-4]. El uso principal previsto de estos elementos es en plantillas y reglas show. De esta manera, el contenido del documento puede ser totalmente independiente del destino de exportación y el contenido se puede compartir entre la exportación a PDF y a HTML.

En el formato de exportación `html`, Typst producirá un único archivo HTML. También podés usar Typst para crear un sitio web que conste de varios documentos HTML y recursos adicionales. Typst llama a esa colección un *bundle* y admite este caso de uso con el destino `bundle`[3.14.2] dedicado. Consultá su documentación para aprender más. Tené en cuenta que, tanto en el destino `html` como en el destino `bundle`, Typst actualmente siempre emitirá un archivo HTML independiente. Está planeado para el futuro admitir la emisión de fragmentos que se puedan integrar en otros documentos HTML.

Typst actualmente no produce hojas de estilo CSS, sino que se centra en emitir marcado semántico. Por supuesto, podés escribir tus propios estilos CSS y aun así beneficiarte de compartir tu *contenido* entre PDF y HTML. Para el futuro, planeamos darte la opción de emitir CSS automáticamente, teniendo en cuenta más de tus reglas set existentes.

## /1 Exportar como HTML

### /1.1 Línea de comandos

Pasá `--format html` al subcomando `compile` o `watch`, o proporcioná un nombre de archivo de salida que termine en `.html`. Tené en cuenta que también tenés que pasar `--features html` o establecer `TYPST_FEATURES=html` para habilitar este destino de exportación experimental.

Al usar `typst watch`, Typst lanzará un servidor HTTP con recarga en vivo. Podés configurarlo así:

- Pasá `--port` para cambiar el puerto. (Por defecto, el primer puerto libre del rango 3000-3005.)
- Pasá `--no-reload` para desactivar la inyección de un script de recarga en vivo. (El HTML que se escribe en el disco no se ve afectado en ningún caso.)
- Pasá `--no-serve` para desactivar por completo el servidor.

### /1.2 App web

La exportación a HTML se puede habilitar como característica experimental en la app web.

[A] https://github.com/typst/typst/issues/5512

## /2 Funcionalidad específica de HTML

Typst expone la funcionalidad específica de HTML en el elemento global `html`. Mirá abajo las definiciones y opciones que contiene.

## /3 Parámetros

```
html(pretty: bool) → content
```

### /3.1 `pretty` `bool` *Configurable con set* Por defecto: `false`

Si se da formato legible al documento HTML producido.

Esto da formato a la salida de una manera más legible para las personas, pero menos eficiente en espacio.

## Definiciones

3.14.3/4 Elem An HTML element that can contain Typst content. . . . . 521
3.14.3/5 Frame An element that lays out its content as an inline SVG. . . . . 522
3.14.3/6 HTML tipado Una capa tipada sobre los elementos HTML sin procesar. . . . . 523

## /4 elem Elemento Desde: 0.13.0 Requires the html feature

Un elemento HTML que puede contener contenido de Typst.

La exportación a HTML de Typst genera automáticamente las etiquetas apropiadas para la mayoría de los elementos. Sin embargo, a veces es deseable conservar más control. Por ejemplo, al usar Typst para generar tu blog, podrías usar esta función para envolver cada artículo en una etiqueta `<article>`.

Typst sabe qué es HTML válido. Una etiqueta y sus atributos deben formar HTML sintácticamente válido. Algunas etiquetas, como `meta`, no aceptan contenido. Por lo tanto, no debés proporcionarles un cuerpo. Es posible que agreguemos más comprobaciones en el futuro, así que asegurate de generar HTML válido al usar esta función.

Normalmente, Typst generará por vos las etiquetas `html`, `head` y `body`. Si en cambio las creás con esta función, Typst omitirá las suyas.

```
#html.elem("div", attrs: (style: "background: aqua"))[
  A div with _Typst content_ inside!
]
```

### /4.1 Parámetros

```
html.elem(
  str,
  attrs: dictionary,
  none content,
) → content
```

#### /4.1.1 tag str *Obligatorio* *Posicional*

La etiqueta del elemento.

#### /4.1.2 attrs dictionary *Configurable con set* Por defecto: (:)

Los atributos HTML del elemento.

#### /4.1.3 body none or content *Posicional* *Configurable con set* Por defecto: none

El contenido del elemento HTML.

El cuerpo puede ser contenido arbitrario de Typst.

/5 **`frame`** Elemento Desde: 0.13.0 Requires the `html` feature

Un elemento que dispone su contenido como un SVG en línea.

A veces, convertir contenido de Typst a HTML no es deseable. Puede ser el caso de los gráficos y de otro contenido que depende del posicionamiento y del estilo para transmitir su mensaje.

Esta función te permite usar el motor de diseño de Typst, el mismo que se usaría para la exportación a PDF, SVG y PNG, para renderizar una parte de tu documento exactamente como aparecería al exportarlo en uno de esos formatos. Incrusta el contenido como un SVG en línea.

## /5.1 Parámetros

```
html.frame(content) → content
```

### /5.1.1 **body** `content` *Obligatorio* *Posicional*

El contenido que se debe disponer.

# /6 HTML tipado

Una capa tipada sobre los elementos HTML sin procesar.

El módulo `html` ofrece una capa tipada sobre la función `html.elem`[3.14.3/4] sin procesar que te permite crear elementos HTML cómodamente. Los atributos HTML se exponen como parámetros de función que aceptan tipos de Typst y se encargan automáticamente de convertirlos al HTML apropiado.

Algunos parámetros son comunes a todas las funciones de HTML tipado. Por legibilidad, se enumeran al final, en la sección Atributos globales[3.14.3/6.3], en lugar de hacerlo explícitamente en cada elemento.

## /6.1 Ejemplo

```
#html.video(
  controls: true,
  width: 1280,
  height: 720,
  src: "sunrise.mp4",
)[
  Your browser does not support the video tag.
]
```

## /6.2 Funciones

### /6.2.1 a

Desde: 0.14.0 · Requires the `html` feature

Hipervínculo.

```
html.a(
  download: str,
  href: str,
  hreflang: str,
  ping: str array,
  referrerpolicy: none str,
  rel: str array,
  target: str,
  type: str,
  content,
) → content
```

#### /6.2.1.1 **download** `str`

Indica si se debe descargar el recurso en lugar de navegar hacia él y, en ese caso, con qué nombre de archivo.

#### /6.2.1.2 **href** `str`

Dirección del hipervínculo.

#### /6.2.1.3 **hreflang** `str`

Idioma del recurso enlazado.

#### /6.2.1.4 **ping** `str` or `array`

URL a las que se hace ping.

#### /6.2.1.5 **referrerpolicy** `none` or `str`

Política de referrer para las solicitudes que inicia el elemento.

| Variante | Detalles |
|---|---|
| `"no-referrer"` | |
| `"no-referrer-when-downgrade"` | |

| Variante | Detalles |
|---|---|
| `"same-origin"` | |
| `"origin"` | |
| `"strict-origin"` | |
| `"origin-when-cross-origin"` | |
| `"strict-origin-when-cross-origin"` | |
| `"unsafe-url"` | |

/6.2.1.6 **`rel`** `str` or `array`

Relación entre la ubicación del documento que contiene el hipervínculo y el recurso de destino.

| Variante | Detalles |
|---|---|
| `"alternate"` | |
| `"canonical"` | |
| `"author"` | |
| `"bookmark"` | |
| `"dns-prefetch"` | |
| `"expect"` | |
| `"external"` | |
| `"help"` | |
| `"icon"` | |
| `"manifest"` | |
| `"modulepreload"` | |
| `"license"` | |
| `"next"` | |
| `"nofollow"` | |
| `"noopener"` | |
| `"noreferrer"` | |
| `"opener"` | |
| `"pingback"` | |
| `"preconnect"` | |
| `"prefetch"` | |
| `"preload"` | |
| `"prev"` | |
| `"privacy-policy"` | |
| `"search"` | |
| `"stylesheet"` | |
| `"tag"` | |
| `"terms-of-service"` | |

/6.2.1.7 **`target`** `str`

Navegable para la navegación por hipervínculos.

| Variante | Detalles |
|---|---|
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

/6.2.1.8 **`type`** `str`

Pista del tipo del recurso referenciado.

#### /6.2.1.9 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.2 **abbr** Desde: 0.14.0 Requires the `html` feature

Abreviatura.

```
html.abbr(content) → content
```

#### /6.2.2.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.3 **address** Desde: 0.14.0 Requires the `html` feature

Información de contacto de una página o de un elemento article.

```
html.address(content) → content
```

#### /6.2.3.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.4 **area** Desde: 0.14.0 Requires the `html` feature

Hipervínculo o área muerta de un mapa de imagen.

```
html.area(
  alt: str,
  coords: array,
  download: str,
  href: str,
  ping: str array,
  referrerpolicy: none str,
  rel: str array,
  shape: str,
  target: str,
) → content
```

#### /6.2.4.1 **alt** `str`

Texto de reemplazo para cuando las imágenes no están disponibles.

#### /6.2.4.2 **coords** `array`

Coordenadas de la forma que se creará en un mapa de imagen. Espera un array de números de punto flotante.

#### /6.2.4.3 **download** `str`

Indica si se debe descargar el recurso en lugar de navegar hacia él y, en ese caso, con qué nombre de archivo.

#### /6.2.4.4 **href** `str`

Dirección del hipervínculo.

#### /6.2.4.5 **ping** `str` or `array`

URL a las que se hace ping.

#### /6.2.4.6 **referrerpolicy** `none` or `str`

Política de referrer para las solicitudes que inicia el elemento.

| Variante | Detalles |
|---|---|
| `"no-referrer"` | |

| Variante | Detalles |
| --- | --- |
| `"no-referrer-when-downgrade"` | |
| `"same-origin"` | |
| `"origin"` | |
| `"strict-origin"` | |
| `"origin-when-cross-origin"` | |
| `"strict-origin-when-cross-origin"` | |
| `"unsafe-url"` | |

/6.2.4.7 **rel** `str` or `array`

Relación entre la ubicación del documento que contiene el hipervínculo y el recurso de destino.

| Variante | Detalles |
| --- | --- |
| `"alternate"` | |
| `"canonical"` | |
| `"author"` | |
| `"bookmark"` | |
| `"dns-prefetch"` | |
| `"expect"` | |
| `"external"` | |
| `"help"` | |
| `"icon"` | |
| `"manifest"` | |
| `"modulepreload"` | |
| `"license"` | |
| `"next"` | |
| `"nofollow"` | |
| `"noopener"` | |
| `"noreferrer"` | |
| `"opener"` | |
| `"pingback"` | |
| `"preconnect"` | |
| `"prefetch"` | |
| `"preload"` | |
| `"prev"` | |
| `"privacy-policy"` | |
| `"search"` | |
| `"stylesheet"` | |
| `"tag"` | |
| `"terms-of-service"` | |

/6.2.4.8 **shape** `str`

El tipo de forma que se creará en un mapa de imagen.

| Variante | Detalles |
| --- | --- |
| `"circle"` | |
| `"default"` | |
| `"poly"` | |
| `"rect"` | |

#### /6.2.4.9 **target** `str`

Navegable para la navegación por hipervínculos.

| Variante | Detalles |
|---|---|
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

### /6.2.5 **article** Desde: 0.14.0 Requires the `html` feature

Composición autocontenida, sindicable o reutilizable.

```
html.article(content) → content
```

#### /6.2.5.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.6 **aside** Desde: 0.14.0 Requires the `html` feature

Barra lateral para contenido relacionado de forma tangencial.

```
html.aside(content) → content
```

#### /6.2.6.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.7 **audio** Desde: 0.14.0 Requires the `html` feature

Reproductor de audio.

```
html.audio(
  autoplay: bool,
  controls: bool,
  crossorigin: str,
  loop: bool,
  muted: bool,
  preload: none auto str,
  src: str,
  content,
) → content
```

#### /6.2.7.1 **autoplay** `bool`

Indica que el recurso multimedia puede iniciarse automáticamente al cargar la página.

#### /6.2.7.2 **controls** `bool`

Muestra los controles del agente de usuario.

#### /6.2.7.3 **crossorigin** `str`

Cómo maneja el elemento las solicitudes crossorigin.

| Variante | Detalles |
|---|---|
| `"anonymous"` | |
| `"use-credentials"` | |

#### /6.2.7.4 **loop** `bool`

Indica si se repite en bucle el recurso multimedia.

#### /6.2.7.5 **muted** `bool`

Indica si se silencia el recurso multimedia de forma predeterminada.

#### /6.2.7.6 **preload** `none` or `auto` or `str`

Sugiere cuánta memoria intermedia necesitará probablemente el recurso multimedia.

| Variante | Detalles |
| --- | --- |
| `"metadata"` | |

#### /6.2.7.7 **src** `str`

Dirección del recurso.

#### /6.2.7.8 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.8 **b** Desde: 0.14.0 Requires the `html` feature

Palabras clave.

```
html.b(content) → content
```

#### /6.2.8.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.9 **base** Desde: 0.14.0 Requires the `html` feature

URL base y navegable de destino predeterminado para hipervínculos y formularios.

```
html.base(
  href: str,
  target: str,
) → content
```

#### /6.2.9.1 **href** `str`

URL base del documento.

#### /6.2.9.2 **target** `str`

Navegable predeterminado para la navegación por hipervínculos y el envío de formularios.

| Variante | Detalles |
| --- | --- |
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

### /6.2.10 **bdi** Desde: 0.14.0 Requires the `html` feature

Aislamiento de la direccionalidad del texto.

```
html.bdi(content) → content
```

#### /6.2.10.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.11 **bdo** Desde: 0.14.0 Requires the `html` feature

Formato de direccionalidad del texto.

```
html.bdo(content) → content
```

#### /6.2.11.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.12 **blockquote** Desde: 0.14.0 Requires the `html` feature

Una sección citada de otra fuente.

```
html.blockquote(
  cite: str,
  content,
) → content
```

#### /6.2.12.1 **cite** `str`

Enlace a la fuente de la cita o a más información sobre la edición.

#### /6.2.12.2 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.13 **body** Desde: 0.14.0 Requires the `html` feature

Cuerpo del documento.

```
html.body(content) → content
```

#### /6.2.13.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.14 **br** Desde: 0.14.0 Requires the `html` feature

Salto de línea, p. ej., en un poema o en una dirección postal.

```
html.br() → content
```

### /6.2.15 **button** Desde: 0.14.0 Requires the `html` feature

Control de botón.

```
html.button(
  command: str,
  commandfor: str,
  disabled: bool,
  form: str,
  formaction: str,
  formenctype: str,
  formmethod: str,
  formnovalidate: bool,
  formtarget: str,
  name: str,
  popovertarget: str,
  popovertargetaction: str,
  type: str,
  value: str,
  content,
) → content
```

#### /6.2.15.1 **command** `str`

Indica al elemento de destino qué acción realizar.

| Variante | Detalles |
| --- | --- |
| `"toggle-popover"` | |
| `"show-popover"` | |
| `"hide-popover"` | |
| `"close"` | |
| `"request-close"` | |
| `"show-modal"` | |

/6.2.15.2 **`commandfor`** `str`

Apunta a otro elemento para invocarlo.

/6.2.15.3 **`disabled`** `bool`

Indica si el control de formulario está deshabilitado.

/6.2.15.4 **`form`** `str`

Asocia el elemento con un elemento de formulario.

/6.2.15.5 **`formaction`** `str`

URL que se usa para enviar el formulario.

/6.2.15.6 **`formenctype`** `str`

Tipo de codificación de la lista de entradas que se usa al enviar el formulario.

| Variante | Detalles |
| --- | --- |
| `"application/x-www-form-urlencoded"` | |
| `"multipart/form-data"` | |
| `"text/plain"` | |

/6.2.15.7 **`formmethod`** `str`

Variante que se usa para enviar el formulario.

| Variante | Detalles |
| --- | --- |
| `"GET"` | |
| `"POST"` | |
| `"dialog"` | |

/6.2.15.8 **`formnovalidate`** `bool`

Omite la validación de los controles del formulario al enviarlo.

/6.2.15.9 **`formtarget`** `str`

Navegable para el envío de formularios.

| Variante | Detalles |
| --- | --- |
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

/6.2.15.10 **`name`** `str`

Nombre del elemento que se usa para enviar el formulario y en la API form.elements.

/6.2.15.11 **`popovertarget`** `str`

Apunta a un elemento popover para alternarlo, mostrarlo u ocultarlo.

/6.2.15.12 **popovertargetaction** `str`

Indica si un elemento popover de destino se debe alternar, mostrar u ocultar.

| Variante | Detalles |
|---|---|
| `"toggle"` | |
| `"show"` | |
| `"hide"` | |

/6.2.15.13 **type** `str`

Tipo de botón.

| Variante | Detalles |
|---|---|
| `"submit"` | |
| `"reset"` | |
| `"button"` | |

/6.2.15.14 **value** `str`

Valor que se usa para enviar el formulario.

/6.2.15.15 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.16 **canvas** Desde: 0.14.0 Requires the `html` feature

Lienzo de mapa de bits programable.

```
html.canvas(
  height: int,
  width: int,
  content,
) → content
```

/6.2.16.1 **height** `int`

Dimensión vertical.

/6.2.16.2 **width** `int`

Dimensión horizontal.

/6.2.16.3 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.17 **caption** Desde: 0.14.0 Requires the `html` feature

Epígrafe de tabla.

```
html.caption(content) → content
```

/6.2.17.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.18 **cite** Desde: 0.14.0 Requires the `html` feature

Título de una obra.

```
html.cite(content) → content
```

/6.2.18.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.19 **code** Desde: 0.14.0 Requires the `html` feature

Código de computadora.

```
html.code(content) → content
```

/6.2.19.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.20 **col** Desde: 0.14.0 Requires the `html` feature

Columna de tabla.

```
html.col(span: int) → content
```

/6.2.20.1 **span** `int`

Cantidad de columnas que abarca el elemento.

/6.2.21 **colgroup** Desde: 0.14.0 Requires the `html` feature

Grupo de columnas de una tabla.

```
html.colgroup(
  span: int,
  content,
) → content
```

/6.2.21.1 **span** `int`

Cantidad de columnas que abarca el elemento.

/6.2.21.2 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.22 **data** Desde: 0.14.0 Requires the `html` feature

Equivalente legible por máquinas.

```
html.data(
  value: str,
  content,
) → content
```

/6.2.22.1 **value** `str`

Valor legible por máquinas.

/6.2.22.2 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.23 **datalist** Desde: 0.14.0 Requires the `html` feature

Contenedor de las opciones de un control de lista combinada (combo box).

```
html.datalist(content) → content
```

/6.2.23.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.24 **dd** Desde: 0.14.0 Requires the `html` feature

Contenido de los elementos dt correspondientes.

```
html.dd(content) → content
```

#### /6.2.24.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.25 **del** Desde: 0.14.0 Requires the `html` feature

Una eliminación del documento.

```
html.del(
  cite: str,
  datetime: datetime,
  content,
) → content
```

#### /6.2.25.1 **cite** `str`

Enlace a la fuente de la cita o a más información sobre la edición.

#### /6.2.25.2 **datetime** `datetime`

Fecha y (opcionalmente) hora del cambio.

#### /6.2.25.3 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.26 **details** Desde: 0.14.0 Requires the `html` feature

Control de divulgación para ocultar detalles.

```
html.details(
  name: str,
  open: bool,
  content,
) → content
```

#### /6.2.26.1 **name** `str`

Nombre de un grupo de elementos details mutuamente excluyentes.

#### /6.2.26.2 **open** `bool`

Indica si los detalles son visibles.

#### /6.2.26.3 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.27 **dfn** Desde: 0.14.0 Requires the `html` feature

Instancia que se define.

```
html.dfn(content) → content
```

#### /6.2.27.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.28 **dialog** Desde: 0.14.0 Requires the `html` feature

Cuadro de diálogo o ventana.

```
html.dialog(
  open: bool,
  content,
) → content
```

/6.2.28.1 **open** `bool`

Indica si el cuadro de diálogo se está mostrando.

/6.2.28.2 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.29 **div** Desde: 0.14.0 Requires the `html` feature

Contenedor de flujo genérico, o contenedor de grupos de nombre y valor en elementos dl.

```
html.div(content) → content
```

/6.2.29.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.30 **dl** Desde: 0.14.0 Requires the `html` feature

Lista de asociación formada por cero o más grupos de nombre y valor.

```
html.dl(content) → content
```

/6.2.30.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.31 **dt** Desde: 0.14.0 Requires the `html` feature

Leyenda de los elementos dd correspondientes.

```
html.dt(content) → content
```

/6.2.31.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.32 **em** Desde: 0.14.0 Requires the `html` feature

Énfasis de acento.

```
html.em(content) → content
```

/6.2.32.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.33 **embed** Desde: 0.14.0 Requires the `html` feature

Plugin.

```
html.embed(
  height: int,
  src: str,
  type: str,
  width: int,
) → content
```

/6.2.33.1 **height** `int`

Dimensión vertical.

/6.2.33.2 **src** `str`

Dirección del recurso.

/6.2.33.3 **type** `str`

Tipo de recurso incrustado.

#### /6.2.33.4 **width** `int`

Dimensión horizontal.

### /6.2.34 **fieldset** Desde: 0.14.0 Requires the `html` feature

Grupo de controles de formulario.

```
html.fieldset(
  disabled: bool,
  form: str,
  name: str,
  content,
) → content
```

#### /6.2.34.1 **disabled** `bool`

Indica si los controles de formulario descendientes, salvo los que están dentro de una legend, están deshabilitados.

#### /6.2.34.2 **form** `str`

Asocia el elemento con un elemento de formulario.

#### /6.2.34.3 **name** `str`

Nombre del elemento que se usa para enviar el formulario y en la API form.elements.

#### /6.2.34.4 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.35 **figcaption** Desde: 0.14.0 Requires the `html` feature

Epígrafe de una figura.

```
html.figcaption(content) → content
```

#### /6.2.35.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.36 **figure** Desde: 0.14.0 Requires the `html` feature

Figura con epígrafe opcional.

```
html.figure(content) → content
```

#### /6.2.36.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.37 **footer** Desde: 0.14.0 Requires the `html` feature

Pie de una página o sección.

```
html.footer(content) → content
```

#### /6.2.37.1 **body** `content` *Posicional*

El contenido del elemento HTML.

### /6.2.38 **form** Desde: 0.14.0 Requires the `html` feature

Formulario que el usuario puede enviar.

```
html.form(
  accept-charset: str,
  action: str,
```

```
  autocomplete: bool,
  enctype: str,
  method: str,
  name: str,
  novalidate: bool,
  rel: str array,
  target: str,
  content,
) → content
```

/6.2.38.1 **accept-charset** `str`

Codificaciones de caracteres que se usan al enviar el formulario.

| Variante | Detalles |
|---|---|
| `"utf-8"` | |

/6.2.38.2 **action** `str`

URL que se usa para enviar el formulario.

/6.2.38.3 **autocomplete** `bool`

Configuración predeterminada de la función de autocompletado para los controles del formulario.

/6.2.38.4 **enctype** `str`

Tipo de codificación de la lista de entradas que se usa al enviar el formulario.

| Variante | Detalles |
|---|---|
| `"application/x-www-form-urlencoded"` | |
| `"multipart/form-data"` | |
| `"text/plain"` | |

/6.2.38.5 **method** `str`

Variante que se usa para enviar el formulario.

| Variante | Detalles |
|---|---|
| `"GET"` | |
| `"POST"` | |
| `"dialog"` | |

/6.2.38.6 **name** `str`

Nombre del formulario que se usa en la API document.forms.

/6.2.38.7 **novalidate** `bool`

Omite la validación de los controles del formulario al enviarlo.

/6.2.38.8 **rel** `str` or `array`

Relación entre el documento que contiene el formulario y el destino de su acción.

| Variante | Detalles |
|---|---|
| `"alternate"` | |
| `"canonical"` | |
| `"author"` | |
| `"bookmark"` | |
| `"dns-prefetch"` | |

| Variante | Detalles |
|---|---|
| `"expect"` | |
| `"external"` | |
| `"help"` | |
| `"icon"` | |
| `"manifest"` | |
| `"modulepreload"` | |
| `"license"` | |
| `"next"` | |
| `"nofollow"` | |
| `"noopener"` | |
| `"noreferrer"` | |
| `"opener"` | |
| `"pingback"` | |
| `"preconnect"` | |
| `"prefetch"` | |
| `"preload"` | |
| `"prev"` | |
| `"privacy-policy"` | |
| `"search"` | |
| `"stylesheet"` | |
| `"tag"` | |
| `"terms-of-service"` | |

/6.2.38.9 **target** `str`

Navegable para el envío de formularios.

| Variante | Detalles |
|---|---|
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

/6.2.38.10 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.39 **h1** Desde: 0.14.0 Requires the `html` feature

Título.

```
html.h1(content) → content
```

/6.2.39.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.40 **h2** Desde: 0.14.0 Requires the `html` feature

Título.

```
html.h2(content) → content
```

/6.2.40.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.41 **h3** Desde: 0.14.0 Requires the `html` feature

Título.

```
html.h3(content) → content
```

/6.2.41.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.42 **h4** Desde: 0.14.0 Requires the `html` feature

Título.

```
html.h4(content) → content
```

/6.2.42.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.43 **h5** Desde: 0.14.0 Requires the `html` feature

Título.

```
html.h5(content) → content
```

/6.2.43.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.44 **h6** Desde: 0.14.0 Requires the `html` feature

Título.

```
html.h6(content) → content
```

/6.2.44.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.45 **head** Desde: 0.14.0 Requires the `html` feature

Contenedor de los metadatos del documento.

```
html.head(content) → content
```

/6.2.45.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.46 **header** Desde: 0.14.0 Requires the `html` feature

Ayudas introductorias o de navegación para una página o sección.

```
html.header(content) → content
```

/6.2.46.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.47 **hgroup** Desde: 0.14.0 Requires the `html` feature

Contenedor de encabezado.

```
html.hgroup(content) → content
```

/6.2.47.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.48 **hr** Desde: 0.14.0 Requires the `html` feature

Corte temático.

```
html.hr() → content
```

/6.2.49 **html** Desde: 0.14.0 Requires the `html` feature

Elemento raíz.

```
html.html(content) → content
```

/6.2.49.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.50 **i** Desde: 0.14.0 Requires the `html` feature

Voz alternativa.

```
html.i(content) → content
```

/6.2.50.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.51 **iframe** Desde: 0.14.0 Requires the `html` feature

Navegable hijo.

```
html.iframe(
  allow: str,
  allowfullscreen: bool,
  height: int,
  loading: str,
  name: str,
  referrerpolicy: none str,
  sandbox: str array,
  src: str,
  srcdoc: str,
  width: int,
  content,
) → content
```

/6.2.51.1 **allow** `str`

Política de permisos que se aplica al contenido del iframe.

/6.2.51.2 **allowfullscreen** `bool`

Indica si se permite que el contenido del iframe use requestFullscreen().

/6.2.51.3 **height** `int`

Dimensión vertical.

/6.2.51.4 **loading** `str`

Se usa para determinar el diferimiento de la carga.

| Variante | Detalles |
|---|---|
| `"lazy"` | |
| `"eager"` | |

/6.2.51.5 **name** `str`

Nombre del navegable de contenido.

| Variante | Detalles |
|---|---|
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

/6.2.51.6 **referrerpolicy** `none` or `str`

Política de referrer para las solicitudes que inicia el elemento.

| Variante | Detalles |
|---|---|
| `"no-referrer"` | |
| `"no-referrer-when-downgrade"` | |
| `"same-origin"` | |
| `"origin"` | |
| `"strict-origin"` | |
| `"origin-when-cross-origin"` | |
| `"strict-origin-when-cross-origin"` | |
| `"unsafe-url"` | |

/6.2.51.7 **sandbox** `str` or `array`

Reglas de seguridad para el contenido anidado.

| Variante | Detalles |
|---|---|
| `"allow-downloads"` | |
| `"allow-forms"` | |
| `"allow-modals"` | |
| `"allow-orientation-lock"` | |
| `"allow-pointer-lock"` | |
| `"allow-popups"` | |
| `"allow-popups-to-escape-sandbox"` | |
| `"allow-presentation"` | |
| `"allow-same-origin"` | |
| `"allow-scripts"` | |
| `"allow-top-navigation"` | |
| `"allow-top-navigation-by-user-activation"` | |
| `"allow-top-navigation-to-custom-protocols"` | |

/6.2.51.8 **src** `str`

Dirección del recurso.

/6.2.51.9 **srcdoc** `str`

Un documento para renderizar en el iframe.

/6.2.51.10 **width** `int`

Dimensión horizontal.

/6.2.51.11 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.52 **img** Desde: 0.14.0 Requires the `html` feature

Imagen.

```
html.img(
  alt: str,
  crossorigin: str,
  decoding: auto str,
  fetchpriority: auto str,
  height: int,
  ismap: bool,
  loading: str,
  referrerpolicy: none str,
  sizes: array,
  src: str,
  srcset: array,
  usemap: str,
  width: int,
) → content
```

/6.2.52.1 **alt** `str`

Texto de reemplazo para cuando las imágenes no están disponibles.

/6.2.52.2 **crossorigin** `str`

Cómo maneja el elemento las solicitudes crossorigin.

| Variante | Detalles |
|---|---|
| `"anonymous"` | |
| `"use-credentials"` | |

/6.2.52.3 **decoding** `auto` or `str`

Sugerencia de decodificación que se usa al procesar esta imagen para su presentación.

| Variante | Detalles |
|---|---|
| `"sync"` | |
| `"async"` | |

/6.2.52.4 **fetchpriority** `auto` or `str`

Establece la prioridad de las solicitudes que inicia el elemento.

| Variante | Detalles |
|---|---|
| `"high"` | |
| `"low"` | |

/6.2.52.5 **height** `int`

Dimensión vertical.

/6.2.52.6 **ismap** `bool`

Indica si la imagen es un mapa de imagen del lado del servidor.

/6.2.52.7 **loading** `str`

Se usa para determinar el diferimiento de la carga.

| Variante | Detalles |
|---|---|
| `"lazy"` | |

| Variante | Detalles |
| --- | --- |
| `"eager"` | |

/6.2.52.8 **`referrerpolicy`** `none` or `str`

Política de referrer para las solicitudes que inicia el elemento.

| Variante | Detalles |
| --- | --- |
| `"no-referrer"` | |
| `"no-referrer-when-downgrade"` | |
| `"same-origin"` | |
| `"origin"` | |
| `"strict-origin"` | |
| `"origin-when-cross-origin"` | |
| `"strict-origin-when-cross-origin"` | |
| `"unsafe-url"` | |

/6.2.52.9 **`sizes`** `array`

Tamaños de imagen para distintos diseños de página. Espera un array de diccionarios con las claves `condition` (cadena de texto) y `size` (longitud).

/6.2.52.10 **`src`** `str`

Dirección del recurso.

/6.2.52.11 **`srcset`** `array`

Imágenes para usar en distintas situaciones, p. ej., pantallas de alta resolución, monitores chicos, etc. Espera un array de diccionarios con las claves `src` (cadena de texto) y `width` (entero) o `density` (flotante).

/6.2.52.12 **`usemap`** `str`

Nombre del mapa de imagen que se usa.

/6.2.52.13 **`width`** `int`

Dimensión horizontal.

/6.2.53 **`input`** Desde: 0.14.0 Requires the `html` feature

Control de formulario.

```
html.input(
  accept: str array,
  alpha: bool,
  alt: str,
  autocomplete: str array,
  checked: bool,
  colorspace: str,
  dirname: str,
  disabled: bool,
  form: str,
  formaction: str,
  formenctype: str,
  formmethod: str,
  formnovalidate: bool,
  formtarget: str,
  height: int,
  list: str,
```

```
  max: float datetime str,
  maxlength: int,
  min: float datetime str,
  minlength: int,
  multiple: bool,
  name: str,
  pattern: str,
  placeholder: str,
  popovertarget: str,
  popovertargetaction: str,
  readonly: bool,
  required: bool,
  size: int,
  src: str,
  step: float str,
  type: str,
  value: float color datetime str array,
  width: int,
) → content
```

/6.2.53.1 **accept** `str` or `array`

Pista del tipo de archivo esperado en controles de carga de archivos.

/6.2.53.2 **alpha** `bool`

Permite establecer el componente alfa del color.

/6.2.53.3 **alt** `str`

Texto de reemplazo para cuando las imágenes no están disponibles.

/6.2.53.4 **autocomplete** `str` or `array`

Pista para la función de autocompletado de formularios.

| Variante | Detalles |
|---|---|
| `"shipping"` | |
| `"billing"` | |
| `"name"` | |
| `"honorific-prefix"` | |
| `"given-name"` | |
| `"additional-name"` | |
| `"family-name"` | |
| `"honorific-suffix"` | |
| `"nickname"` | |
| `"username"` | |
| `"new-password"` | |
| `"current-password"` | |
| `"one-time-code"` | |
| `"organization-title"` | |
| `"organization"` | |
| `"street-address"` | |
| `"address-line1"` | |
| `"address-line2"` | |
| `"address-line3"` | |

| Variante | Detalles |
|---|---|
| `"address-level4"` | |
| `"address-level3"` | |
| `"address-level2"` | |
| `"address-level1"` | |
| `"country"` | |
| `"country-name"` | |
| `"postal-code"` | |
| `"cc-name"` | |
| `"cc-given-name"` | |
| `"cc-additional-name"` | |
| `"cc-family-name"` | |
| `"cc-number"` | |
| `"cc-exp"` | |
| `"cc-exp-month"` | |
| `"cc-exp-year"` | |
| `"cc-csc"` | |
| `"cc-type"` | |
| `"transaction-currency"` | |
| `"transaction-amount"` | |
| `"language"` | |
| `"bday"` | |
| `"bday-day"` | |
| `"bday-month"` | |
| `"bday-year"` | |
| `"sex"` | |
| `"url"` | |
| `"photo"` | |
| `"home"` | |
| `"work"` | |
| `"mobile"` | |
| `"fax"` | |
| `"pager"` | |
| `"tel"` | |
| `"tel-country-code"` | |
| `"tel-national"` | |
| `"tel-area-code"` | |
| `"tel-local"` | |
| `"tel-local-prefix"` | |
| `"tel-local-suffix"` | |
| `"tel-extension"` | |
| `"email"` | |
| `"impp"` | |

/6.2.53.5 **checked** `bool`

Indica si el control está marcado.

/6.2.53.6 **colorspace** `str`

El espacio de color del color serializado.

| Variante | Detalles |
| --- | --- |
| `"limited-srgb"` | |
| `"display-p3"` | |

/6.2.53.7 **dirname** `str`

Nombre del control de formulario que se usa para enviar la direccionalidad del elemento al enviar el formulario.

/6.2.53.8 **disabled** `bool`

Indica si el control de formulario está deshabilitado.

/6.2.53.9 **form** `str`

Asocia el elemento con un elemento de formulario.

/6.2.53.10 **formaction** `str`

URL que se usa para enviar el formulario.

/6.2.53.11 **formenctype** `str`

Tipo de codificación de la lista de entradas que se usa al enviar el formulario.

| Variante | Detalles |
| --- | --- |
| `"application/x-www-form-urlencoded"` | |
| `"multipart/form-data"` | |
| `"text/plain"` | |

/6.2.53.12 **formmethod** `str`

Variante que se usa para enviar el formulario.

| Variante | Detalles |
| --- | --- |
| `"GET"` | |
| `"POST"` | |
| `"dialog"` | |

/6.2.53.13 **formnovalidate** `bool`

Omite la validación de los controles del formulario al enviarlo.

/6.2.53.14 **formtarget** `str`

Navegable para el envío de formularios.

| Variante | Detalles |
| --- | --- |
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

/6.2.53.15 **height** `int`

Dimensión vertical.

/6.2.53.16 **list** `str`

Lista de opciones de autocompletado.

/6.2.53.17 **max** `float` or `datetime` or `str`

Valor máximo.

/6.2.53.18 **maxlength** `int`

Longitud máxima del valor.

/6.2.53.19 **min** `float` or `datetime` or `str`

Valor mínimo.

/6.2.53.20 **minlength** `int`

Longitud mínima del valor.

/6.2.53.21 **multiple** `bool`

Indica si se permiten varios valores.

/6.2.53.22 **name** `str`

Nombre del elemento que se usa para enviar el formulario y en la API form.elements.

/6.2.53.23 **pattern** `str`

Patrón con el que debe coincidir el valor del control de formulario.

/6.2.53.24 **placeholder** `str`

Etiqueta visible para el usuario que se ubica dentro del control de formulario.

/6.2.53.25 **popovertarget** `str`

Apunta a un elemento popover para alternarlo, mostrarlo u ocultarlo.

/6.2.53.26 **popovertargetaction** `str`

Indica si un elemento popover de destino se debe alternar, mostrar u ocultar.

| Variante | Detalles |
|---|---|
| `"toggle"` | |
| `"show"` | |
| `"hide"` | |

/6.2.53.27 **readonly** `bool`

Indica si el usuario puede editar el valor.

/6.2.53.28 **required** `bool`

Indica si el control es obligatorio para enviar el formulario.

/6.2.53.29 **size** `int`

Tamaño del control.

/6.2.53.30 **src** `str`

Dirección del recurso.

/6.2.53.31 **step** `float` or `str`

Granularidad con la que debe coincidir el valor del control de formulario.

| Variante | Detalles |
|---|---|
| `"any"` | |

/6.2.53.32 **type** `str`

Tipo de control de formulario.

| Variante | Detalles |
|---|---|
| `"hidden"` | |
| `"text"` | |
| `"search"` | |

| Variante | Detalles |
| --- | --- |
| `"tel"` | |
| `"url"` | |
| `"email"` | |
| `"password"` | |
| `"date"` | |
| `"month"` | |
| `"week"` | |
| `"time"` | |
| `"datetime-local"` | |
| `"number"` | |
| `"range"` | |
| `"color"` | |
| `"checkbox"` | |
| `"radio"` | |
| `"file"` | |
| `"submit"` | |
| `"image"` | |
| `"reset"` | |
| `"button"` | |

/6.2.53.33 **value** `float` or `color` or `datetime` or `str` or `array`

Valor del control de formulario.

/6.2.53.34 **width** `int`

Dimensión horizontal.

/6.2.54 **ins** Desde: 0.14.0 Requires the `html` feature

Una adición al documento.

```
html.ins(
  cite: str,
  datetime: datetime,
  content,
) → content
```

/6.2.54.1 **cite** `str`

Enlace a la fuente de la cita o a más información sobre la edición.

/6.2.54.2 **datetime** `datetime`

Fecha y (opcionalmente) hora del cambio.

/6.2.54.3 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.55 **kbd** Desde: 0.14.0 Requires the `html` feature

Entrada del usuario.

```
html.kbd(content) → content
```

/6.2.55.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.56 **label** Desde: 0.14.0 Requires the html feature

Epígrafe de un control de formulario.

```
html.label(
  for: str,
  content,
) → content
```

/6.2.56.1 **for** str

Asocia la etiqueta con un control de formulario.

/6.2.56.2 **body** content *Posicional*

El contenido del elemento HTML.

/6.2.57 **legend** Desde: 0.14.0 Requires the html feature

Epígrafe de un fieldset.

```
html.legend(content) → content
```

/6.2.57.1 **body** content *Posicional*

El contenido del elemento HTML.

/6.2.58 **li** Desde: 0.14.0 Requires the html feature

Elemento de lista.

```
html.li(
  value: int,
  content,
) → content
```

/6.2.58.1 **value** int

Valor ordinal del elemento de lista.

/6.2.58.2 **body** content *Posicional*

El contenido del elemento HTML.

/6.2.59 **link** Desde: 0.14.0 Requires the html feature

Metadatos de enlace.

```
html.link(
  as: str,
  blocking: str array,
  color: color,
  crossorigin: str,
  disabled: bool,
  fetchpriority: auto str,
  href: str,
  hreflang: str,
  imagesizes: array,
  imagesrcset: array,
  integrity: str,
  media: str,
  referrerpolicy: none str,
  rel: str array,
  sizes: array,
```

```
  type: str,
) → content
```

/6.2.59.1 **as** `str`

Posible destino de una solicitud de precarga (para rel="preload" y rel="modulepreload").

| Variante | Detalles |
|---|---|
| `"audio"` | |
| `"audioworklet"` | |
| `"document"` | |
| `"embed"` | |
| `"font"` | |
| `"frame"` | |
| `"iframe"` | |
| `"image"` | |
| `"json"` | |
| `"manifest"` | |
| `"object"` | |
| `"paintworklet"` | |
| `"report"` | |
| `"script"` | |
| `"serviceworker"` | |
| `"sharedworker"` | |
| `"style"` | |
| `"track"` | |
| `"video"` | |
| `"webidentity"` | |
| `"worker"` | |
| `"xslt"` | |

/6.2.59.2 **blocking** `str` or `array`

Indica si el elemento puede bloquear el renderizado.

| Variante | Detalles |
|---|---|
| `"blocking"` | |

/6.2.59.3 **color** `color`

Color que se usa al personalizar el ícono de un sitio (para rel="mask-icon").

/6.2.59.4 **crossorigin** `str`

Cómo maneja el elemento las solicitudes crossorigin.

| Variante | Detalles |
|---|---|
| `"anonymous"` | |
| `"use-credentials"` | |

/6.2.59.5 **disabled** `bool`

Indica si el enlace está deshabilitado.

/6.2.59.6 **fetchpriority** `auto` or `str`

Establece la prioridad de las solicitudes que inicia el elemento.

| Variante | Detalles |
|---|---|
| `"high"` | |
| `"low"` | |

/6.2.59.7 **`href`** `str`

Dirección del hipervínculo.

/6.2.59.8 **`hreflang`** `str`

Idioma del recurso enlazado.

/6.2.59.9 **`imagesizes`** `array`

Tamaños de imagen para distintos diseños de página (para rel="preload"). Espera un array de diccionarios con las claves `condition` (cadena de texto) y `size` (longitud).

/6.2.59.10 **`imagesrcset`** `array`

Imágenes para usar en distintas situaciones, p. ej., pantallas de alta resolución, monitores chicos, etc. (para rel="preload"). Espera un array de diccionarios con las claves `src` (cadena de texto) y `width` (entero) o `density` (flotante).

/6.2.59.11 **`integrity`** `str`

Metadatos de integridad que se usan en las comprobaciones de Subresource Integrity.

/6.2.59.12 **`media`** `str`

Medios aplicables.

/6.2.59.13 **`referrerpolicy`** `none` or `str`

Política de referrer para las solicitudes que inicia el elemento.

| Variante | Detalles |
|---|---|
| `"no-referrer"` | |
| `"no-referrer-when-downgrade"` | |
| `"same-origin"` | |
| `"origin"` | |
| `"strict-origin"` | |
| `"origin-when-cross-origin"` | |
| `"strict-origin-when-cross-origin"` | |
| `"unsafe-url"` | |

/6.2.59.14 **`rel`** `str` or `array`

Relación entre el documento que contiene el hipervínculo y el recurso de destino.

| Variante | Detalles |
|---|---|
| `"alternate"` | |
| `"canonical"` | |
| `"author"` | |
| `"bookmark"` | |
| `"dns-prefetch"` | |
| `"expect"` | |
| `"external"` | |
| `"help"` | |
| `"icon"` | |
| `"manifest"` | |
| `"modulepreload"` | |

| Variante | Detalles |
| --- | --- |
| `"license"` | |
| `"next"` | |
| `"nofollow"` | |
| `"noopener"` | |
| `"noreferrer"` | |
| `"opener"` | |
| `"pingback"` | |
| `"preconnect"` | |
| `"prefetch"` | |
| `"preload"` | |
| `"prev"` | |
| `"privacy-policy"` | |
| `"search"` | |
| `"stylesheet"` | |
| `"tag"` | |
| `"terms-of-service"` | |

/6.2.59.15 **sizes** `array`

Tamaños de los íconos (para rel="icon"). Espera un array de tamaños. Cada tamaño se especifica como un array de dos enteros (ancho y alto).

/6.2.59.16 **type** `str`

Pista del tipo del recurso referenciado.

/6.2.60 **main** Desde: 0.14.0 Requires the `html` feature

Contenedor del contenido dominante del documento.

```
html.main(content) → content
```

/6.2.60.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.61 **map** Desde: 0.14.0 Requires the `html` feature

Mapa de imagen.

```
html.map(
  name: str,
  content,
) → content
```

/6.2.61.1 **name** `str`

Nombre del mapa de imagen al que se hace referencia desde el atributo usemap.

/6.2.61.2 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.62 **mark** Desde: 0.14.0 Requires the `html` feature

Resaltado.

```
html.mark(content) → content
```

/6.2.62.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.63 **menu** Desde: 0.14.0 Requires the `html` feature

Menú de comandos.

```
html.menu(content) → content
```

/6.2.63.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.64 **meta** Desde: 0.14.0 Requires the `html` feature

Metadatos de texto.

```
html.meta(
  charset: str,
  content: str,
  http-equiv: str,
  media: str,
  name: str,
) → content
```

/6.2.64.1 **charset** `str`

Declaración de la codificación de caracteres.

| Variante | Detalles |
|---|---|
| `"utf-8"` | |

/6.2.64.2 **content** `str`

Valor del elemento.

/6.2.64.3 **http-equiv** `str`

Directiva pragma.

| Variante | Detalles |
|---|---|
| `"content-type"` | |
| `"default-style"` | |
| `"refresh"` | |
| `"x-ua-compatible"` | |
| `"content-security-policy"` | |

/6.2.64.4 **media** `str`

Medios aplicables.

/6.2.64.5 **name** `str`

Nombre de los metadatos.

/6.2.65 **meter** Desde: 0.14.0 Requires the `html` feature

Medidor.

```
html.meter(
  high: float,
  low: float,
  max: float,
  min: float,
  optimum: float,
  value: float,
```

```
  content,
) → content
```

/6.2.65.1 **high** `float`

Límite inferior del rango alto.

/6.2.65.2 **low** `float`

Límite superior del rango bajo.

/6.2.65.3 **max** `float`

Límite superior del rango.

/6.2.65.4 **min** `float`

Límite inferior del rango.

/6.2.65.5 **optimum** `float`

Valor óptimo del medidor.

/6.2.65.6 **value** `float`

Valor actual del elemento.

/6.2.65.7 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.66 **nav** Desde: 0.14.0 Requires the `html` feature

Sección con enlaces de navegación.

```
html.nav(content) → content
```

/6.2.66.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.67 **noscript** Desde: 0.14.0 Requires the `html` feature

Contenido alternativo para un script.

```
html.noscript(content) → content
```

/6.2.67.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.68 **object** Desde: 0.14.0 Requires the `html` feature

Imagen, navegable hijo o plugin.

```
html.object(
  data: str,
  form: str,
  height: int,
  name: str,
  type: str,
  width: int,
  content,
) → content
```

/6.2.68.1 **data** `str`

Dirección del recurso.

/6.2.68.2 **form** `str`

Asocia el elemento con un elemento de formulario.

/6.2.68.3 **height** `int`

Dimensión vertical.

/6.2.68.4 **name** `str`

Nombre del navegable de contenido.

| Variante | Detalles |
|---|---|
| `"_blank"` | |
| `"_self"` | |
| `"_parent"` | |
| `"_top"` | |

/6.2.68.5 **type** `str`

Tipo de recurso incrustado.

/6.2.68.6 **width** `int`

Dimensión horizontal.

/6.2.68.7 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.69 **ol** Desde: 0.14.0 Requires the `html` feature

Lista ordenada.

```
html.ol(
  reversed: bool,
  start: int,
  type: str,
  content,
) → content
```

/6.2.69.1 **reversed** `bool`

Numera la lista hacia atrás.

/6.2.69.2 **start** `int`

Valor inicial de la lista.

/6.2.69.3 **type** `str`

Tipo de marcador de lista.

| Variante | Detalles |
|---|---|
| `"1"` | |
| `"a"` | |
| `"A"` | |
| `"i"` | |
| `"I"` | |

/6.2.69.4 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.70 **optgroup** Desde: 0.14.0 Requires the `html` feature

Grupo de opciones de una lista.

```
html.optgroup(
  disabled: bool,
  label: str,
  content,
) → content
```

/6.2.70.1 **disabled** `bool`

Indica si el control de formulario está deshabilitado.

/6.2.70.2 **label** `str`

Etiqueta visible para el usuario.

/6.2.70.3 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.71 **option** Desde: 0.14.0 Requires the `html` feature

Opción de un control de lista o de lista combinada.

```
html.option(
  disabled: bool,
  label: str,
  selected: bool,
  value: str,
  content,
) → content
```

/6.2.71.1 **disabled** `bool`

Indica si el control de formulario está deshabilitado.

/6.2.71.2 **label** `str`

Etiqueta visible para el usuario.

/6.2.71.3 **selected** `bool`

Indica si la opción está seleccionada de forma predeterminada.

/6.2.71.4 **value** `str`

Valor que se usa para enviar el formulario.

/6.2.71.5 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.72 **output** Desde: 0.14.0 Requires the `html` feature

Valor de salida calculado.

```
html.output(
  for: str array,
  form: str,
  name: str,
  content,
) → content
```

/6.2.72.1 **for** `str` or `array`

Especifica los controles a partir de los cuales se calculó la salida.

/6.2.72.2 **form** `str`

Asocia el elemento con un elemento de formulario.

/6.2.72.3 **name** `str`

Nombre del elemento que se usa para enviar el formulario y en la API form.elements.

/6.2.72.4 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.73 **p** Desde: 0.14.0 Requires the `html` feature

Párrafo.

```
html.p(content) → content
```

/6.2.73.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.74 **picture** Desde: 0.14.0 Requires the `html` feature

Imagen.

```
html.picture(content) → content
```

/6.2.74.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.75 **pre** Desde: 0.14.0 Requires the `html` feature

Bloque de texto preformateado.

```
html.pre(content) → content
```

/6.2.75.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.76 **progress** Desde: 0.14.0 Requires the `html` feature

Barra de progreso.

```
html.progress(
  max: float,
  value: float,
  content,
) → content
```

/6.2.76.1 **max** `float`

Límite superior del rango.

/6.2.76.2 **value** `float`

Valor actual del elemento.

/6.2.76.3 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.77 **q** Desde: 0.14.0 Requires the `html` feature

Cita.

```
html.q(
  cite: str,
  content,
) → content
```

/6.2.77.1 **cite** `str`

Enlace a la fuente de la cita o a más información sobre la edición.

/6.2.77.2 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.78 **rp** Desde: 0.14.0 Requires the `html` feature

Paréntesis para el texto de una anotación ruby.

```
html.rp(content) → content
```

/6.2.78.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.79 **rt** Desde: 0.14.0 Requires the `html` feature

Texto de anotación ruby.

```
html.rt(content) → content
```

/6.2.79.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.80 **ruby** Desde: 0.14.0 Requires the `html` feature

Anotaciones ruby.

```
html.ruby(content) → content
```

/6.2.80.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.81 **s** Desde: 0.14.0 Requires the `html` feature

Texto inexacto.

```
html.s(content) → content
```

/6.2.81.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.82 **samp** Desde: 0.14.0 Requires the `html` feature

Salida de computadora.

```
html.samp(content) → content
```

/6.2.82.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.83 **script** Desde: 0.14.0 Requires the `html` feature

Script incrustado.

```
html.script(
  async: bool,
  blocking: str array,
  crossorigin: str,
  defer: bool,
  fetchpriority: auto str,
  integrity: str,
```

```
  nomodule: bool,
  referrerpolicy: none str,
  src: str,
  type: str,
  str,
) → content
```

/6.2.83.1 **async** `bool`

Ejecuta el script cuando esté disponible, sin bloquear mientras se lo obtiene.

/6.2.83.2 **blocking** `str` or `array`

Indica si el elemento puede bloquear el renderizado.

| Variante | Detalles |
| --- | --- |
| `"blocking"` | |

/6.2.83.3 **crossorigin** `str`

Cómo maneja el elemento las solicitudes crossorigin.

| Variante | Detalles |
| --- | --- |
| `"anonymous"` | |
| `"use-credentials"` | |

/6.2.83.4 **defer** `bool`

Difiere la ejecución del script.

/6.2.83.5 **fetchpriority** `auto` or `str`

Establece la prioridad de las solicitudes que inicia el elemento.

| Variante | Detalles |
| --- | --- |
| `"high"` | |
| `"low"` | |

/6.2.83.6 **integrity** `str`

Metadatos de integridad que se usan en las comprobaciones de Subresource Integrity.

/6.2.83.7 **nomodule** `bool`

Impide la ejecución en los agentes de usuario que admiten scripts de módulo.

/6.2.83.8 **referrerpolicy** `none` or `str`

Política de referrer para las solicitudes que inicia el elemento.

| Variante | Detalles |
| --- | --- |
| `"no-referrer"` | |
| `"no-referrer-when-downgrade"` | |
| `"same-origin"` | |
| `"origin"` | |
| `"strict-origin"` | |
| `"origin-when-cross-origin"` | |
| `"strict-origin-when-cross-origin"` | |
| `"unsafe-url"` | |

/6.2.83.9 **src** `str`

Dirección del recurso.

/6.2.83.10 **type** `str`

Tipo de script.

| Variante | Detalles |
|---|---|
| `"module"` | |

/6.2.83.11 **body** `str` *Posicional*

El contenido de texto del elemento HTML.

/6.2.84 **search** Desde: 0.14.0 Requires the `html` feature

Contenedor de controles de búsqueda.

```
html.search(content) → content
```

/6.2.84.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.85 **section** Desde: 0.14.0 Requires the `html` feature

Sección genérica de un documento o aplicación.

```
html.section(content) → content
```

/6.2.85.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.86 **select** Desde: 0.14.0 Requires the `html` feature

Control de lista (list box).

```
html.select(
  autocomplete: str array,
  disabled: bool,
  form: str,
  multiple: bool,
  name: str,
  required: bool,
  size: int,
  content,
) → content
```

/6.2.86.1 **autocomplete** `str` or `array`

Pista para la función de autocompletado de formularios.

| Variante | Detalles |
|---|---|
| `"shipping"` | |
| `"billing"` | |
| `"name"` | |
| `"honorific-prefix"` | |
| `"given-name"` | |
| `"additional-name"` | |
| `"family-name"` | |
| `"honorific-suffix"` | |
| `"nickname"` | |
| `"username"` | |
| `"new-password"` | |

| Variante | Detalles |
| --- | --- |
| `"current-password"` | |
| `"one-time-code"` | |
| `"organization-title"` | |
| `"organization"` | |
| `"street-address"` | |
| `"address-line1"` | |
| `"address-line2"` | |
| `"address-line3"` | |
| `"address-level4"` | |
| `"address-level3"` | |
| `"address-level2"` | |
| `"address-level1"` | |
| `"country"` | |
| `"country-name"` | |
| `"postal-code"` | |
| `"cc-name"` | |
| `"cc-given-name"` | |
| `"cc-additional-name"` | |
| `"cc-family-name"` | |
| `"cc-number"` | |
| `"cc-exp"` | |
| `"cc-exp-month"` | |
| `"cc-exp-year"` | |
| `"cc-csc"` | |
| `"cc-type"` | |
| `"transaction-currency"` | |
| `"transaction-amount"` | |
| `"language"` | |
| `"bday"` | |
| `"bday-day"` | |
| `"bday-month"` | |
| `"bday-year"` | |
| `"sex"` | |
| `"url"` | |
| `"photo"` | |
| `"home"` | |
| `"work"` | |
| `"mobile"` | |
| `"fax"` | |
| `"pager"` | |
| `"tel"` | |
| `"tel-country-code"` | |
| `"tel-national"` | |
| `"tel-area-code"` | |
| `"tel-local"` | |
| `"tel-local-prefix"` | |
| `"tel-local-suffix"` | |
| `"tel-extension"` | |
| `"email"` | |

| Variante | Detalles |
| --- | --- |
| `"impp"` | |

/6.2.86.2 **`disabled`** `bool`

Indica si el control de formulario está deshabilitado.

/6.2.86.3 **`form`** `str`

Asocia el elemento con un elemento de formulario.

/6.2.86.4 **`multiple`** `bool`

Indica si se permiten varios valores.

/6.2.86.5 **`name`** `str`

Nombre del elemento que se usa para enviar el formulario y en la API form.elements.

/6.2.86.6 **`required`** `bool`

Indica si el control es obligatorio para enviar el formulario.

/6.2.86.7 **`size`** `int`

Tamaño del control.

/6.2.86.8 **`body`** `content` *Posicional*

El contenido del elemento HTML.

/6.2.87 **`slot`** Desde: 0.14.0 Requires the `html` feature

Slot del shadow tree.

```
html.slot(
  name: str,
  content,
) → content
```

/6.2.87.1 **`name`** `str`

Nombre del slot del shadow tree.

/6.2.87.2 **`body`** `content` *Posicional*

El contenido del elemento HTML.

/6.2.88 **`small`** Desde: 0.14.0 Requires the `html` feature

Comentario al margen.

```
html.small(content) → content
```

/6.2.88.1 **`body`** `content` *Posicional*

El contenido del elemento HTML.

/6.2.89 **`source`** Desde: 0.14.0 Requires the `html` feature

Fuente de la imagen para img, o fuente multimedia para video o audio.

```
html.source(
  height: int,
  media: str,
  sizes: array,
  src: str,
  srcset: array,
  type: str,
```

```
  width: int,
) → content
```

/6.2.89.1 **height** `int`

Dimensión vertical.

/6.2.89.2 **media** `str`

Medios aplicables.

/6.2.89.3 **sizes** `array`

Tamaños de imagen para distintos diseños de página. Espera un array de diccionarios con las claves `condition` (cadena de texto) y `size` (longitud).

/6.2.89.4 **src** `str`

Dirección del recurso.

/6.2.89.5 **srcset** `array`

Imágenes para usar en distintas situaciones, p. ej., pantallas de alta resolución, monitores chicos, etc. Espera un array de diccionarios con las claves `src` (cadena de texto) y `width` (entero) o `density` (flotante).

/6.2.89.6 **type** `str`

Tipo de recurso incrustado.

/6.2.89.7 **width** `int`

Dimensión horizontal.

/6.2.90 **span** Desde: 0.14.0 Requires the `html` feature

Contenedor genérico de contenido en línea (phrasing).

```
html.span(content) → content
```

/6.2.90.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.91 **strong** Desde: 0.14.0 Requires the `html` feature

Importancia.

```
html.strong(content) → content
```

/6.2.91.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.92 **style** Desde: 0.14.0 Requires the `html` feature

Información de estilo incrustada.

```
html.style(
  blocking: str array,
  media: str,
  str,
) → content
```

/6.2.92.1 **blocking** `str` or `array`

Indica si el elemento puede bloquear el renderizado.

| Variante | Detalles |
| --- | --- |
| `"blocking"` | |

/6.2.92.2 **media** `str`

Medios aplicables.

/6.2.92.3 **body** `str` *Posicional*

El contenido de texto del elemento HTML.

/6.2.93 **sub** Desde: 0.14.0 Requires the `html` feature

Subíndice.

```
html.sub(content) → content
```

/6.2.93.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.94 **summary** Desde: 0.14.0 Requires the `html` feature

Epígrafe de los detalles.

```
html.summary(content) → content
```

/6.2.94.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.95 **sup** Desde: 0.14.0 Requires the `html` feature

Superíndice.

```
html.sup(content) → content
```

/6.2.95.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.96 **table** Desde: 0.14.0 Requires the `html` feature

Tabla.

```
html.table(content) → content
```

/6.2.96.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.97 **tbody** Desde: 0.14.0 Requires the `html` feature

Grupo de filas de una tabla.

```
html.tbody(content) → content
```

/6.2.97.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.98 **td** Desde: 0.14.0 Requires the `html` feature

Celda de tabla.

```
html.td(
  colspan: int,
  headers: str array,
```

```
  rowspan: int,
  content,
) → content
```

/6.2.98.1 **colspan** `int`

Cantidad de columnas que debe abarcar la celda.

/6.2.98.2 **headers** `str` or `array`

Las celdas de encabezado de esta celda.

/6.2.98.3 **rowspan** `int`

Cantidad de filas que debe abarcar la celda.

/6.2.98.4 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.99 **template** Desde: 0.14.0 Requires the `html` feature

Plantilla.

```
html.template(
  shadowrootclonable: bool,
  shadowrootcustomelementregistry: bool,
  shadowrootdelegatesfocus: bool,
  shadowrootmode: str,
  shadowrootserializable: bool,
  content,
) → content
```

/6.2.99.1 **shadowrootclonable** `bool`

Establece clonable en un shadow root declarativo.

/6.2.99.2 **shadowrootcustomelementregistry** `bool`

Permite que los shadow roots declarativos indiquen que usarán un registro de elementos personalizado.

/6.2.99.3 **shadowrootdelegatesfocus** `bool`

Establece que delegue el foco en un shadow root declarativo.

/6.2.99.4 **shadowrootmode** `str`

Habilita los shadow roots declarativos en streaming.

| Variante | Detalles |
|---|---|
| `"open"` | |
| `"closed"` | |

/6.2.99.5 **shadowrootserializable** `bool`

Establece serializable en un shadow root declarativo.

/6.2.99.6 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.100 **textarea** Desde: 0.14.0 Requires the `html` feature

Controles de texto multilínea.

```
html.textarea(
  autocomplete: str array,
```

```
  cols: int,
  dirname: str,
  disabled: bool,
  form: str,
  maxlength: int,
  minlength: int,
  name: str,
  placeholder: str,
  readonly: bool,
  required: bool,
  rows: int,
  wrap: str,
  content,
) -> content
```

/6.2.100.1 **autocomplete** `str` or `array`

Pista para la función de autocompletado de formularios.

| Variante | Detalles |
|---|---|
| `"shipping"` | |
| `"billing"` | |
| `"name"` | |
| `"honorific-prefix"` | |
| `"given-name"` | |
| `"additional-name"` | |
| `"family-name"` | |
| `"honorific-suffix"` | |
| `"nickname"` | |
| `"username"` | |
| `"new-password"` | |
| `"current-password"` | |
| `"one-time-code"` | |
| `"organization-title"` | |
| `"organization"` | |
| `"street-address"` | |
| `"address-line1"` | |
| `"address-line2"` | |
| `"address-line3"` | |
| `"address-level4"` | |
| `"address-level3"` | |
| `"address-level2"` | |
| `"address-level1"` | |
| `"country"` | |
| `"country-name"` | |
| `"postal-code"` | |
| `"cc-name"` | |
| `"cc-given-name"` | |
| `"cc-additional-name"` | |
| `"cc-family-name"` | |
| `"cc-number"` | |
| `"cc-exp"` | |

| Variante | Detalles |
|---|---|
| `"cc-exp-month"` | |
| `"cc-exp-year"` | |
| `"cc-csc"` | |
| `"cc-type"` | |
| `"transaction-currency"` | |
| `"transaction-amount"` | |
| `"language"` | |
| `"bday"` | |
| `"bday-day"` | |
| `"bday-month"` | |
| `"bday-year"` | |
| `"sex"` | |
| `"url"` | |
| `"photo"` | |
| `"home"` | |
| `"work"` | |
| `"mobile"` | |
| `"fax"` | |
| `"pager"` | |
| `"tel"` | |
| `"tel-country-code"` | |
| `"tel-national"` | |
| `"tel-area-code"` | |
| `"tel-local"` | |
| `"tel-local-prefix"` | |
| `"tel-local-suffix"` | |
| `"tel-extension"` | |
| `"email"` | |
| `"impp"` | |

/6.2.100.2 **`cols`** `int`

Cantidad máxima de caracteres por línea.

/6.2.100.3 **`dirname`** `str`

Nombre del control de formulario que se usa para enviar la direccionalidad del elemento al enviar el formulario.

/6.2.100.4 **`disabled`** `bool`

Indica si el control de formulario está deshabilitado.

/6.2.100.5 **`form`** `str`

Asocia el elemento con un elemento de formulario.

/6.2.100.6 **`maxlength`** `int`

Longitud máxima del valor.

/6.2.100.7 **`minlength`** `int`

Longitud mínima del valor.

/6.2.100.8 **`name`** `str`

Nombre del elemento que se usa para enviar el formulario y en la API form.elements.

/6.2.100.9 **placeholder** `str`

Etiqueta visible para el usuario que se ubica dentro del control de formulario.

/6.2.100.10 **readonly** `bool`

Indica si el usuario puede editar el valor.

/6.2.100.11 **required** `bool`

Indica si el control es obligatorio para enviar el formulario.

/6.2.100.12 **rows** `int`

Cantidad de líneas que se muestran.

/6.2.100.13 **wrap** `str`

Cómo debe ajustarse el valor del control de formulario al enviar el formulario.

| Variante | Detalles |
| --- | --- |
| `"soft"` | |
| `"hard"` | |

/6.2.100.14 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.101 **tfoot** Desde: 0.14.0 Requires the `html` feature

Grupo de filas de pie de una tabla.

```
html.tfoot(content) → content
```

/6.2.101.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.102 **th** Desde: 0.14.0 Requires the `html` feature

Celda de encabezado de tabla.

```
html.th(
  abbr: str,
  colspan: int,
  headers: str array,
  rowspan: int,
  scope: str,
  content,
) → content
```

/6.2.102.1 **abbr** `str`

Etiqueta alternativa que se usa para la celda de encabezado cuando se la referencia en otros contextos.

/6.2.102.2 **colspan** `int`

Cantidad de columnas que debe abarcar la celda.

/6.2.102.3 **headers** `str` or `array`

Las celdas de encabezado de esta celda.

/6.2.102.4 **rowspan** `int`

Cantidad de filas que debe abarcar la celda.

/6.2.102.5 **scope** `str`

Especifica a qué celdas se aplica la celda de encabezado.

| Variante | Detalles |
|---|---|
| `"row"` | |
| `"col"` | |
| `"rowgroup"` | |
| `"colgroup"` | |

/6.2.102.6 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.103 **thead** Desde: 0.14.0 Requires the `html` feature

Grupo de filas de encabezado de una tabla.

```
html.thead(content) → content
```

/6.2.103.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.104 **time** Desde: 0.14.0 Requires the `html` feature

Equivalente legible por máquinas de datos relacionados con fechas u horas.

```
html.time(
  datetime: datetime duration,
  content,
) → content
```

/6.2.104.1 **datetime** `datetime` or `duration`

Valor legible por máquinas.

/6.2.104.2 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.105 **title** Desde: 0.14.0 Requires the `html` feature

Título del documento.

```
html.title(content) → content
```

/6.2.105.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.106 **tr** Desde: 0.14.0 Requires the `html` feature

Fila de tabla.

```
html.tr(content) → content
```

/6.2.106.1 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.107 **track** Desde: 0.14.0 Requires the `html` feature

Pista de texto con temporización.

```
html.track(
  default: bool,
  kind: str,
  label: str,
```

```
  src: str,
  srclang: str,
) → content
```

/6.2.107.1 **`default`** `bool`

Habilita la pista de texto si no hay otra pista de texto más adecuada.

/6.2.107.2 **`kind`** `str`

El tipo de pista de texto.

| Variante | Detalles |
|---|---|
| `"subtitles"` | |
| `"captions"` | |
| `"descriptions"` | |
| `"chapters"` | |
| `"metadata"` | |

/6.2.107.3 **`label`** `str`

Etiqueta visible para el usuario.

/6.2.107.4 **`src`** `str`

Dirección del recurso.

/6.2.107.5 **`srclang`** `str`

Idioma de la pista de texto.

/6.2.108 **`u`** Desde: 0.14.0 Requires the `html` feature

Anotación no articulada.

```
html.u(content) → content
```

/6.2.108.1 **`body`** `content` *Posicional*

El contenido del elemento HTML.

/6.2.109 **`ul`** Desde: 0.14.0 Requires the `html` feature

Lista.

```
html.ul(content) → content
```

/6.2.109.1 **`body`** `content` *Posicional*

El contenido del elemento HTML.

/6.2.110 **`var`** Desde: 0.14.0 Requires the `html` feature

Variable.

```
html.var(content) → content
```

/6.2.110.1 **`body`** `content` *Posicional*

El contenido del elemento HTML.

/6.2.111 **`video`** Desde: 0.14.0 Requires the `html` feature

Reproductor de video.

```
html.video(
  autoplay: bool,
```

```
  controls: bool,
  crossorigin: str,
  height: int,
  loop: bool,
  muted: bool,
  playsinline: bool,
  poster: str,
  preload: none auto str,
  src: str,
  width: int,
  content,
) → content
```

/6.2.111.1 **autoplay** `bool`

Indica que el recurso multimedia puede iniciarse automáticamente al cargar la página.

/6.2.111.2 **controls** `bool`

Muestra los controles del agente de usuario.

/6.2.111.3 **crossorigin** `str`

Cómo maneja el elemento las solicitudes crossorigin.

| Variante | Detalles |
| --- | --- |
| `"anonymous"` | |
| `"use-credentials"` | |

/6.2.111.4 **height** `int`

Dimensión vertical.

/6.2.111.5 **loop** `bool`

Indica si se repite en bucle el recurso multimedia.

/6.2.111.6 **muted** `bool`

Indica si se silencia el recurso multimedia de forma predeterminada.

/6.2.111.7 **playsinline** `bool`

Recomienda al agente de usuario mostrar el contenido de video dentro del área de reproducción del elemento.

/6.2.111.8 **poster** `str`

Fotograma de póster que se muestra antes de reproducir el video.

/6.2.111.9 **preload** `none` or `auto` or `str`

Sugiere cuánta memoria intermedia necesitará probablemente el recurso multimedia.

| Variante | Detalles |
| --- | --- |
| `"metadata"` | |

/6.2.111.10 **src** `str`

Dirección del recurso.

/6.2.111.11 **width** `int`

Dimensión horizontal.

/6.2.111.12 **body** `content` *Posicional*

El contenido del elemento HTML.

/6.2.112 **wbr** Desde: 0.14.0 Requires the `html` feature

Oportunidad de salto de línea.

```
html.wbr() → content
```

## /6.3 Atributos globales

Estos parámetros son comunes a todas las funciones de HTML tipado. Se enumeran acá una sola vez, en lugar de hacerlo explícitamente en cada elemento, por legibilidad.

## 3.14.4 pdf Elemento Desde: 0.13.0

El formato de exportación a PDF de Typst.

Los archivos PDF se centran en describir documentos visualmente con precisión, pero también tienen facilidades para anotar su estructura. Este enfoque híbrido los hace muy adecuados para el intercambio de documentos: se renderizan exactamente igual en todos los dispositivos, pero también admiten la extracción del contenido y la estructura de un documento (al menos hasta cierto punto). A diferencia de los archivos PNG, los PDF no están atados a una resolución específica. Por lo tanto, podés verlos a cualquier tamaño sin sufrir una pérdida de calidad.

### /1 Exportar como PDF

#### /1.1 Línea de comandos

PDF es el formato de exportación por defecto de Typst. Ejecutar el subcomando `compile` o `watch` sin especificar un formato creará un PDF. Al exportar a PDF, tenés las siguientes opciones de configuración:

- Con qué estándares de PDF[3.14.4/3.2] debe garantizar Typst la conformidad, especificando `--pdf-standard` seguido de uno o varios estándares separados por comas. Los estándares válidos son `1.4`, `1.5`, `1.6`, `1.7`, `2.0`, `a-1b`, `a-1a`, `a-2b`, `a-2u`, `a-2a`, `a-3b`, `a-3u`, `a-3a`, `a-4`, `a-4f`, `a-4e` y `ua-1`. Por defecto, Typst produce archivos conformes con PDF-1.7.
- Podés desactivar por completo el etiquetado del PDF con `--pdf-tagged=false`. Por defecto, Typst siempre escribirá un *PDF etiquetado (Tagged PDF)* para ofrecer un nivel básico de accesibilidad. Con este indicador, podés desactivar las etiquetas. Esto hará que tu archivo sea inaccesible e impedirá la conformidad con los niveles de conformidad accesibles de PDF/A y con todas las partes de PDF/UA.
- Qué páginas exportar, especificando `--pages` seguido de una lista de números separados por comas o de rangos de números separados por guiones. Los rangos pueden ser semiabiertos. Ejemplo: `2,3,7-9,11-`.

#### /1.2 App web

Hacé clic en el botón de descarga rápida de la parte superior derecha para exportar un PDF con la configuración por defecto. Para configurar más, hacé clic en "File" > "Export as" > "PDF" o hacé clic en la flecha hacia abajo junto al botón de descarga rápida y seleccioná "Export as PDF". Al exportar a PDF, tenés las siguientes opciones de configuración:

- Con qué estándares de PDF debe garantizar Typst la conformidad. Por defecto, Typst produce archivos conformes con PDF-1.7. Podés elegir libremente la versión de PDF entre 1.4 y 2.0. Los estándares adicionales válidos son `A-1b`, `A-1a`, `A-2b`, `A-2u`, `A-2a`, `A-3b`. `A-3u`, `A-3a`, `A-4`, `A-4f`, `A-4e` y `UA-1`.
- Qué páginas exportar. Las opciones válidas son "All pages", "Current page" y "Custom ranges". Los rangos personalizados son una lista de números separados por comas o de rangos de números separados por guiones. Los rangos pueden ser semiabiertos. Ejemplo: `2,3,7-9,11-`.

### /2 Funcionalidad específica de PDF

Typst expone la funcionalidad específica de PDF en el elemento global `pdf`. Mirá abajo las definiciones y opciones que contiene.

Este elemento contiene algunas funciones sin una API definitiva. Están diseñadas para mejorar la accesibilidad de documentos con tablas complejas. Esto incluye @[`table-summary`], `header-cell`[3.14.4/7] y `data-cell`[3.14.4/6]. Todas estas funciones se eliminarán en una futura versión de Typst, ya sea integrándolas en las funciones de tablas o eliminándolas por completo. Podés habilitar estas funciones pasando `--features a11y-extras` o estableciendo la variable de entorno `TYPST_FEATURES` en `a11y-extras`. En la app web, estas características no están disponibles por el momento.

### /3 Parámetros

```
pdf(
  pages: none int array dictionary,
```

```
  standard: str array,
  tagged: auto bool,
  pretty: bool,
) → content
```

/3.1 **pages** none or int or array or dictionary *Configurable con set* Por defecto: none

Especifica qué rangos de páginas se deben incluir en el PDF. Puede ser:

- none para exportar todas las páginas.
- Un único rango de páginas.
- Un array de rangos de páginas.

Un rango de páginas es uno de los siguientes:

- Un número que especifica exactamente una página (los números de página empiezan en uno).
- Un diccionario con las claves `from` (que indica el inicio de un rango de páginas) y/o `to` (que indica el final de un rango de páginas). Debe estar presente al menos una de las claves y tanto el número inicial como el final son inclusivos. Por lo tanto, el rango `(from: 1, to: 3)` es la lista de páginas `(1, 2, 3)`.

Especificar una única página o rango de páginas:

```
#set pdf(pages: 3)
#set pdf(pages: (from: 3))
#set pdf(pages: (to: 5))
#set pdf(pages: (from: 3, to: 5))
```

Especificar varias páginas, rangos de páginas o una mezcla:

```
#set pdf(pages: (5, 7, 9))
#set pdf(pages: ((to: 3), 5, (from: 7, to: 8), (from: 10)))
```

/3.2 **standard** str or array *Configurable con set* Por defecto: `("1.7",)`

Una lista de estándares de PDF con los que Typst garantizará la conformidad. Puede ser un único estándar o un array de estándares.

La Organización Internacional de Normalización (ISO) publicó el estándar PDF base y varios estándares que lo extienden para que los PDF sean más adecuados para casos de uso específicos. Por defecto, Typst exporta archivos PDF 1.7. Adobe Acrobat 8 y posteriores, así como todos los demás visores de PDF de uso común, son compatibles con esta versión de PDF.

Algunas características de Typst pueden no estar disponibles según el estándar de PDF que elijas. Podés combinar estándares PDF/A y PDF/UA compatibles. En particular, PDF/UA-1 y PDF/A-4 son mutuamente incompatibles, ya que el primero requiere PDF 1.7 o anterior y el segundo requiere PDF 2.0.

/3.2.1 **Versiones de PDF**

Typst admite cinco versiones distintas de PDF: 1.4, 1.5, 1.6, 1.7 (por defecto) y 2.0. Podés elegir cualquiera de estas versiones para la exportación de tu documento. Sin embargo, según las características que hayas usado puede haber una versión mínima. Del mismo modo, los estándares a los que apuntes pueden limitar qué versiones podés elegir (mirá más abajo para más información).

Esta es una lista de cómo cada versión nueva mejora respecto de PDF 1.4 para los documentos de Typst:

- **PDF 1.5** (2003): Mejor gestión del color, extracción de texto, accesibilidad de tablas, reflujo y fuentes de emojis
- **PDF 1.6** (2004): Enlaces más flexibles
- **PDF 1.7** (2006): Permite archivos adjuntos[3.14.4/5], mejor reflujo
- **PDF 2.0** (2017): Mejores metadatos y semántica de etiquetas para la accesibilidad

El software que se use para leer tu archivo debe admitir tu versión de PDF. En circunstancias normales, esto no plantea ningún problema, pero puede ser una fuente de errores al trabajar con hardware más antiguo. Para el intercambio general, recomendamos mantener la configuración por defecto de PDF 1.7 o elegir PDF 2.0.

Al usar archivos PDF como imágenes[3.11.7] en tu documento, la versión de PDF de exportación debe igualar o superar las versiones de los archivos de imagen.

### /3.2.2 PDF/UA

Typst admite escribir archivos conformes con PDF/UA. Los archivos PDF/UA están diseñados para el *acceso universal*[4.4.1]. Cuando elegís este estándar de PDF, Typst ejecutará comprobaciones adicionales al exportar tu documento. Estas comprobaciones se asegurarán de que estés siguiendo las mejores prácticas de accesibilidad. Por ejemplo, se asegurará de que todas tus imágenes vengan con descripciones alternativas.

Tené en cuenta que hay algunas reglas en PDF/UA que son cruciales para la accesibilidad pero que no se pueden comprobar automáticamente. Por lo tanto, al exportar un documento PDF/UA-1, asegurate de haber hecho lo siguiente:

- Si tu documento está escrito en un idioma distinto del inglés, asegurate de establecer el idioma del texto[3.7.12/2.16] antes de cualquier contenido.
- Asegurate de usar los elementos semánticos de Typst (como los títulos[3.6.9], las figuras[3.6.7] y las listas[3.6.2]) cuando corresponda, en lugar de definir construcciones personalizadas. Esto le permite a Typst saber (y exportar) el rol que cumple una construcción en el documento. Mirá la guía de accesibilidad[4.4] para más detalles.
- No uses exclusivamente el contraste, el color, el formato o el diseño para comunicar una idea. Usá texto o descripciones alternativas en lugar de estos elementos, o además de ellos.
- Envolvé en `pdf.artifact`[3.14.4/4] todos los elementos decorativos sin significado semántico.
- No uses imágenes de texto. En cambio, insertá el texto directamente en tu marcado.

Actualmente, Typst solo admite la parte uno (PDF/UA-1), que se basa en PDF 1.7 (2006). Al exportar a PDF/UA-1, tené en cuenta que vas a tener que proporcionar manualmente descripciones alternativas de la matemática[3.8.8] en lenguaje natural.

Se agregaron nuevas características de accesibilidad a PDF 2.0 (2017). Cuando se establece la exportación a PDF 2.0, Typst aprovechará algunas de estas características. Sin embargo, PDF 2.0 y PDF/UA-1 son mutuamente incompatibles. Para documentos accesibles, actualmente recomendamos exportar a PDF/UA-1 en lugar de PDF 2.0, por las comprobaciones adicionales y la mayor compatibilidad. La segunda parte de PDF/UA está diseñada para PDF 2.0, pero Typst todavía no la admite.

### /3.2.3 PDF/A

Typst admite opcionalmente emitir archivos conformes con PDF/A. Los archivos PDF/A están orientados a la máxima compatibilidad con las herramientas de PDF actuales y futuras. No dependen de características difíciles de implementar o propietarias y contienen metadatos exhaustivos. Esto los hace adecuados para el archivado a largo plazo.

El estándar PDF/A tiene varias versiones (*partes* en la terminología de ISO) y la mayoría de las partes tienen varios perfiles que indican el nivel de conformidad del archivo. Podés apuntar a una parte y a un nivel de conformidad a la vez. Actualmente, Typst admite estos perfiles de salida PDF/A:

- **PDF/A-1b:** El nivel de conformidad *básico* de ISO 19005-1. Esta versión de PDF/A se basa en PDF 1.4 (2001) y produce archivos PDF autocontenidos y archivables. A diferencia de las partes posteriores del estándar PDF/A, no se permite la transparencia en los archivos PDF/A-1.
- **PDF/A-1a:** Este es el nivel de conformidad *accesible* que se construye sobre el nivel básico PDF/A-1b. Para cumplir con este nivel, tu archivo debe ser un *PDF etiquetado* accesible. Tené en cuenta que la accesibilidad mejora en las partes posteriores, ya que no todas las características de accesibilidad de PDF están disponibles para archivos PDF 1.4. Además, todo el texto del archivo debe consistir en puntos de código Unicode conocidos.
- **PDF/A-2b:** El nivel de conformidad *básico* de ISO 19005-2. Esta versión de PDF/A se basa en PDF 1.7 (2006) y produce archivos PDF autocontenidos y archivables.
- **PDF/A-2u:** Este es el nivel de conformidad *asignable a Unicode* que se construye sobre el nivel básico A-2b. También agrega reglas según las cuales todo el texto del documento debe consistir en puntos de código Unicode conocidos. Si es posible, preferí siempre este estándar en lugar de PDF/A-2b.
- **PDF/A-2a:** Este es el nivel de conformidad *accesible* que se construye sobre el nivel asignable a Unicode A-2u. Este nivel de conformidad también agrega dos requisitos: tu archivo debe ser un *PDF etiquetado*

accesible. Typst agrega automáticamente etiquetas para ayudarte a alcanzar este nivel de conformidad. Prestá atención también a las secciones de accesibilidad a lo largo de la referencia y a la guía de accesibilidad[4.4] cuando apuntes a este nivel de conformidad. Por último, PDF/A-2a te prohíbe usar puntos de código del área de uso privado de Unicode[A]. Si querés construir un archivo accesible, considerá también apuntar adicionalmente a PDF/UA-1, que habilita más comprobaciones de accesibilidad automáticas.

- **PDF/A-3b:** El nivel de conformidad *básico* de ISO 19005-3. Esta versión de PDF/A se basa en PDF 1.7 (2006) y produce archivos PDF archivables que pueden contener otros archivos relacionados arbitrarios como archivos adjuntos[3.14.4/5]. La única diferencia con PDF/A-2b es la capacidad de adjuntar archivos que no cumplen con PDF/A.
- **PDF/A-3u:** Este es el nivel de conformidad *asignable a Unicode* que se construye sobre el nivel básico A-3b. Igual que PDF/A-2b, requiere que todo el texto consista en puntos de código Unicode conocidos. Estas reglas no se aplican a los archivos adjuntos. Si es posible, preferí siempre este estándar en lugar de PDF/A-3b.
- **PDF/A-3a:** Este es el nivel de conformidad *accesible* que se construye sobre el nivel asignable a Unicode A-3u. Igual que PDF/A-2a, requiere que los archivos sean un *PDF etiquetado* accesible y que no usen caracteres del área de uso privado de Unicode. Igual que antes, estas reglas no se aplican a los archivos adjuntos.
- **PDF/A-4:** El nivel de conformidad básico de ISO 19005-4. Esta versión de PDF/A se basa en PDF 2.0 (2017) y produce archivos PDF autocontenidos y archivables. PDF/A-4 no tiene partes relacionadas con la accesibilidad. En cambio, el tema se elaboró más en el estándar PDF/UA dedicado. Los archivos PDF/A-4 pueden cumplir con PDF/UA-2 (actualmente no admitido en Typst).
- **PDF/A-4f:** El nivel de conformidad de *archivos incrustados* que se construye sobre el nivel básico A-4. Los archivos que cumplen con este nivel pueden contener otros archivos relacionados arbitrarios como archivos adjuntos[3.14.4/5], igual que los archivos que cumplen con la parte 3 de ISO 19005. La única diferencia con PDF/A-4 es la capacidad de adjuntar archivos que no cumplen con PDF/A.
- **PDF/A-4e:** El nivel de conformidad de *ingeniería* que se construye sobre el nivel de archivos incrustados A-4f. Los archivos que cumplen con este nivel pueden contener objetos 3D. Typst no admite contenido 3D, por lo que, desde la perspectiva de Typst, esto es funcionalmente equivalente a PDF/A-4f.

Si querés apuntar a PDF/A pero no estás seguro de qué configuración particular usar, hay algunas buenas reglas empíricas. Primero, tenés que determinar tu **parte,** que es el "número de versión" del estándar. Hacete estas preguntas:

1. **¿Necesita leer mi archivo software o equipamiento anterior a 2006?** Si es así, elegí la parte uno (PDF/A-1).
2. **Si no, ¿necesita mi archivo archivos adjuntos[3.14.4/5]?** Si es así, tenés que elegir la parte tres (PDF/A-3) o el nivel de archivos incrustados de la parte cuatro (PDF/A-4f).
3. **¿Necesita tu archivo usar características introducidas en PDF 2.0?** Actualmente, el uso de características de PDF 2.0 en Typst se limita a mejoras menores en accesibilidad, por ejemplo para el elemento title[3.6.21]. Si podés prescindir de estas mejoras, maximizá la compatibilidad eligiendo la parte dos (cuando no necesitás adjuntos) o la parte tres (cuando sí necesitás adjuntos). Si dependés de características de PDF 2.0, usá la parte cuatro.

Ahora solo necesitás elegir un **nivel de conformidad** (la letra minúscula del final).

- **Si elegiste la parte uno, dos o tres,** normalmente deberías elegir el nivel de conformidad accesible de tu parte, indicado con una `a` minúscula al final. Si tu documento es inherentemente inaccesible, por ejemplo el portafolio de un artista que no se puede reducir a descripciones alternativas, elegí en cambio el nivel de conformidad `u`. Solo si esto produce un error del compilador, por ejemplo porque usaste puntos de código del área de uso privado de Unicode, usá el nivel básico `b`.
- **Si elegiste la parte cuatro,** deberías elegir el nivel de conformidad básico (PDF/A-4), salvo que necesites incrustar archivos.

Al elegir entre exportar PDF/A y PDF normal, tené en cuenta que los archivos PDF/A contienen metadatos adicionales y que algunos lectores impedirán que el usuario modifique un archivo PDF/A.

[A] https://en.wikipedia.org/wiki/Private_Use_Areas

| Variante | Detalles |
|---|---|
| `"1.4"` | PDF 1.4. |
| `"1.5"` | PDF 1.5. |
| `"1.6"` | PDF 1.6. |
| `"1.7"` | PDF 1.7. |
| `"2.0"` | PDF 2.0. |
| `"a-1b"` | PDF/A-1b. |
| `"a-1a"` | PDF/A-1a. |
| `"a-2b"` | PDF/A-2b. |
| `"a-2u"` | PDF/A-2u. |
| `"a-2a"` | PDF/A-2a. |
| `"a-3b"` | PDF/A-3b. |
| `"a-3u"` | PDF/A-3u. |
| `"a-3a"` | PDF/A-3a. |
| `"a-4"` | PDF/A-4. |
| `"a-4f"` | PDF/A-4f. |
| `"a-4e"` | PDF/A-4e. |
| `"ua-1"` | PDF/UA-1. |

/3.3 **tagged** `auto` or `bool` *Configurable con set* Por defecto: `auto`

Si se produce un documento PDF etiquetado.

El etiquetado está habilitado por defecto para ofrecer un nivel básico de accesibilidad. Se puede desactivar manualmente, por ejemplo para reducir el tamaño del documento, y se desactivará automáticamente al exportar un rango de páginas específico.

/3.4 **pretty** `bool` *Configurable con set* Por defecto: `false`

Si se da formato legible al documento PDF producido.

Esto da formato a la salida de una manera más legible para las personas, pero menos eficiente en espacio.

## Definiciones

3.14.4/4 Artifact Marks content as a PDF artifact. 577
3.14.4/5 Attach A file that will be attached to the output PDF. 579
3.14.4/6 Data Cell Explicitly defines this cell as a data cell. 581
3.14.4/7 Header Cell Explicitly defines a cell as a header cell. 583
3.14.4/8 Table Summary A summary of the purpose and structure of a complex table. 585

## /4 `artifact` Elemento Desde: 0.14.0

Marca contenido como artefacto de PDF.

Los artefactos son partes del documento que no están pensadas para ser leídas por tecnologías de asistencia (TA), como los lectores de pantalla. Algunos ejemplos típicos son las imágenes puramente decorativas que no contribuyen al significado del documento, las marcas de agua o el contenido repetido, como los números de página.

Typst marcará automáticamente cierto contenido, como los encabezados, pies de página, fondos y primeros planos de las páginas, como artefactos. Del mismo modo, las rutas y las formas se marcan automáticamente como artefactos, pero su contenido no. Los números de línea creados con `par.line`[3.6.14/5.1] se marcan automáticamente como artefactos, al igual que las repeticiones de encabezados y pies de tabla.

Una vez que algo se marca como artefacto, no podés volver a hacer accesible ninguna parte de su contenido. Si necesitás marcar como artefacto solo una parte de algo, quizás tengas que usar esta función varias veces.

Si no estás seguro de qué constituye un artefacto, consultá la guía de accesibilidad[4.4.5].

En el futuro, esta función podría moverse fuera del módulo `pdf`, lo que permitiría ocultar contenido a las TA en la exportación a HTML.

### /4.1 Parámetros

```
pdf.artifact(
  kind: str,
  content,
) → content
```

#### /4.1.1 `kind` `str` *Configurable con set* Por defecto: `"other"`

El tipo de artefacto.

Podés mejorar la accesibilidad usando el tipo de artefacto más específico disponible. Tu elección determinará cómo trata el lector de PDF al artefacto durante el reflujo y la extracción de contenido (por ejemplo, copiar y pegar).

Los tipos de artefacto se introdujeron en varias especificaciones de PDF distintas. Según la versión de PDF que apuntes, Typst seleccionará el tipo de artefacto más apropiado a partir de tu selección acá.

| Variante | Detalles |
|---|---|
| `"header"` | Se repite en la parte superior de cada página. |
| `"footer"` | Se repite al pie de cada página. |
| `"watermark"` | Texto o gráficos en el fondo o en el primer plano de todas las páginas. |
| `"page-number"` | Números de página. |
| `"line-number"` | Números de línea o de párrafo. |
| `"redaction"` | Marcadores de posición para áreas en las que había contenido en otra versión del documento y que desde entonces se eliminó. |
| `"bates"` | Numeración Bates. |
| `"page"` | No forma parte del documento, sino de la página en la que se imprime. Un ejemplo son las marcas de corte o las barras de color. |
| `"pagination-other"` | Artefactos que surgen de paginar el documento y que no están cubiertos por otros tipos de artefacto. Esta categoría se aplica, en general, si el artefacto no aparecería en tu documento si fuera un sitio web. Si tu artefacto está cubierto por otras categorías, preferí esas categorías. |
| `"layout"` | Contenido puramente cosmético o adornos tipográficos que no contribuyen al contenido del documento. |
| `"background"` | Fondo de una página o de un elemento gráfico. Este tipo de artefacto se agregó en PDF 1.7. Sin embargo, por requisitos de la especificación de PDF 1.7 que las |

| Variante | Detalles |
| --- | --- |
| | especificaciones posteriores eliminaron, Typst solo usa este tipo de artefacto en PDF 2.0. Si lo usás en PDF 1.7 o anterior, Typst usará en su lugar el tipo `"other"`. |
| `"other"` | Otros artefactos. |

/4.1.2 **body** `content` *Obligatorio* *Posicional*

El contenido que es un artefacto.

## /5 attach Elemento Desde: 0.14.0

Un archivo que se adjuntará al PDF de salida.

Esto se puede usar para distribuir dentro del PDF archivos adicionales asociados a él. Los lectores de PDF mostrarán los archivos en una lista de archivos.

Algunos estándares internacionales usan este mecanismo para adjuntar datos legibles por máquinas (por ejemplo, ZUGFeRD/Factur-X para facturas) que reflejan el contenido visual del PDF.

### /5.1 Ejemplo

```
#pdf.attach(
  "experiment.csv",
  relationship: "supplement",
  mime-type: "text/csv",
  description: "Raw Oxygen readings from the Arctic experiment",
)
```

### /5.2 Notas

- Este elemento se ignora al exportar a un formato distinto de PDF.
- Actualmente no se admiten los archivos adjuntos en PDF/A-2, aunque el archivo adjunto cumpla con PDF/A-1 o PDF/A-2.

### /5.3 Parámetros

```
pdf.attach(
  str path ,
  bytes ,
  relationship: none str ,
  mime-type: none str ,
  description: none str ,
) → content
```

#### /5.3.1 path str or path *Obligatorio* *Posicional*

La ruta del archivo que se adjunta.

Siempre se debe especificar, pero solo se lee de ella si no se proporcionan datos en el siguiente argumento.

#### /5.3.2 data bytes *Obligatorio* *Posicional*

Datos sin procesar del archivo, opcionales.

Si se omite, los datos se leen de la ruta especificada.

#### /5.3.3 relationship none or str *Configurable con set* Por defecto: none

La relación del archivo adjunto con el documento.

Se ignora si la exportación no apunta a PDF/A-3.

| Variante | Detalles |
|---|---|
| `"source"` | El documento PDF se creó a partir del archivo fuente. |
| `"data"` | El archivo se usó para derivar una presentación visual en el PDF. |
| `"alternative"` | Una representación alternativa del documento. |
| `"supplement"` | Recursos adicionales para el documento. |

#### /5.3.4 mime-type none or str *Configurable con set* Por defecto: none

El tipo MIME del archivo adjunto.

/5.3.5 **description** none or str *Configurable con set* Por defecto: none

Una descripción del archivo adjunto.

/6 **data-cell** Desde: 0.14.0 Requires the a11y-extras feature

Define explícitamente esta celda como una celda de datos.

Cada celda de una tabla es una celda de encabezado o una celda de datos. Por defecto, todas las celdas de `table.header`[3.6.19/4.4] son celdas de encabezado y todas las demás celdas son celdas de datos.

Si tu encabezado contiene una celda que no es una celda de encabezado, podés usar esta función para marcarla como celda de datos.

La API de esta característica es temporal. Por lo tanto, llamar a esta función requiere por el momento habilitar el indicador de característica `a11y-extras`. En una futura versión de Typst, esta funcionalidad podría salir del módulo `pdf` para que las tablas de otros destinos de exportación puedan contener la misma información.

```
#show table.cell.where(x: 0): set text(weight: "bold")
#show table.cell.where(x: 1): set text(style: "italic")
#show table.cell.where(x: 1, y: 0): set text(style: "normal")

#table(
  columns: 3,
  align: (left, left, center),

  table.header[Objective][Key Result][Status],

  table.header(
    level: 2,
    table.cell(colspan: 2)[Improve Customer Satisfaction],
    // Status is data for this objective, not a header
    pdf.data-cell[✓ On Track],
  ),
  [], [Increase NPS to 50+], [45],
  [], [Reduce churn to \<5%], [4.2%],

  table.header(
    level: 2,
    table.cell(colspan: 2)[Grow Revenue],
    pdf.data-cell[⚠ At Risk],
  ),
  [], [Achieve \$2M ARR], [\$1.8M],
  [], [Close 50 enterprise deals], [38],
)
```

| **Objective** | Key Result | Status |
|---|---|---|
| **Improve Customer Satisfaction** | | ✓ On Track |
| | *Increase NPS to 50+* | 45 |
| | *Reduce churn to <5%* | 4.2% |
| **Grow Revenue** | | ⚠ At Risk |
| | *Achieve $2M ARR* | $1.8M |
| | *Close 50 enterprise deals* | 38 |

## /6.1 Parámetros

```
pdf.data-cell(content) → content
```

### /6.1.1 **cell** `content` *Obligatorio* *Posicional*

La celda de la tabla.

Puede ser contenido o una llamada a `table.cell`[3.6.19/4.1].

### /7 `header-cell` Desde: 0.14.0 Requires the `a11y-extras` feature

Define explícitamente una celda como celda de encabezado.

Las celdas de encabezado ayudan a los usuarios de tecnologías de asistencia (TA) a entender y navegar tablas complejas. Cuando tu tabla está correctamente marcada con celdas de encabezado, las TA pueden anunciar la información de encabezado relevante a demanda al entrar en una celda.

Por defecto, Typst marcará automáticamente todas las celdas de `table.header`[3.6.19/4.4] como celdas de encabezado. Se aplicarán a las columnas que tienen debajo. Podés usar el parámetro `level`[3.6.19/4.4.2] de esa función para que las celdas de encabezado estén etiquetadas por otras celdas de encabezado.

La función `pdf.header-cell` te permite indicar que una celda es una celda de encabezado en las siguientes situaciones adicionales:

- Tenés una **columna de encabezado** en la que cada celda se aplica a su fila. En ese caso, pasás `"row"` como argumento del parámetro `scope`[3.14.4/7.1.2] para indicar que la celda de encabezado se aplica a la fila.
- Tenés una celda en `table.header`[3.6.19/4.4], por ejemplo al comienzo, que etiqueta tanto su fila como su columna. En ese caso, pasás `"both"` como argumento del parámetro `scope`[3.14.4/7.1.2].
- Tenés una celda de encabezado en una fila que no contiene otras celdas de encabezado. En ese caso, podés usar esta función para marcarla como celda de encabezado.

La API de esta característica es temporal. Por lo tanto, llamar a esta función requiere por el momento habilitar el indicador de característica `a11y-extras`. En una futura versión de Typst, esta funcionalidad podría salir del módulo `pdf` para que las tablas de otros destinos de exportación puedan contener la misma información.

```
#show table.cell.where(x: 0): set text(weight: "medium")
#show table.cell.where(y: 0): set text(weight: "bold")

#table(
  columns: 3,
  align: (start, end, end),

  table.header(
    // Top-left cell: Labels both the nutrient rows
    // and the serving size columns.
    pdf.header-cell(scope: "both")[Nutrient],
    [Per 100g],
    [Per Serving],
  ),

  // First column cells are row headers
  pdf.header-cell(scope: "row")[Calories],
  [250 kcal], [375 kcal],
  pdf.header-cell(scope: "row")[Protein],
  [8g], [12g],
  pdf.header-cell(scope: "row")[Fat],
  [12g], [18g],
  pdf.header-cell(scope: "row")[Carbs],
  [30g], [45g],
)
```

| Nutrient | Per 100g | Per Serving |
|---|---|---|
| Calories | 250 kcal | 375 kcal |
| Protein | 8g | 12g |
| Fat | 12g | 18g |
| Carbs | 30g | 45g |

## /7.1 Parámetros

```
pdf.header-cell(
  level: int,
  scope: str,
  content,
) → content
```

### /7.1.1 **level** `int` Por defecto: 1

El nivel de anidamiento de esta celda de encabezado.

### /7.1.2 **scope** `str` Por defecto: `"column"`

A qué pista de la tabla se aplica esta celda de encabezado.

| Variante | Detalles |
|---|---|
| `"both"` | La celda de encabezado se refiere tanto a la fila como a la columna. |
| `"column"` | La celda de encabezado se refiere a la columna. |
| `"row"` | La celda de encabezado se refiere a la fila. |

### /7.1.3 **cell** `content` *Obligatorio* *Posicional*

La celda de la tabla.

Puede ser contenido o una llamada a `table.cell`[3.6.19/4.1].

## /8 `table-summary` Desde: 0.14.0 Requires the `a11y-extras` feature

Un resumen del propósito y la estructura de una tabla compleja.

Estará disponible para las tecnologías de asistencia (TA), como los lectores de pantalla, al exportar a PDF, pero no para los lectores videntes de tu archivo.

Este campo está pensado para instrucciones que ayuden al usuario a navegar la tabla con TA. No es una descripción alternativa, así que no dupliques adentro el contenido de la tabla. Del mismo modo, no lo uses para la conclusión central de la tabla. En cambio, incluila en el texto que rodea la tabla o, mejor aún, en un epígrafe de figura[3.6.7/6.1].

Si tenés dudas sobre si tu tabla es lo bastante compleja como para justificar un resumen, inclinate por no incluir uno. Si estás seguro de que tu tabla es lo bastante compleja, considerá si un usuario vidente podría encontrarla difícil. Podría beneficiarse de las instrucciones que pongas acá, así que considerá imprimirlas visiblemente en el documento.

La API de esta característica es temporal. Por lo tanto, llamar a esta función requiere por el momento habilitar el indicador de característica `a11y-extras`. Aunque esta funcionalidad llegue a estar disponible sin un indicador de característica en el futuro, el resumen seguirá siendo exclusivo de la exportación a PDF.

```
#figure(
  pdf.table-summary(
    // The summary just provides orientation and structural
    // information for AT users.
    summary: "The first two columns list the names of each participant. The last
column contains cells spanning multiple rows for their assigned group.",
    table(
      columns: 3,
      table.header[First Name][Given Name][Group],
      [Mike], [Davis], table.cell(rowspan: 3)[Sales],
      [Anna], [Smith],
      [John], [Johnson],
      [Sara], [Wilkins], table.cell(rowspan: 2)[Operations],
      [Tom], [Brown],
    ),
  ),
  // This is the key takeaway of the table, so we put it in the caption.
  caption: [The Sales org now has a new member],
)
```

| First Name | Given Name | Group |
|---|---|---|
| Mike | Davis | Sales |
| Anna | Smith | |
| John | Johnson | |
| Sara | Wilkins | Operations |
| Tom | Brown | |

Tabla 1: The Sales org now has a new member

### /8.1 Parámetros

```
pdf.table-summary(
  summary: str,
  content,
) → content
```

/8.1.1 **summary** `str`

/8.1.2 **table** `content` *Obligatorio* *Posicional*

La tabla.

### 3.14.5 png Elemento Desde: unreleased

El formato de exportación a PNG de Typst.

En lugar de crear un PDF, Typst también puede renderizar páginas directamente a gráficos ráster PNG. Los PNG son imágenes comprimidas sin pérdida que pueden contener una página a la vez. Al exportar un documento de varias páginas, Typst emitirá varios PNG. Los PNG son una buena opción cuando querés usar la salida de Typst en un software de edición de imágenes o cuando no podés usar ninguno de los otros formatos de exportación de Typst.

A diferencia de los otros formatos de exportación de Typst, los PNG están atados a una resolución específica. Al exportar a PNG, podés configurar la resolución como píxeles por pulgada (PPI). Si el medio en el que ves el PNG tiene una resolución más fina que el PNG que exportaste, vas a notar una pérdida de calidad. Typst calcula la resolución de tus PNG según las dimensiones físicas de cada página y el PPI. Si necesitás una guía para elegir un valor de PPI, considerá lo siguiente:

- Un valor de 300 o 600 es típico para la impresión de escritorio.
- Las impresiones profesionales de gráficos detallados pueden llegar hasta 1200 PPI.
- Si tu documento solo se ve a distancia, por ejemplo un póster, podés elegir un valor menor que 300.
- Si tu documento se ve en pantallas, un valor de PPI típico para un smartphone es 400-500.

Como los PNG solo contienen una trama de píxeles, el texto que contienen no se puede extraer automáticamente (sin OCR), por ejemplo copiando y pegando o con un lector de pantalla. Si necesitás que el texto sea accesible, exportá en cambio un archivo PDF o HTML.

Los PNG pueden tener fondos transparentes. Por defecto, Typst producirá un PNG con un fondo blanco opaco. Podés hacer transparente el fondo con `#set page(fill: none)`. Conocé más en la página de referencia de la función page[3.10.17/4.9].

#### /1 Exportar como PNG

##### /1.1 Línea de comandos

Pasá `--format png` al subcomando `compile` o `watch`, o proporcioná un nombre de archivo de salida que termine en `.png`.

Si tu documento tiene más de una página, Typst creará varios archivos de imagen. El nombre del archivo de salida debe ser entonces una cadena de plantilla que contenga al menos uno de estos
- `{p}`, que se reemplazará por el número de página
- `{0p}`, que se reemplazará por el número de página rellenado con ceros (para que todos los números tengan la misma longitud)
- `{t}`, que se reemplazará por la cantidad total de páginas

Al exportar a PNG, tenés las siguientes opciones de configuración:

- Con qué resolución renderizar, especificando `--ppi` seguido de una cantidad de píxeles por pulgada. El valor por defecto es 144.
- Qué páginas exportar, especificando `--pages` seguido de una lista de números separados por comas o de rangos de números separados por guiones. Los rangos pueden ser semiabiertos. Ejemplo: `2,3,7-9,11-`.

##### /1.2 App web

Hacé clic en "File" > "Export as" > "PNG" o hacé clic en la flecha hacia abajo junto al botón de descarga rápida y seleccioná "Export as PNG". Al exportar a PNG, tenés las siguientes opciones de configuración:

- La resolución con la que se deben renderizar las páginas, como una cantidad de píxeles por pulgada. El valor por defecto es 144.
- Qué páginas exportar. Las opciones válidas son "All pages", "Current page" y "Custom ranges". Los rangos personalizados son una lista de números separados por comas o de rangos de números separados por guiones. Los rangos pueden ser semiabiertos. Ejemplo: `2,3,7-9,11-`.

## /2 Parámetros

```
format.png(ppi: float) → content
```

### /2.1 ppi `float` *Configurable con set*

Por defecto: `144.0`

El PPI (píxeles por pulgada) que se usa para la exportación a PNG.

### 3.14.6 `svg` Elemento Desde: unreleased

El formato de exportación a SVG de Typst.

En lugar de crear un PDF, Typst también puede renderizar páginas directamente a gráficos vectoriales escalables (SVG), que son el formato preferido para incrustar gráficos vectoriales en páginas web. Igual que los archivos PDF, los SVG muestran tu documento exactamente como lo dispusiste en Typst. Del mismo modo, comparten el beneficio de no estar atados a una resolución específica. Por lo tanto, podés imprimir o ver archivos SVG en cualquier dispositivo sin sufrir una pérdida de calidad. (Tené en cuenta que la calidad de impresión de las fuentes puede ser mejor con un PDF.) A diferencia de un PDF, un SVG no puede contener varias páginas. Al exportar un documento de varias páginas, Typst emitirá varios SVG.

Los SVG pueden representar el texto de dos maneras: incrustando el propio texto y renderizándolo con las fuentes disponibles en la computadora del visor, o incrustando las formas de cada glifo de la fuente usada para crear el documento. Para asegurar que el archivo SVG se vea igual en todos los dispositivos en los que se lo vea, Typst elige el segundo método. Esto significa que el texto del SVG no se puede extraer automáticamente, por ejemplo copiando y pegando o con un lector de pantalla. Si necesitás que el texto sea accesible, exportá en cambio un archivo PDF o HTML.

Los SVG pueden tener fondos transparentes. Por defecto, Typst producirá un SVG con un fondo blanco opaco. Podés hacer transparente el fondo con `#set page(fill: none)`. Conocé más en la página de referencia de la función page[3.10.17/4.9].

#### /1 Exportar como SVG

##### /1.1 Línea de comandos

Pasá `--format svg` al subcomando `compile` o `watch`, o proporcioná un nombre de archivo de salida que termine en `.svg`.

Si tu documento tiene más de una página, Typst creará varios archivos de imagen. El nombre del archivo de salida debe ser entonces una cadena de plantilla que contenga al menos uno de estos
- `{p}`, que se reemplazará por el número de página
- `{0p}`, que se reemplazará por el número de página rellenado con ceros (para que todos los números tengan la misma longitud)
- `{t}`, que se reemplazará por la cantidad total de páginas

Al exportar a SVG, tenés las siguientes opciones de configuración:

- Qué páginas exportar, especificando `--pages` seguido de una lista de números separados por comas o de rangos de números separados por guiones. Los rangos pueden ser semiabiertos. Ejemplo: `2,3,7-9,11-`.

##### /1.2 App web

Hacé clic en "File" > "Export as" > "SVG" o hacé clic en la flecha hacia abajo junto al botón de descarga rápida y seleccioná "Export as SVG". Al exportar a SVG, tenés las siguientes opciones de configuración:

- Qué páginas exportar. Las opciones válidas son "All pages", "Current page" y "Custom ranges". Los rangos personalizados son una lista de números separados por comas o de rangos de números separados por guiones. Los rangos pueden ser semiabiertos. Ejemplo: `2,3,7-9,11-`.

#### /2 Parámetros

```
format.svg(pretty: bool) → content
```

##### /2.1 `pretty` `bool` *Configurable con set* Por defecto: `false`

Si se da formato legible al archivo SVG producido.

Esto da formato a la salida de una manera más legible para las personas, pero menos eficiente en espacio.

# 4 Guías

¡Te damos la bienvenida a la sección de guías! Acá vas a encontrar material útil para grupos de usuarios o casos de uso específicos. Mirá la lista de abajo para ver las guías disponibles. ¡No dudes en proponer otros temas para guías!

## Lista de guías

4.1 Guía para usuarios de LaTeX ..... 591
4.2 Guía de configuración de página ..... 601
4.3 Guía de tablas ..... 612
4.4 Guía de accesibilidad ..... 639

## 4.1 Guía para usuarios de LaTeX

Esta página es un buen punto de partida si ya usaste LaTeX y querés probar Typst. Vamos a explorar las principales diferencias entre estos dos sistemas desde la perspectiva del usuario. Aunque Typst no está construido sobre LaTeX y tiene una sintaxis distinta, vas a aprender a aprovechar tus conocimientos de LaTeX para arrancar con ventaja.

Igual que LaTeX, Typst es un sistema de composición tipográfica basado en marcado: componés tu documento en un archivo de texto y lo marcás con comandos y otra sintaxis. Después usás un compilador para componer el archivo fuente en un PDF. Sin embargo, Typst también se diferencia de LaTeX en varios aspectos: por un lado, Typst usa una sintaxis más dedicada (como la que quizás conozcas de Markdown) para las tareas comunes. Los comandos de Typst también son más coherentes: todos funcionan de la misma manera, así que, a diferencia de LaTeX, solo necesitás entender unos pocos conceptos generales en lugar de aprender convenciones distintas para cada paquete. Además, Typst compila más rápido que LaTeX: la compilación suele tardar milisegundos, no segundos, por lo que tanto la app web como el compilador pueden ofrecer vistas previas instantáneas.

A continuación, vamos a cubrir algunas de las preguntas más comunes que tendrá un usuario que pasa de LaTeX al componer un documento en Typst. Si preferís una introducción paso a paso a Typst, mirá nuestro tutorial[2].

### 4.1.1 Instalación

Tenés dos formas de usar Typst: en nuestra app web[A] o instalando el compilador[B] en tu computadora. Cuando usás la app web, ofrecemos un editor colaborativo con todo incluido y ejecutamos Typst en tu navegador, sin necesidad de instalación.

Si en cambio elegís usar Typst en tu computadora, podés descargar el compilador como un único binario pequeño que cualquier usuario puede ejecutar, sin necesidad de privilegios de administrador. A diferencia de las distribuciones populares de LaTeX, como TeX Live, los paquetes se descargan la primera vez que los usás y después se guardan en caché localmente, lo que mantiene liviana tu instalación de Typst. Con el compilador local, podés usar tu propio editor y decidir dónde guardar tus archivos.

### 4.1.2 ¿Cómo creo un documento nuevo y vacío?

Es fácil. Simplemente creás un archivo de texto nuevo y vacío (la extensión del archivo es `.typ`). No hace falta ningún código base para empezar. Simplemente empezá escribiendo tu texto. Se compondrá en una página A4 vacía. Si usás la app web, hacé clic en "+ Empty document" para crear un proyecto nuevo con un archivo y entrar al editor. Los saltos de párrafo[3.6.15] funcionan igual que en LaTeX: simplemente usá una línea en blanco.

```
Hey there!

Here are two paragraphs. The
output is shown to the right.
```

Hey there!

Here are two paragraphs. The output is shown to the right.

Si en cambio querés partir de un documento LaTeX preexistente, podés usar Pandoc[C] para convertir tu código fuente a marcado de Typst. Esta conversión también está integrada en nuestra app web, así que podés subir tu archivo `.tex` para empezar tu proyecto en Typst.

[A] https://typst.app/signup/
[B] https://github.com/typst/typst/releases
[C] https://pandoc.org

### 4.1.3 ¿Cómo creo títulos de sección, énfasis, ...?

LaTeX usa el comando `\section` para crear un título de sección. Los títulos anidados se indican con `\subsection`, `\subsubsection`, etcétera. Según tu clase de documento, también existe `\part` o `\chapter`.

En Typst, los títulos[3.6.9] son menos verbosos: le ponés como prefijo a la línea del título un signo igual y un espacio para obtener un título de primer orden: **`= Introduction`**. Si necesitás un título de segundo orden, usás dos signos igual: **`== In this paper`**. Podés anidar títulos tan profundamente como quieras agregando más signos igual.

El énfasis (que normalmente se renderiza como texto en cursiva) se expresa encerrando el texto entre `_underscores_` y el énfasis fuerte (que normalmente se renderiza en negrita) usando en cambio **`*stars*`**.

Esta es una lista de comandos de marcado comunes usados en LaTeX y sus equivalentes en Typst. También podés consultar la hoja de referencia completa de la sintaxis[3.1].

| Elemento | LaTeX | Typst | Ver |
|---|---|---|---|
| Énfasis fuerte | `\textbf{strong}` | `*strong*` | `strong`[3.6.18] |
| Énfasis | `\emph{emphasis}` | `_emphasis_` | `emph`[3.6.6] |
| Enlace | `\url{https://typst.app}` | `https://typst.app/` | `link`[3.6.10] |
| Etiqueta | `\label{intro}` | `<intro>` | `label`[3.5.18] |
| Referencia | `\ref{intro}` | `@intro` | `ref`[3.6.17] |
| Cita | `\cite{humphrey97}` | `@humphrey97` | `cite`[3.6.3] |
| Monoespaciado (máquina de escribir) | `\texttt{mono}` | funciones `text` o `mono` | `text`[3.7.12], `mono`[3.8.27/1.4] |
| Código | entorno `lstlisting` | `` `print(f"{x}")` `` | `raw`[3.7.6] |
| Literal | entorno `verbatim` | `` `#typst-code()` `` | `raw`[3.7.6] |
| Lista con viñetas | entorno `itemize` | `- List` | `list`[3.6.2] |
| Lista numerada | entorno `enumerate` | `+ List` | `enum`[3.6.11] |
| Lista de términos | entorno `description` | `/ Term: List` | `terms`[3.6.20] |
| Figura | entorno `figure` | función `figure` | `figure`[3.6.7] |
| Tabla | entorno `table` | función `table` | `table`[3.6.19] |
| Ecuación | $x$, entornos `align` / `equation` | `$x$`, `$ x = y $` | `equation`[3.8.16] |

Las listas[3.6.2] no dependen de entornos en Typst. En cambio, tienen una sintaxis liviana como los títulos. Para crear una lista sin orden (`itemize`), poné como prefijo de cada línea de un elemento un guion:

````
To write this list in Typst...

```latex
\begin{itemize}
  \item Fast
  \item Flexible
  \item Intuitive
\end{itemize}
```

...just type this:

- Fast
- Flexible
- Intuitive
````

To write this list in Typst...

```latex
\begin{itemize}
  \item Fast
  \item Flexible
  \item Intuitive
\end{itemize}
```

...just type this:

- Fast
- Flexible
- Intuitive

Anidar listas funciona simplemente usando la sangría adecuada. Agregar una línea en blanco entre los elementos da como resultado una lista con un espaciado más amplio[3.6.2/3.1].

Para obtener en cambio una lista numerada[3.6.11] (`enumerate`), usá un + en lugar del guion. Para una lista de términos[3.6.20] (`description`), escribí `/ Term: Description`.

Tené en cuenta que la función `raw`[3.7.6] y su sintaxis (por ejemplo, `` `raw` ``) solo funcionan para texto literal (sin formato). Si necesitás formato, podés usar en cambio la función `text`[3.7.12] con una fuente monoespaciada, como en el siguiente ejemplo:

```
#text(
  font: "DejaVu Sans Mono",
  size: 0.8em,
)[monospace *bold*]
```

monospace **bold**

### 4.1.4 ¿Cómo uso un comando?

LaTeX depende mucho de los comandos (con prefijo de barra invertida). Usa estas *macros* para afectar el proceso de composición y para insertar y manipular contenido. Algunos comandos aceptan argumentos, que con mayor frecuencia van entre llaves: `\cite{rasmus}`.

Typst distingue entre el modo marcado y el modo código[3.3.2]. El modo por defecto es el modo marcado, donde componés texto y aplicás construcciones sintácticas como `*stars for bold text*`. El modo código, en cambio, es paralelo a los lenguajes de programación como Python y ofrece la opción de ingresar y ejecutar segmentos de código.

Dentro del marcado de Typst, podés cambiar al modo código para un único comando (o, mejor dicho, *expresión*) usando un numeral (#). Así es como llamás a funciones para, por ejemplo, dividir tu proyecto en distintos archivos[3.3.8] o renderizar texto según alguna condición[3.3.4]. Dentro del modo código, es posible incluir *contenido*[3.5.8] de marcado normal usando corchetes. Dentro del modo código, este contenido se trata igual que cualquier otro valor normal de una variable.

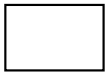
```
First, a rectangle:
#rect()

Let me show how to do
#underline([_underlined_ text])

We can also do some maths:
#calc.max(3, 2 * 4)

And finally a little loop:
#for x in range(3) [
  Hi #x.
]
```

First, a rectangle:



Let me show how to do *underlined* text

We can also do some maths: 8

And finally a little loop: Hi 0. Hi 1. Hi 2.

Una llamada a función siempre involucra el nombre de la función (`rect`[3.11.10], `underline`[3.7.13], `calc.max`[3.5.9/2.34], `range`[3.5.2/3.13]) seguido de paréntesis (a diferencia de LaTeX, donde los corchetes y las llaves son opcionales si la macro no requiere argumentos). La lista de argumentos esperada que se pasa dentro de esos paréntesis depende de la función concreta y está especificada en la referencia[3].

#### /1 Argumentos

Una función puede tener varios argumentos. Algunos argumentos son posicionales, es decir, simplemente proporcionás el valor: la función `#lower("SCREAM")` devuelve su argumento en minúsculas. Muchas funciones usan argumentos con nombre en lugar de argumentos posicionales para aumentar la legibilidad. Por ejemplo, las dimensiones y el trazo de un rectángulo se definen con argumentos con nombre:

```
#rect(
  width: 2cm,
  height: 1cm,
  stroke: red,
)
```



Un argumento con nombre se especifica ingresando primero su nombre (arriba, `width`, `height` y `stroke`), después dos puntos y, a continuación, el valor (`2cm`, `1cm`, `red`). Podés encontrar los argumentos con nombre disponibles en la página de referencia[3] de cada función o en el panel de autocompletado al escribir. Los argumentos con nombre son similares a cómo se configuran algunos entornos de LaTeX; por ejemplo, escribirías `\begin{enumerate}[label={\alph*)}]` para empezar una lista con las etiquetas `a)`, `b)`, etcétera.

A menudo querés proporcionarle algo de contenido[3.5.8] a una función. Por ejemplo, el comando de LaTeX `\underline{Alternative A}` se traduciría en Typst como `#underline([Alternative A])`. Los corchetes indican que un valor es contenido[3.5.8]. Dentro de estos corchetes, podés usar marcado normal. Sin embargo, son muchos paréntesis para una construcción bastante simple. Por eso también podés mover los argumentos de contenido finales después de los paréntesis (y omitir los paréntesis si quedarían vacíos).

```
Typst is an #underline[alternative]
to LaTeX.

#rect(fill: aqua)[Get started here!]
```




## /2 Tipos de datos

Probablemente ya notaste que los argumentos tienen tipos de datos distintivos. Typst admite muchos tipos de datos[3.5.31]. A continuación hay una tabla con algunos de los más importantes y cómo escribirlos. ¡Para especificar valores de cualquiera de estos tipos, tenés que estar en modo código!

| Tipo de dato | Ejemplo |
|---|---|
| Content[3.5.8] | `[*fast* typesetting]` |
| String[3.5.28] | `"Pietro S. Author"` |
| Integer[3.5.17] | `23` |
| Floating point number[3.5.15] | `1.459` |
| Absolute length[3.10.13] | `12pt`, `5in`, `0.3cm`, … |
| Relative length[3.10.20] | `65%` |

La diferencia entre contenido y cadena de texto es que el contenido puede contener marcado, incluidas llamadas a funciones, mientras que una cadena de texto es realmente solo una secuencia simple de caracteres.

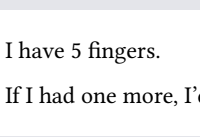
Typst ofrece construcciones de control de flujo[3.3.4] y operadores[3.3.10] como `+` para sumar cosas o `==` para comprobar la igualdad entre dos variables.

También podés guardar valores, incluidas funciones, en tus propias variables[3.3.3]. Esto puede ser útil para realizar cálculos con ellos, crear automatizaciones reutilizables o hacer referencia a un valor varias veces. El enlace de variables se logra con la palabra clave let, que funciona de forma similar a `\newcommand`:

```
// Store the integer `5`.
#let five = 5

// Define a function that
// increments a value.
#let inc(i) = i + 1

// Reference the variables.
I have #five fingers.
```



```
If I had one more, I'd have
#inc(five) fingers. Whoa!
```

### /3 Comandos que afectan al resto del documento

En LaTeX, algunos comandos como `\textbf{bold text}` reciben un argumento entre llaves y solo lo afectan a él. Otros comandos, como `\bfseries bold text`, actúan como interruptores (LaTeX los llama declaraciones) y alteran el aspecto de todo el contenido posterior dentro del documento o del ámbito actual.

En Typst, la misma función se puede usar tanto para afectar el aspecto del resto del documento, de un bloque (o ámbito) o solo de sus argumentos. Por ejemplo, `#text(weight: "bold")[bold text]` solo pondrá en negrita su argumento, mientras que `#set text(weight: "bold")` pondrá en negrita todo el texto hasta el final del bloque actual o hasta el final del documento, si no hay ninguno. Los efectos de una función quedan de inmediato en evidencia según se use en una llamada o en una regla set.[3.2.1]

```
I am starting out with small text.

#set text(14pt)

This is a bit #text(18pt)[larger,]
don't you think?
```

I am starting out with small text.

This is a bit larger, don't you think?

Las reglas set pueden aparecer en cualquier parte del documento. Se pueden pensar como valores por defecto de los argumentos de su función respectiva:

```
#set enum(numbering: "I.")

Good results can only be obtained by
+ following best practices
+ being aware of current results
  of other researchers
+ checking the data for biases
```

Good results can only be obtained by

I. following best practices
II. being aware of current results of other researchers
III. checking the data for biases

El + es azúcar sintáctico (pensalo como una abreviatura) para una llamada a la función enum[3.6.11], a la que arriba le aplicamos una regla set. La mayor parte de la sintaxis está vinculada de esta manera a una función.[3.1] Si necesitás dar estilo a un elemento más allá de lo que permiten sus argumentos, podés redefinir por completo su aspecto con una regla show[3.2.2] (algo comparable a `\renewcommand`).

Podés lograr los efectos de comandos de LaTeX como `\textbf`, `\textsf`, `\rmfamily`, `\mdseries` e `\itshape` con los argumentos font[3.7.12/2.1], style[3.7.12/2.3] y weight[3.7.12/2.4] de la función `text`. La función text se puede usar en una regla set (estilo declaración) o con un argumento de contenido. Para reemplazar `\textsc`, podés usar la función smallcaps[3.7.7], que renderiza su argumento de contenido en versalitas. Si querés usarla al estilo declaración (como `\scshape`), podés usar una regla show de *todo*[3.2.2] que aplique la función al resto del ámbito:

```
#show: smallcaps

Boisterous Accusations
```

Boisterous Accusations

## 4.1.5 ¿Cómo cargo una clase de documento?

En LaTeX, empezás tu archivo `.tex` principal con el comando `\documentclass{article}` para definir cómo debe verse tu documento. En ese comando, quizás hayas reemplazado `article` por otro valor, como `report` o `amsart`, para seleccionar otro aspecto.

Al usar Typst, das estilo a tus documentos con funciones[3.5.16]. Normalmente usás una plantilla que ofrece una función que da estilo a todo tu documento. Primero importás la función desde un archivo de plantilla. Después la aplicás a todo tu documento. Esto se logra con una regla show[3.2.2] que envuelve el documento siguiente en una función dada. El siguiente ejemplo ilustra cómo funciona:

```
#import "conf.typ": conf
#show: conf.with(
  title: [
    Towards Improved Modelling
  ],
  authors: (
    (
      name: "Theresa Tungsten",
      affiliation: "Artos Institute",
      email: "tung@artos.edu",
    ),
    (
      name: "Eugene Deklan",
      affiliation: "Honduras State",
      email: "e.deklan@hstate.hn",
    ),
  ),
  abstract: lorem(80),
)

Let's get started writing this
article by putting insightful
paragraphs right here!
```


Towards Improved Modelling

Towards Improved Modelling

Theresa Tungsten
Artos Institute
tung@artos.edu

Eugene Deklan
Honduras State
e.deklan@hstate.hn

**Abstract**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et.

Let's get started writing this article by putting insightful paragraphs right here! Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae sine metu degendae praesidia firmissima. – Filium morte multavit. – Si sine causa, nollem me ab eo delectari, quod ista Platonis, Aristoteli, Theophrasti orationis ornamenta neglexerit. Nam illud quidem physici, credere aliquid esse minimum, quod profecto numquam putavisset, si a Polyaeno, familiari suo, geometrica discere maluisset quam illum etiam ipsum dedocere. Sol Democrito magnus videtur, quippe homini erudito in geometriaque perfecto, huic pedalis fortasse; tantum enim esse omnino in nostris poetis aut inertissimae segnitiae est aut fastidii delicatissimi. Mihi quidem videtur, inermis ac nudus est. Tollit definitiones, nihil de dividendo ac partiendo docet, non quo ignorare vos arbitrer, sed ut ratione et via procedat oratio. Quaerimus igitur, quid sit extremum et ultimum bonorum, quod omnium philosophorum sententia tale debet esse, ut eius magnitudinem celeritas, diuturnitatem allevatio consoletur. Ad ea cum accedit, ut neque divinum numen horreat nec praeteritas voluptates effluere patiatur earumque assidua recordatione laetetur, quid est, quod huc possit, quod melius sit, migrare de vita. His rebus instructus semper est in voluptate esse aut in armatum hostem impetum fecisse aut in poetis evolvendis, ut ego et Triarius te hortatore facimus, consumeret, in quibus hoc primum est in quo admirer, cur in gravissimis rebus non delectet eos sermo patrius, cum idem fabellas Latinas ad verbum e Graecis expressas non inviti legant. Quis enim tam inimicus paene nomini Romano est, qui Ennii Medeam aut Antiopam Pacuvii spernat aut reiciat, quod se isdem Euripidis fabulis delectari dicat, Latinas litteras oderit? Synephebos ego, inquit, potius Caecilii aut Andriam

1

La sentencia `import`[3.3.8] pone a disposición funciones[3.5.16] (y otras definiciones) de otro archivo. En este ejemplo, importa la función `conf` del archivo `conf.typ`. Esta función da formato a un documento como un artículo de conferencia. Usamos una regla show para aplicarla al documento y también configurar algunos metadatos del artículo. Después de aplicar la regla show, ¡podemos empezar a escribir nuestro artículo de inmediato!

También podés usar plantillas de Typst Universe (el equivalente de CTAN en Typst) con una sentencia de importación como esta: `#import "@preview/elsearticle:0.2.1": elsearticle`. Consultá la documentación de cada plantilla para conocer el nombre de su función de plantilla. Las plantillas y los paquetes de Typst Universe se descargan automáticamente la primera vez que los usás.

En la app web, podés elegir crear un proyecto a partir de una plantilla de Typst Universe o incluso crear la tuya con el asistente de plantillas. De forma local, podés usar la CLI `typst init` para crear un proyecto nuevo a partir de una plantilla. Mirá la lista de plantillas[A] publicadas en Typst Universe. También podés echarle un vistazo al repositorio `awesome-typst`[B] para encontrar plantillas de la comunidad que no están disponibles a través de Universe.

También podés crear tus propias plantillas personalizadas.[2.4] Son más cortas y más legibles que los archivos `.sty` de LaTeX correspondientes por órdenes de magnitud, ¡así que probalo!

INFO

Las funciones son los "comandos" de Typst y pueden transformar sus argumentos en un valor de salida, incluido *contenido* del documento. Las funciones son "puras", lo que significa que no pueden tener ningún efecto más allá de crear un valor / contenido de salida. Esto contrasta marcadamente con las macros de LaTeX, que pueden tener efectos arbitrarios sobre tu documento.

Para que una función dé estilo a todo tu documento, la regla show procesa todo lo que viene después y llama a la función especificada después de los dos puntos con el resultado como argumento. La

[A] https://typst.app/universe/search/?kind=templates
[B] https://github.com/qjcg/awesome-typst

parte `.with` es un *método* que toma la función `conf` y preconfigura algunos de sus argumentos antes de pasársela a la regla show.

### 4.1.6 ¿Cómo cargo paquetes?

Typst viene con "todo incluido", así que el equivalente de muchos paquetes populares de LaTeX está integrado. A continuación armamos una tabla con los paquetes que se cargan con frecuencia y sus funciones correspondientes de Typst.

| Paquete de LaTeX | Alternativa en Typst |
|---|---|
| graphicx, svg | función `image`[3.11.7] |
| tabularx, tabularray | funciones `table`[3.6.19], `grid`[3.10.10] |
| fontenc, inputenc, unicode-math | ¡Simplemente empezá a escribir! |
| babel, polyglossia | función `text`[3.7.12/2.16]: `#set text(lang: "zh")` |
| amsmath | Modo matemático[3.8] |
| amsfonts, amssymb | módulo `sym`[3.9] y sintaxis[3.1.3] |
| geometry, fancyhdr | función `page`[3.10.17] |
| xcolor | función `text`[3.7.12/2.7]: `#set text(fill: rgb("#0178A4"))` |
| hyperref | función `link`[3.6.10] |
| bibtex, biblatex, natbib | funciones `cite`[3.6.3], `bibliography`[3.6.1] |
| lstlisting, minted | función y sintaxis `raw`[3.7.6] |
| parskip | funciones `block`[3.10.4/2.9] y `par`[3.6.14/4.6] |
| csquotes | Establecé el idioma en `text`[3.7.12/2.16] y escribí `"` o `'` |
| caption | función `figure`[3.6.7] |
| enumitem | funciones `list`[3.6.2], `enum`[3.6.11], `terms`[3.6.20] |
| nicefrac | propiedad `frac.style`[3.8.19/3.3] |

Aunque *muchas* cosas están integradas, no todo puede estarlo. Por eso Typst tiene su propio ecosistema de paquetes[A], donde la comunidad comparte sus creaciones y automatizaciones. Tomemos, por ejemplo, el paquete *CeTZ*: este paquete te permite crear dibujos y gráficos complejos. Para usar CeTZ en tu documento, simplemente podés escribir:

```
#import "@preview/cetz:0.4.1"
```

(El `@preview` es un *espacio de nombres* que se usa mientras el gestor de paquetes todavía está en un estado temprano y experimental. Se reemplazará en el futuro.)

Además del centro oficial de paquetes, también podés echarle un vistazo al repositorio awesome-typst[B], que reúne una lista curada de recursos creados para Typst.

Si necesitás cargar funciones y variables de otro archivo de tu proyecto, por ejemplo para usar una plantilla, podés usar la misma sentencia `import`[3.3.8] con un nombre de archivo en lugar de una especificación de paquete. Para en cambio incluir el contenido textual de otro archivo, podés usar una sentencia `include`[3.3.8]. Obtendrá el contenido del archivo especificado y lo pondrá en tu documento.

### 4.1.7 ¿Cómo escribo matemática?

Para entrar en el modo matemático en Typst, simplemente encerrá tu ecuación entre signos de pesos. Podés entrar en el modo de visualización (display) agregando espacios o saltos de línea entre el contenido de la ecuación y los signos de pesos que la encierran.

[A] https://typst.app/universe
[B] https://github.com/qjcg/awesome-typst

```
The sum of the numbers from
$1$ to $n$ is:

$ sum_(k=1)^n k = (n(n+1))/2 $
```

The sum of the numbers from 1 to $n$ is:

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

El modo matemático[3.8] funciona de manera distinta que el marcado normal o el modo código. Los números y los caracteres sueltos se muestran literalmente, mientras que varios caracteres consecutivos (que no sean números) se interpretarán como variables de Typst.

Typst predefine muchas variables útiles en modo matemático. Todas las letras griegas (`alpha`, `beta`, ...) y algunas hebreas (`aleph`, `beth`, ...) están disponibles por su nombre. Algunos símbolos también están disponibles mediante atajos, como `<=`, `>=` y `->`.

Consultá las páginas de símbolos[3.9] para ver la lista completa de símbolos. Si falta un símbolo, también podés acceder a él mediante una secuencia de escape Unicode[3.1.6].

Las formas alternativas y relacionadas de los símbolos a menudo se pueden seleccionar agregando un modificador[3.5.29] después de un punto. Por ejemplo, `arrow.l.squiggly` inserta una flecha ondulada que apunta hacia la izquierda. Si en cambio querés insertar texto de varias letras en tu expresión, encerralo entre comillas dobles:

```
$ delta "if" x <= 5 $
```

$$\delta \text{ if } x \leq 5$$

En Typst, los delimitadores se escalan automáticamente según sus expresiones, igual que si se insertaran implícitamente los comandos `\left` y `\right` en LaTeX. Podés personalizar el comportamiento de los delimitadores con la función `lr`[3.8.20/2.1]. Para evitar que un par de delimitadores se escale, podés escaparlos con barras invertidas.

Typst compondrá automáticamente como fracción los términos alrededor de una barra `/`, respetando la precedencia de operadores. Todos los paréntesis redondos que la fracción no vuelva redundantes aparecerán en la salida. Las fracciones se componen verticalmente, salvo que se personalicen con `frac.style`[3.8.19/3.3]. También podés producir una barra tal cual escapándola con una barra invertida (`\/`).

```
$ f(x) = (x + 1) / x $
```

$$f(x) = \frac{x+1}{x}$$

Los subíndices y superíndices[3.8.10/3.1] funcionan de manera similar en Typst y en LaTeX. `$x^2$` produce un superíndice y `$x_2$` da un subíndice. Si querés incluir más de un valor en un subíndice o superíndice, encerrá su contenido entre paréntesis: `$x_(a -> epsilon)$`.

Como las variables en modo matemático no necesitan un # como prefijo (ni una \ como en LaTeX), también podés llamar a funciones sin estos caracteres especiales:

```
$ f(x, y) := cases(
  1 "if" (x dot y)/2 <= 0,
  2 "if" x "is even",
  3 "if" x in NN,
  4 "else",
) $
```

$$f(x,y) := \begin{cases} 1 \text{ if } \frac{x \cdot y}{2} \leq 0 \\ 2 \text{ if } x \text{ is even} \\ 3 \text{ if } x \in \mathbb{N} \\ 4 \text{ else} \end{cases}$$

El ejemplo anterior usa la función cases[3.8.13] para describir f. Dentro de la función cases, los argumentos se delimitan con comas y los argumentos también se interpretan como matemática. Si necesitás en cambio interpretar los argumentos como valores de Typst, ponéles un # como prefijo:

```
$ (a + b)^2
  = a^2
  + text(fill: #maroon, 2 a b)
  + b^2 $
```

$$(a + b)^2 = a^2 + 2ab + b^2$$

Podés usar todas las funciones de Typst dentro del modo matemático e insertar cualquier contenido. Si querés que funcionen normalmente, con el modo código en la lista de argumentos, podés poner un # como prefijo de su llamada. Ya nadie puede impedirte usar rectángulos o emojis como tus variables:

```
$ sum^10_(🤓=1)
  #rect(width: 4mm, height: 2mm)/🤓
  = 🧠 maltese $
```

$$\sum_{🤓=1}^{10} \frac{▭}{🤓} = 🧠✠$$

Si querés ingresar tus símbolos matemáticos directamente como Unicode, ¡eso también es posible!

Las llamadas matemáticas pueden tener listas de argumentos bidimensionales usando ; como delimitador. El uso más común es la función mat[3.8.21], que crea matrices:

```
$ mat(
  1, 2, ..., 10;
  2, 2, ..., 10;
  dots.v, dots.v, dots.down, dots.v;
  10, 10, ..., 10;
) $
```

$$\begin{pmatrix} 1 & 2 & \dots & 10 \\ 2 & 2 & \dots & 10 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 10 & \dots & 10 \end{pmatrix}$$

### 4.1.8 ¿Cómo consigo el “aspecto LaTeX”?

Los trabajos compuestos en LaTeX tienen un aspecto inconfundible. Esto se debe sobre todo a su fuente, Computer Modern, a la justificación, al interlineado ajustado y a los márgenes amplios.

El siguiente ejemplo
- establece márgenes[3.10.17/4.5] amplios
- habilita la justificación[3.6.14/4.3], las líneas más juntas[3.6.14/4.1] y la sangría de primera línea[3.6.14/4.6]
- establece la fuente[3.7.12/2.1] “New Computer Modern”, un derivado OpenType de Computer Modern, tanto para el texto como para los bloques de código[3.7.6]
- reduce el peso de la fuente[3.7.12/2.4] en el modo matemático
- desactiva el espaciado[3.10.4/2.9] de párrafo
- aumenta el espaciado[3.10.4/2.9] alrededor de los títulos[3.6.9]

```
#set page(margin: 1.75in)
#set par(leading: 0.55em, spacing: 0.55em, first-line-indent: 1.8em, justify: true)
#set text(font: "New Computer Modern")
#show raw: set text(font: "New Computer Modern Mono")
#show math.equation: set text(weight: "regular")
#show heading: set block(above: 1.4em, below: 1em)
```

¡Este debería ser un buen punto de partida! Si querés ir más allá, ¿por qué no crear una plantilla reutilizable?

### 4.1.9 Bibliografías

Typst incluye un sistema de bibliografía completo que es compatible con archivos BibTeX. Podés seguir usando tus bibliotecas de literatura `.bib` cargándolas con la función `bibliography`[3.6.1]. Otra posibilidad es usar el formato nativo de Typst basado en YAML[A].

Typst usa Citation Style Language para definir y procesar los estilos de citas y bibliografía. Podés comparar los archivos CSL con los archivos `.bbx` de BibLaTeX. El compilador ya incluye más de 80 estilos de citas[3.6.1/4.4], pero podés usar cualquier estilo compatible con CSL del repositorio de CSL[B] o escribir el tuyo.

Podés citar una entrada de tu bibliografía o hacer referencia a una etiqueta de tu documento con la misma sintaxis: `@key` (esto haría referencia a una entrada llamada `key`). Como alternativa, podés usar la función `cite`[3.6.3].

Las formas alternativas de tu cita, como solo el año y las citas para uso natural en prosa (cf. `\citet` y `\textcite`), están disponibles con `#cite(<key>, form: "prose")`[3.6.3/3.3].

Podés encontrar más información en la página de documentación de la función `bibliography`[3.6.1].

### 4.1.10 ¿Qué limitaciones tiene Typst actualmente en comparación con LaTeX?

Aunque hoy Typst puede reemplazar a LaTeX para muchas personas, todavía hay características que Typst (aún) no admite. Esta es una lista de ellas que, cuando corresponde, contiene posibles soluciones alternativas.

- **Un ecosistema de gráficos bien establecido.** Los usuarios de LaTeX a menudo crean gráficos elaborados junto con sus documentos en PGF/TikZ. El ecosistema de Typst todavía no ofrece la misma amplitud de opciones disponibles, pero el ecosistema alrededor del paquete `cetz`[C] se está poniendo al día rápidamente.
- **Cambiar los márgenes de página sin un salto de página.** En LaTeX, los márgenes siempre se pueden ajustar, incluso sin un salto de página. Para cambiar los márgenes en Typst, usás la función `page`[3.10.17], que fuerza un salto de página. Si solo querés que unos pocos párrafos se extiendan hacia los márgenes y luego volver a los márgenes anteriores, podés usar la función `pad`[3.10.16] con relleno negativo.

[A] https://github.com/typst/hayagriva/blob/main/docs/file-format.md
[B] https://github.com/citation-style-language/styles
[C] https://typst.app/universe/package/cetz

## 4.2 Guía de configuración de página

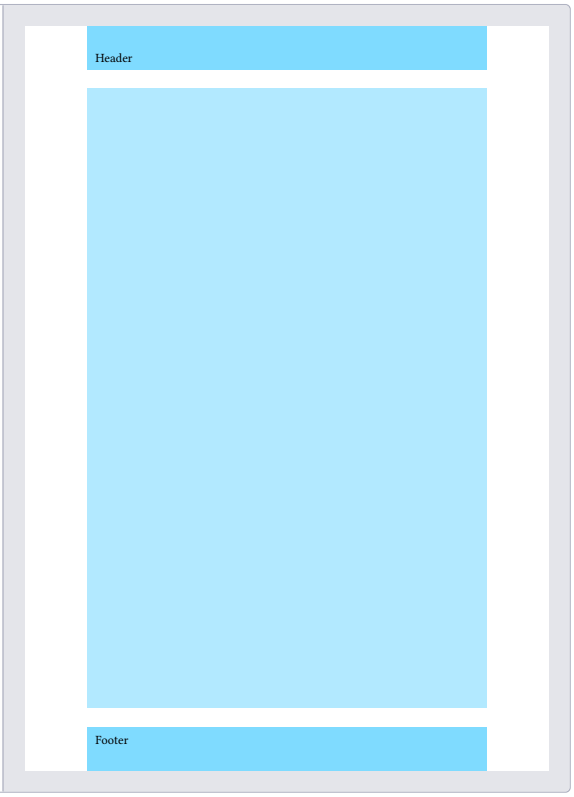
La configuración de tu página es una gran parte de la primera impresión que da tu documento. La longitud de las líneas, los márgenes y las columnas influyen en la apariencia[A] y en la legibilidad[B], mientras que los encabezados y pies de página adecuados ayudarán a tu lector a navegar fácilmente por el documento. Esta guía te va a ayudar a personalizar las páginas, los márgenes, los encabezados, los pies de página y los números de página para que se ajusten bien a tu contenido y puedas ponerte a escribir.

En Typst, cada página tiene un ancho, un alto y márgenes en los cuatro lados. Los márgenes superior e inferior pueden contener un encabezado y un pie de página. La regla set del elemento page[3.10.17] es donde controlás toda la configuración de la página. Si hacés cambios con esta regla set, Typst se asegurará de que haya después una página vacía nueva y conforme, por lo que puede insertar un salto de página. Por lo tanto, lo mejor es especificar tu regla set de page[3.10.17] al comienzo de tu documento o en tu plantilla.

```
#set rect(
  width: 100%,
  height: 100%,
  inset: 4pt,
)

#set page(
  paper: "iso-b7",
  header: rect(fill: aqua)[Header],
  footer: rect(fill: aqua)[Footer],
  number-align: center,
)

#rect(fill: aqua.lighten(40%))
```




Este ejemplo visualiza las dimensiones del contenido de la página, los encabezados y los pies de página. El contenido de la página es el tamaño de la página (ISO B7) menos el margen por defecto de cada lado. En los márgenes superior e inferior hay rectángulos con trazo que visualizan el encabezado y el pie de página. No tocan el contenido principal, sino que están desplazados el 30 % del margen respectivo. Podés controlar este desplazamiento especificando los argumentos `header-ascent`[3.10.17/4.14] y `footer-descent`[3.10.17/4.16].

A continuación, la guía va a entrar en más detalle sobre cómo cumplir con requisitos comunes de configuración de página, con ejemplos.

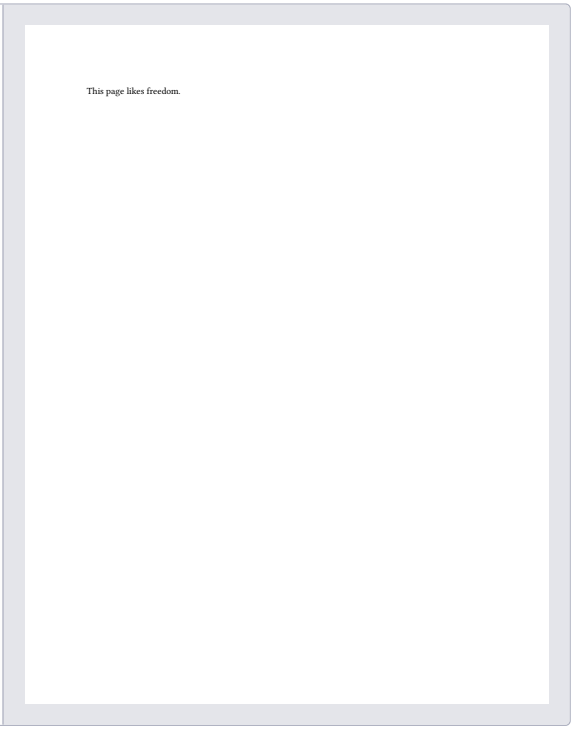
### 4.2.1 Personalizar el tamaño de página y los márgenes

El tamaño de página por defecto de Typst es papel A4. Según tu región y tu caso de uso, vas a querer cambiarlo. Podés hacerlo con la regla set de page[3.10.17], pasándole un argumento de tipo cadena de texto para usar un tamaño de página común. Las opciones incluyen la serie ISO 216 completa (por ejemplo, `"a4"` y `"iso-c2"`), formatos habituales de EE. UU. como `"us-legal"` o `"us-letter"` y más. Consultá la referencia del argumento paper de la página[3.10.17/4.1] para conocer todas las opciones disponibles.

[A] https://practicaltypography.com/page-margins.html
[B] https://designregression.com/article/line-length-revisited-following-the-research

```
#set page("us-letter")

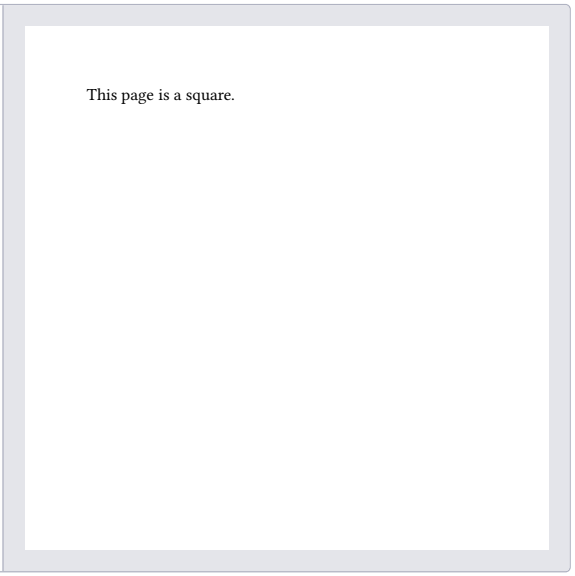
This page likes freedom.
```




Si necesitás personalizar el tamaño de tu página con ciertas dimensiones, podés especificar en cambio los argumentos con nombre width[3.10.17/4.2] y height[3.10.17/4.3].

```
#set page(width: 12cm, height: 12cm)

This page is a square.
```




## /1 Cambiar los márgenes de la página

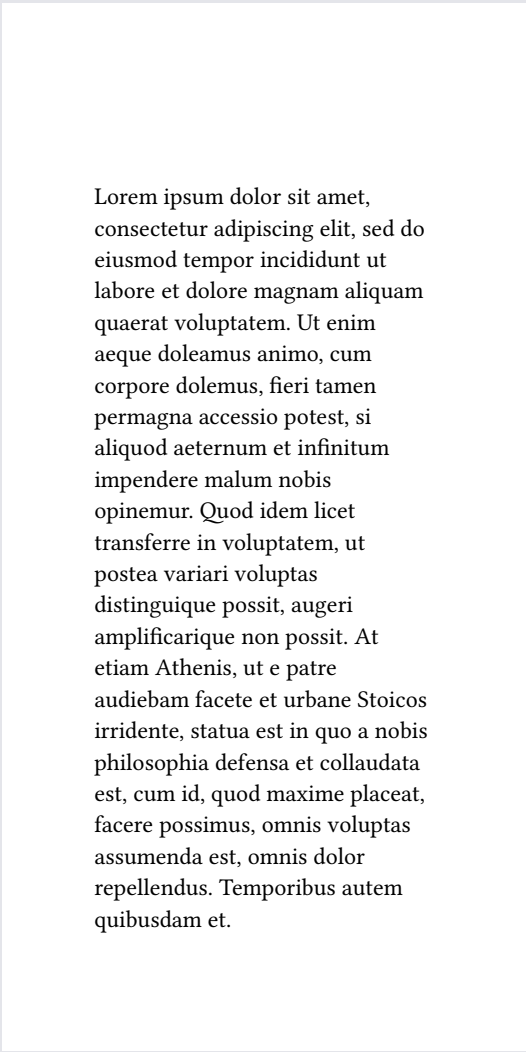
Los márgenes son un ingrediente vital de la buena tipografía: los tipógrafos consideran que las líneas de entre 45 y 75 caracteres son la mejor longitud para la legibilidad[A] y tus márgenes y tus columnas[4.2.4] ayudan a definir el ancho de las líneas. Por defecto, Typst creará márgenes proporcionales al tamaño de página de tu documento. Para establecer márgenes personalizados, vas a usar el argumento margin[3.10.17/4.5] en la regla set de page[3.10.17].

El argumento `margin` acepta una longitud si querés establecer todos los márgenes con el mismo ancho. Sin embargo, a menudo querés establecer márgenes distintos en cada lado. Para eso, podés pasar un diccionario:

[A] https://webtypography.net/2.1.2

```
#set page(margin: (
  top: 3cm,
  bottom: 2cm,
  x: 1.5cm,
))

#lorem(100)
```




El diccionario de márgenes de página puede tener claves para cada lado (`top`, `bottom`, `left`, `right`), pero también podés controlar juntos el izquierdo y el derecho estableciendo la clave `x` del diccionario de márgenes, como en el ejemplo. De la misma manera, los márgenes superior e inferior se pueden ajustar juntos estableciendo la clave `y`.

Si no especificás márgenes para todos los lados en el diccionario de márgenes, los márgenes anteriores seguirán vigentes en los lados sin establecer. Para evitar esto y establecer todos los márgenes restantes con un tamaño común, podés usar la clave `rest`. Por ejemplo, `#set page(margin: (left: 1.5in, rest: 1in))` establecerá el margen izquierdo en 1,5 pulgadas y los demás márgenes en una pulgada.

## /2 Márgenes distintos en páginas alternadas

A veces vas a necesitar alternar los márgenes horizontales entre las páginas pares e impares, por ejemplo, para tener más espacio hacia el lomo de un libro que en los bordes exteriores de sus páginas. Typst lleva registro de si una página está a la izquierda o a la derecha de la encuadernación. Podés usar esta información y establecer las claves `inside` u `outside` del diccionario de márgenes. El margen `inside` apunta hacia el lomo y el margen `outside` apunta hacia el borde del libro encuadernado.

```
#set page(margin: (inside: 2.5cm, outside: 2cm, y: 1.75cm))
```

Typst supondrá que los documentos escritos en escrituras de izquierda a derecha están encuadernados a la izquierda, mientras que los libros escritos en escrituras de derecha a izquierda están encuadernados a la derecha. Sin embargo, en algunos casos vas a tener que cambiar esto: si tu primera página la produce otra aplicación, la encuadernación queda invertida desde la perspectiva de Typst. Además, algunos libros, como

los mangas en inglés, suelen encuadernarse a la derecha, aunque el inglés use una escritura de izquierda a derecha. Para cambiar el lado de la encuadernación y establecer explícitamente dónde están `inside` y `outside`, establecé el argumento `binding`[3.10.17/4.7] en la regla set de `page`[3.10.17].

```
// Produce a book bound on the right,
// even though it is set in Spanish.
#set text(lang: "es")
#set page(binding: right)
```

Si `binding` es `left`, los márgenes `inside` estarán a la izquierda en las páginas impares, y viceversa.

### 4.2.2 Agregar encabezados y pies de página

Los encabezados y los pies de página se insertan en los márgenes superior e inferior de cada página. Podés agregar encabezados y pies de página personalizados o simplemente insertar un número de página.

Si necesitás algo más que un número de página, la mejor forma de insertar un encabezado y un pie de página son los argumentos `header`[3.10.17/4.13] y `footer`[3.10.17/4.15] de la regla set de `page`[3.10.17]. Podés pasar cualquier contenido como valor:

```
#set page(header: [
  _Lisa Strassner's Thesis_
  #h(1fr)
  National Academy of Sciences
])

#lorem(150)
```




Los encabezados están alineados abajo por defecto, para que no choquen con el borde superior de la página. Podés cambiarlo envolviendo tu encabezado en la función `align`[3.10.1].

#### /1 Encabezado y pie distintos en páginas específicas

Vas a necesitar encabezados y pies de página distintos en algunas páginas. Por ejemplo, quizás no quieras encabezado ni pie de página en la portada. El siguiente ejemplo muestra cómo quitar condicionalmente el encabezado de la primera página:

```
#set page(header: context {
  if counter(page).get().first() > 1 [
    _Lisa Strassner's Thesis_
    #h(1fr)
    National Academy of Sciences
```

```
  ]
})

#lorem(150)
```

Este ejemplo puede parecer intimidante, pero vamos a desglosarlo: al usar la palabra clave `context`, le decimos a Typst que el encabezado depende de dónde estamos en el documento. Luego le preguntamos a Typst si el contador[3.12.1] de páginas es mayor que uno en nuestra posición actual (dependiente del contexto). El contador de páginas empieza en uno, así que estamos salteando el encabezado en una sola página. Los contadores pueden tener varios niveles. Esta característica se usa para elementos como los títulos, pero el contador de páginas siempre tendrá un solo nivel, así que podemos mirar simplemente el primero.

Por supuesto, podés agregar un `else` a este ejemplo para agregar en cambio un encabezado distinto en la primera página.

### /2 Adaptar encabezados y pies en páginas con elementos específicos

La técnica descrita en la sección anterior se puede adaptar para realizar tareas más avanzadas usando las etiquetas de Typst. Por ejemplo, las páginas con tablas grandes podrían omitir sus encabezados para ayudar a reducir el desorden. Vamos a marcar nuestras tablas con una etiqueta[3.5.18] `<big-table>` y a usar el sistema de consultas[3.12.6] para averiguar si esa etiqueta existe en la página actual:

```
#set page(header: context {
  let matches = query(<big-table>)
  let current = counter(page).get()
  let has-table = matches.any(m =>
    counter(page).at(m.location()) == current
  )

  if not has-table [
    _Lisa Strassner's Thesis_
    #h(1fr)
    National Academy of Sciences
  ]
})

#lorem(100)
#pagebreak()

#table(
  columns: 2 * (1fr,),
  [A], [B],
  [C], [D],
) <big-table>
```

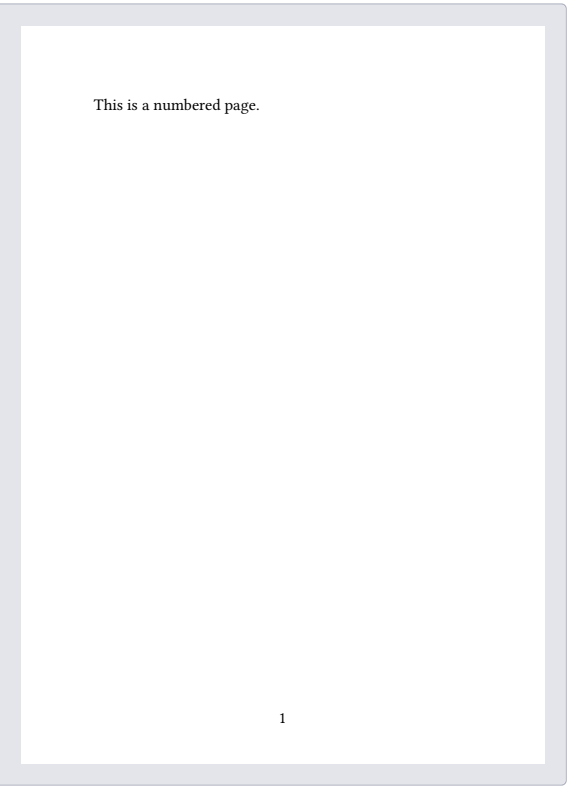
Acá consultamos todas las instancias de la etiqueta `<big-table>`. Luego comprobamos que ninguna de las tablas esté en la página de nuestra posición actual. Si es así, imprimimos el encabezado. Este ejemplo también usa variables para ser más conciso. Igual que arriba, podrías agregar un `else` para agregar otro encabezado en lugar de eliminarlo.

## 4.2.3 Agregar y personalizar números de página

Los números de página ayudan a los lectores a orientarse en tu documento y a hacer referencia a él más fácilmente. La forma más simple de insertar números de página es el argumento `numbering`[3.10.17/4.10] de la regla set de page[3.10.17]. Podés pasar una cadena con un *patrón de numeración*[3.6.12/3.1] que muestre cómo querés que se numeren tus páginas.

```
#set page(numbering: "1")

This is a numbered page.
```




Arriba podés ver el ejemplo más simple imaginable. Agrega un único número de página arábigo en el centro del pie de página. Podés especificar otros caracteres distintos de `"1"` para obtener otros numerales. Por ejemplo, `"i"` dará números romanos en minúscula. Cualquier carácter que no se interprete como número se mostrará tal cual. Por ejemplo, poné guiones alrededor de tu número de página escribiendo esto:

```
#set page(numbering: "- 1 -")

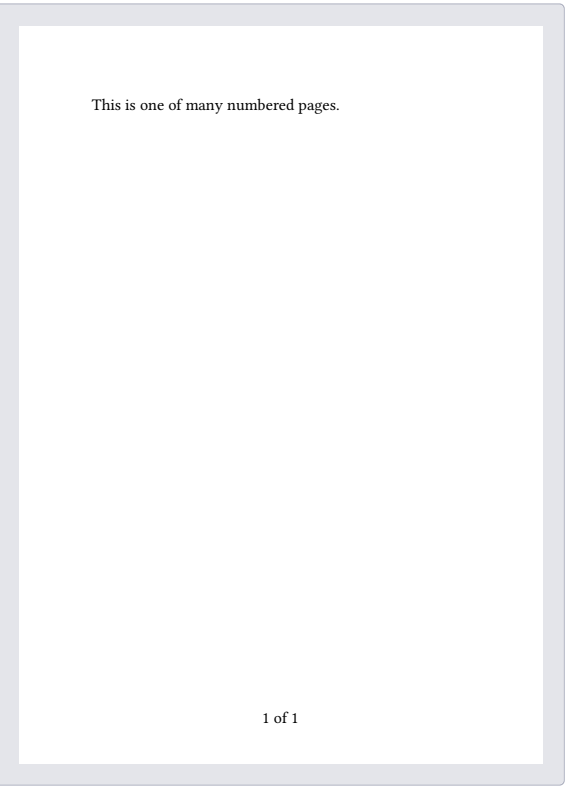
This is a - numbered - page.
```




Podés agregar la cantidad total de páginas ingresando un segundo carácter numérico en la cadena.

```
#set page(numbering: "1 of 1")

This is one of many numbered pages.
```




Andá a la referencia de la función `numbering`[3.6.12/3.1] para aprender más sobre los argumentos que podés pasar acá.

Si necesitás alinear el número de página a la derecha o a la izquierda, usá el argumento `number-align`[3.10.17/4.12] de la regla set de `page`[3.10.17]. Por ahora no se admite alternar la alineación entre páginas pares e impares con esta propiedad. Para hacerlo, vas a tener que especificar un pie de página personalizado y consultar el contador de páginas, como se describe en la sección sobre cómo omitir condicionalmente encabezados y pies de página.

## /1 Pie de página personalizado con números de página

A veces necesitás agregar a tu pie de página otro contenido además de un número de página. Sin embargo, una vez que se especifica un pie de página, el argumento `numbering`[3.10.17/4.10] de la regla set de `page`[3.10.17] se ignora. Esta sección te muestra cómo agregar un pie de página personalizado con números de página y más.

```
#set page(footer: context [
  *American Society of Proceedings*
  #h(1fr)
  #counter(page).display(
    "1/1",
    both: true,
  )
])

This page has a custom footer.
```




Primero agregamos algo de texto con énfasis fuerte a la izquierda y agregamos espacio libre para llenar la línea. Luego llamamos a `counter(page)` para obtener el contador de páginas y usamos su función `display` para mostrar su valor actual. También establecemos `both` en `true` para que nuestro patrón de numeración se aplique al número de página actual y al final.

También podemos ser más creativos con el número de página. Por ejemplo, insertemos un círculo por cada página.

```
#set page(footer: context [
  *Fun Typography Club*
  #h(1fr)
  #let (num,) = counter(page).get()
  #let circles = num * (
    box(circle(
      radius: 2pt,
      fill: navy,
    )),
  )
  #box(
    inset: (bottom: 1pt),
    circles.join(h(1pt))
  )
])

This page has a custom footer.
```



En este ejemplo, usamos el número de páginas para crear un array de círculos[3.11.2]. Los círculos están envueltos en un box[3.10.5] para que todos puedan aparecer en la misma línea, ya que son bloques y, de lo contrario, crearían saltos de párrafo. La longitud de este array[3.5.2] depende del número de página actual.

Luego insertamos los círculos en el lado derecho del pie de página, con 1pt de espacio entre ellos. El método join de un array intentará *unir*[3.3.2] los distintos valores de un array en un único valor, intercalado con su argumento. En nuestro caso, obtenemos un único valor de contenido con círculos y espacios entre ellos que podemos usar con la función align. Por último, usamos otro box para asegurarnos de que el texto y los círculos puedan compartir una línea y usamos el argumento `inset`[3.10.5/2.7] para subir un poco los círculos y que queden bien alineados con el texto.

### /2 Reiniciar el número de página y saltear páginas

¿Necesitás, en algún punto de tu documento, reiniciar el número de página? Quizás querés empezar con la primera página recién después de la portada. O quizás necesitás saltear algunos números de página porque vas a insertar páginas en el producto impreso final.

La forma correcta de modificar el número de página es manipular el contador[3.12.1] de páginas. La manipulación más simple es volver a poner el contador en 1.

```
#counter(page).update(1)
```

Esta línea reiniciará el contador de páginas en uno. Debería ubicarse al comienzo de una página, porque de lo contrario creará un salto de página. También podés actualizar el contador a partir de su valor anterior pasando una función:

```
#counter(page).update(n => n + 5)
```

En este ejemplo, salteamos cinco páginas. `n` es el valor actual del contador de páginas y `n + 5` es el valor de retorno de nuestra función.

Si necesitás obtener el número de página real en lugar del valor del contador de páginas, podés usar el método `page`[3.12.4/2.1] sobre el valor de retorno de la función `here`[3.12.2]:

```
#counter(page).update(n => n + 5)

// This returns one even though the
// page counter was incremented by 5.
#context here().page()
```

```
1
```

También podés obtener el patrón de numeración de páginas a partir de la ubicación devuelta por `here` con el método `page-numbering`[3.12.4/2.2].

## 4.2.4 Agregar columnas

Agregá columnas a tu documento para que entre más en una página manteniendo longitudes de línea legibles. Las columnas son bloques verticales de texto separados por algo de espacio en blanco. Este espacio se llama gutter.

Para disponer tu contenido en columnas, simplemente especificá la cantidad de columnas deseada en una regla set de `page`[3.10.17/4.8]. Para ajustar la cantidad de espacio entre las columnas, agregá una regla set sobre la función `columns`[3.10.7], especificando el parámetro `gutter`.

```
#set page(columns: 2)
#set columns(gutter: 12pt)

#lorem(30)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

Muy comúnmente, los trabajos científicos tienen un título y un resumen a una columna, mientras que el cuerpo principal está compuesto a dos columnas. Para lograr este efecto, la función `place`[3.10.19] de Typst puede salir temporalmente del diseño de dos columnas especificando `float: true` y `scope: "parent"`:

```
#set document(
  title: [Impacts of Odobenidae],
)
#set page(columns: 2)
#set par(justify: true)

#place(
  top + center,
  float: true,
  scope: "parent",
  title(),
)

= About seals in the wild
#lorem(80)
```

# Impacts of Odobenidae

## About seals in the wild

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere ma-

La *ubicación flotante* se refiere a que los elementos se empujan hacia la parte superior o inferior de la columna o de la página, mientras el resto del contenido fluye entremedio. También se usa con frecuencia para las figuras[3.6.7/5.3].

## /1 Usar columnas en cualquier parte de tu documento

Para crear columnas dentro de un diseño anidado, por ejemplo dentro de un rectángulo, podés usar directamente la función `columns`[3.10.7]. Sin embargo, realmente solo debería usarse dentro de diseños anidados. A nivel de página, es preferible la regla set de page, porque interactúa mejor con cosas como los elementos flotantes de página, las notas al pie y los números de línea.

```
#rect(
  width: 6cm,
  height: 3.5cm,
  columns(2, gutter: 12pt)[
    In the dimly lit gas station,
    a solitary taxi stood silently,
    its yellow paint fading with
    time. Its windows were dark,
    its engine idle, and its tires
    rested on the cold concrete.
  ]
)
```

In the dimly lit gas station, a solitary taxi stood silently, its yellow paint fading with time. Its windows were dark, its engine idle, and its tires rested on the cold concrete.

## /2 Columnas equilibradas

Si las columnas de la última página de un documento difieren mucho en longitud, pueden crear un diseño desparejo y poco atractivo. Por eso los tipógrafos suelen igualar la longitud de las columnas de la última página. Este efecto se llama equilibrar las columnas. Typst todavía no puede equilibrar columnas automáti-

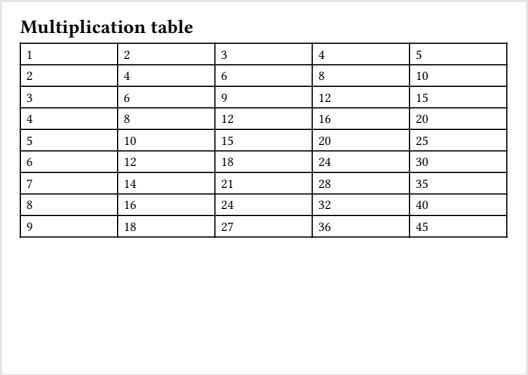
camente. Sin embargo, podés equilibrarlas manualmente colocando #colbreak()[3.10.6] en un lugar apropiado de tu marcado, creando a mano el salto de columna deseado.

### 4.2.5 Modificaciones puntuales

No necesitás reemplazar la configuración de tu página si necesitás insertar una sola página con una configuración distinta. Por ejemplo, quizás quieras insertar una página girada a horizontal para poner una tabla grande, o cambiar el margen y las columnas de tu portada. En ese caso, podés llamar a page[3.10.17] como una función con tu contenido como argumento y los reemplazos como los demás argumentos. Esto insertará suficientes páginas nuevas con tu configuración reemplazada para ubicar tu contenido en ellas. Typst volverá a la configuración de página de la regla set después de la llamada.

```
#page(flipped: true)[
  = Multiplication table

  #table(
    columns: 5 * (1fr,),
    ..for x in range(1, 10) {
      for y in range(1, 6) {
        (str(x*y),)
      }
    }
  )
]
```



## 4.3 Guía de tablas

Las tablas son una excelente manera de presentarles datos a tus lectores de forma legible, compacta y ordenada. No solo se usan para valores numéricos, sino también para respuestas de encuestas, planificación de tareas, horarios y más. Por esta amplia variedad de aplicaciones posibles, no existe una única mejor forma de diagramar una tabla. En cambio, pensá en los datos que querés destacar, en el diseño general de tu documento y, en última instancia, en cómo puede servirles mejor tu tabla a tus lectores.

¡Typst puede ayudarte con tus tablas automatizando el estilo, importando datos de otras aplicaciones y más! Esta guía repasa algunas de las preguntas más comunes que pueden surgirte al agregar una tabla a tu documento con Typst. Sentite libre de saltar a la sección que más te interese: diseñamos esta guía para que pueda leerse sin seguir el orden.

Si querés consultar algún detalle de cómo funcionan las tablas, también podés revisar su página de referencia[3.6.19]. Y si lo que buscás es un índice de contenidos y no una tabla común, la página de referencia de la función `outline`[3.6.13] es el lugar indicado para aprender más.

### 4.3.1 ¿Cómo creo una tabla básica?

Para crear una tabla en Typst, usá la función `table`[3.6.19]. Para una tabla básica, tenés que indicarle a la función table dos cosas:

- La cantidad de columnas
- El contenido de cada una de las celdas de la tabla

Entonces, supongamos que querés crear una tabla de dos columnas que describa los ingredientes de una receta de galletitas:

```
#table(
  columns: 2,
  [*Amount*], [*Ingredient*],
  [360g], [Baking flour],
  [250g], [Butter (room temp.)],
  [150g], [Brown sugar],
  [100g], [Cane sugar],
  [100g], [70% cocoa chocolate],
  [100g], [35-40% cocoa chocolate],
  [2], [Eggs],
  [Pinch], [Salt],
  [Drizzle], [Vanilla extract],
)
```

| Amount | Ingredient |
|---|---|
| 360g | Baking flour |
| 250g | Butter (room temp.) |
| 150g | Brown sugar |
| 100g | Cane sugar |
| 100g | 70% cocoa chocolate |
| 100g | 35-40% cocoa chocolate |
| 2 | Eggs |
| Pinch | Salt |
| Drizzle | Vanilla extract |

Este ejemplo muestra cómo llamar, configurar y completar una tabla. Tanto la cantidad de columnas como el contenido de las celdas se le pasan a la tabla como argumentos. La lista de argumentos[3.5.16] va entre paréntesis. Ahí pasamos primero la cantidad de columnas como argumento con nombre. Después pasamos varios bloques de contenido[3.5.8] como argumentos posicionales. Cada bloque de contenido contiene el contenido de una sola celda.

Para que el ejemplo sea más legible, pusimos dos argumentos de bloque de contenido por línea, imitando cómo aparecerían en la tabla. También podrías escribir cada celda en su propia línea. A Typst no le importa en qué línea pongas los argumentos. En cambio, Typst va a ubicar las celdas de contenido de izquierda a derecha (o de derecha a izquierda, si esa es la dirección de escritura de tu idioma) y luego de arriba hacia abajo. Agrega automáticamente las filas necesarias para que quepa todo tu contenido.

Lo mejor es envolver la fila de encabezado de tu tabla en la función `table.header`[3.6.19/4.4]. Esto aclara tu intención y además le permite a Typst hacer que el resultado sea más accesible[4.4] para usuarios de lectores de pantalla:

```
#table(
  columns: 2,
  table.header[*Amount*][*Ingredient*],
  [360g], [Baking flour],
 // ... the remaining cells
)
```

| Amount | Ingredient |
|---|---|
| 360g | Baking flour |
| 250g | Butter (room temp.) |
| 150g | Brown sugar |
| 100g | Cane sugar |
| 100g | 70% cocoa chocolate |
| 100g | 35-40% cocoa chocolate |
| 2 | Eggs |
| Pinch | Salt |
| Drizzle | Vanilla extract |

También podrías escribir una regla show que ponga automáticamente en énfasis fuerte[3.6.18] el contenido de las primeras celdas de todas las tablas. ¡Esto se vuelve útil enseguida si tu documento contiene varias tablas!

```
#show table.cell.where(y: 0): strong

#table(
  columns: 2,
  table.header[Amount][Ingredient],
  [360g], [Baking flour],
 // ... the remaining cells
)
```

| Amount | Ingredient |
|---|---|
| 360g | Baking flour |
| 250g | Butter (room temp.) |
| 150g | Brown sugar |
| 100g | Cane sugar |
| 100g | 70% cocoa chocolate |
| 100g | 35-40% cocoa chocolate |
| 2 | Eggs |
| Pinch | Salt |
| Drizzle | Vanilla extract |

Acá usamos una regla show con un selector por coordenadas de celda en lugar de aplicar nuestros estilos directamente a `table.header`. Esto se debe a una limitación actual de Typst que se va a corregir en una versión futura.

¡Felicitaciones, creaste tu primera tabla! Ahora podés pasar a cambiar el tamaño de las columnas[4.3.2], ajustar los trazos[4.3.5], agregar filas alternadas[4.3.4] y más.

### 4.3.2 ¿Cómo cambio el tamaño de las columnas?

Si creás una tabla y especificás la cantidad de columnas, Typst hace cada columna lo bastante ancha para que quepa su celda más grande. Muchas veces querés algo distinto, por ejemplo, que la tabla ocupe todo el ancho de la página. Podés pasar una lista con el ancho de cada columna mediante el argumento `columns`. Hay varias formas de especificar los anchos de columna:

- Primero, está `auto`. Es el comportamiento predeterminado e indica a Typst que haga crecer la columna para que entre su contenido. Si no hay espacio suficiente, Typst hace lo posible por repartir el espacio entre las columnas de tamaño `auto`.
- Longitudes[3.10.13] como `6cm`, `0.7in` o `120pt`. Como siempre, también podés usar la unidad `em`, que depende de la fuente y es un múltiplo del tamaño de fuente actual. Es útil si querés dimensionar tu tabla para que siempre entre aproximadamente la misma cantidad de texto, sin importar el tamaño de la fuente.
- Una proporción en porcentaje[3.10.20], como `40%`. Esto hace que la columna ocupe el 40 % del espacio horizontal total disponible para la tabla, es decir, el ancho interior de la página o del contenedor de la tabla. También podés combinar proporciones y longitudes en longitudes relativas[3.10.21]. Tené en cuenta que, aunque especifiques una lista de anchos de columna que sume 100 %, tu tabla igualmente podría ser más

grande que su contenedor. Esto se debe a que puede haber separación[3.6.19/3.3] entre columnas que no está incluida en los anchos de columna. Si querés que una tabla ocupe toda la página, la siguiente opción suele ser muy útil.

– Una fracción del espacio libre[3.10.9] con la unidad `fr`, como `1fr`. Esta unidad te permite repartir el espacio disponible entre las columnas. Funciona así: primero, Typst suma las longitudes de todas las columnas que no usan `fr`. Luego determina cuánto espacio horizontal queda. Ese espacio horizontal se reparte entre todas las columnas expresadas en `fr`. En este proceso, una columna de `2fr` va a ser el doble de ancha que una de `1fr`. De ahí viene el nombre: el ancho de la columna es su fracción del total de columnas de tamaño fraccionario.

Pongamos esto en práctica con una tabla que contiene las fechas, los números y las descripciones de algunos controles de rutina. Las dos primeras columnas son de tamaño `auto` y la última es de `1fr` para que ocupe toda la página.

```
#table(
  columns: (auto, auto, 1fr),
  table.header[Date][°No][Description],
  [24/01/03], [813], [Filtered participant pool],
  [24/01/03], [477], [Transitioned to sec. regimen],
  [24/01/11], [051], [Cycled treatment substrate],
)
```

| Date | °No | Description |
|---|---|---|
| 24/01/03 | 813 | Filtered participant pool |
| 24/01/03 | 477 | Transitioned to sec. regimen |
| 24/01/11 | 051 | Cycled treatment substrate |

Acá pasamos nuestra lista de longitudes de columna como un array[3.5.2], encerrado entre paréntesis y con sus elementos separados por comas. Las dos primeras columnas tienen tamaño automático, de modo que toman el tamaño de su contenido, y la tercera es `1fr` para que ocupe el resto del espacio de la página. Si en cambio quisieras que la segunda columna fuera un poco más espaciosa, podrías reemplazar su entrada en el array `columns` por un valor como `6em`.

### 4.3.3 ¿Cómo le pongo epígrafe a mi tabla y la referencio?

Una tabla vale tanto como la información que tus lectores extraen de ella. Podés mejorar la eficacia tanto de tu texto como de tu tabla estableciendo una conexión clara entre ambos con una referencia cruzada. Typst puede ayudarte con las referencias[3.6.17] automáticas y con la función `figure`[3.6.7].

Igual que con las imágenes, envolver una tabla en la función `figure` te permite agregarle un epígrafe y una etiqueta, para poder referenciar la figura en otro lugar. Envolver tu tabla en una figura también te permite usar el parámetro `placement` de la figura para hacerla flotar en la parte superior o inferior de una página.

Veamos una tabla con epígrafe y cómo referenciarla en el texto:

```
#show table.cell.where(y: 0): set text(weight: "bold")

#figure(
  table(
    columns: 4,
    stroke: none,

    table.header[Test Item][Specification][Test Result][Compliance],
```

```
    [Voltage], [220V ± 5%], [218V], [Pass],
    [Current], [5A ± 0.5A], [4.2A], [Fail],
  ),
  caption: [Probe results for design A],
) <probe-a>

The results from @probe-a show that the design is not yet optimal.
We will show how its performance can be improved in this section.
```

| Test Item | Specification | Test Result | Compliance |
|---|---|---|---|
| Voltage | 220V ± 5% | 218V | Pass |
| Current | 5A ± 0.5A | 4.2A | Fail |

Tabla 1: Probe results for design A

The results from Tabla 1 show that the design is not yet optimal. We will show how its performance can be improved in this section.

El ejemplo muestra cómo envolver una tabla en una figura, definir un epígrafe y una etiqueta, y cómo referenciar esa etiqueta. Empezamos usando la función `figure`, que espera el contenido de la figura como argumento posicional. Simplemente ponemos la llamada a la función table en su lista de argumentos, omitiendo el carácter # porque solo hace falta al llamar a una función en modo de marcado. También agregamos el epígrafe como argumento con nombre (arriba o abajo) de la tabla.

Después de la llamada a figure, ponemos una etiqueta entre paréntesis angulares (`<probe-a>`). Esto le indica a Typst que recuerde este elemento y lo haga referenciable con ese nombre en todo tu documento. Después podemos referenciarlo en el texto usando la arroba y el nombre de la etiqueta, `@probe-a`. Typst imprime una referencia bien formateada y la actualiza automáticamente si cambia el número de la tabla.

### 4.3.4 ¿Cómo hago una tabla con filas alternadas?

Muchas tablas usan filas o columnas alternadas en lugar de trazos para diferenciar unas de otras. Este efecto suele llamarse *rayas de cebra*. Las tablas con rayas de cebra son populares en aplicaciones empresariales, comerciales y de análisis de datos, mientras que las aplicaciones académicas tienden a usar trazos.

Para agregarle rayas de cebra a una tabla, usamos el argumento `fill` de la función `table`. Puede recibir tres tipos de valores:

- Un solo color (también puede ser un degradé o un mosaico) con el que se rellenan todas las celdas. Como queremos que algunas celdas tengan otro color, esto no sirve si queremos armar tablas cebra.
- Un array de colores por los que Typst cicla en cada columna. Podemos usar un array de dos elementos para obtener columnas alternadas.
- Una función que recibe la coordenada horizontal `x` y la coordenada vertical `y` de una celda y devuelve su relleno. Podemos usarla para crear rayas horizontales o patrones de tablero de ajedrez[3.10.10/6.1].

Empecemos con un ejemplo de una tabla con rayas horizontales:

```
#set text(font: "IBM Plex Sans")

// Medium bold table header.
#show table.cell.where(y: 0): set text(weight: "medium")

// Bold titles.
#show table.cell.where(x: 1): set text(weight: "bold")

// See the strokes section for details on this!
```

```
#let frame(stroke) = (x, y) => (
  left: if x > 0 { 0pt } else { stroke },
  right: stroke,
  top: if y < 2 { stroke } else { 0pt },
  bottom: stroke,
)

#set table(
  fill: (rgb("EAF2F5"), none),
  stroke: frame(1pt + rgb("21222C")),
)

#table(
  columns: (0.4fr, 1fr, 1fr, 1fr),

  table.header[Month][Title][Author][Genre],
  [January], [The Great Gatsby], [F. Scott Fitzgerald], [Classic],
  [February], [To Kill a Mockingbird], [Harper Lee], [Drama],
  [March], [1984], [George Orwell], [Dystopian],
  [April], [The Catcher in the Rye], [J.D. Salinger], [Coming-of-Age],
)
```

| Month | Title | Author | Genre |
|---|---|---|---|
| January | **The Great Gatsby** | F. Scott Fitzgerald | Classic |
| February | **To Kill a Mockingbird** | Harper Lee | Drama |
| March | **1984** | George Orwell | Dystopian |
| April | **The Catcher in the Rye** | J.D. Salinger | Coming-of-Age |

Este ejemplo muestra la lista de lecturas de un club de lectura. La línea `fill: (rgb("EAF2F5"), none)` en la regla set de `table` es todo lo que hace falta para agregar columnas alternadas. Le indica a Typst que alterne entre colorear las columnas con un celeste claro (en la llamada a la función rgb[3.11.3/4.5]) y nada (none). Fijate que sacamos todo el estilo de la llamada a la función `table` y lo pusimos en reglas set y show, para poder reutilizarlo automáticamente en varias tablas.

Como configurar las rayas en sí es fácil, agregamos también otros estilos para que se vea bien. El resto del código del ejemplo dibuja un trazo[4.3.5/3] azul oscuro alrededor de la tabla y debajo de la primera línea, y pone en negrita la primera fila y la columna con el título del libro. Mirá la sección de trazos[4.3.5] para ver los detalles de cómo logramos esta configuración.

A continuación veamos cómo cambiar solo la regla set para lograr en cambio rayas horizontales:

```
#set table(
  fill: (_, y) => if calc.odd(y) { rgb("EAF2F5") },
  stroke: frame(1pt + rgb("21222C")),
)
```

| Month | Title | Author | Genre |
|---|---|---|---|
| January | **The Great Gatsby** | F. Scott Fitzgerald | Classic |
| February | **To Kill a Mockingbird** | Harper Lee | Drama |
| March | **1984** | George Orwell | Dystopian |
| April | **The Catcher in the Rye** | J.D. Salinger | Coming-of-Age |

Solo tenemos que reemplazar la regla set del ejemplo anterior por esta para obtener rayas horizontales. Acá le pasamos una función a `fill`. Descarta la coordenada horizontal con un guion bajo y luego comprueba si la coordenada vertical y de la celda es impar. Si lo es, la celda recibe un relleno celeste claro; si no, no se devuelve ningún relleno.

Por supuesto, podés hacer esta función tan compleja como quieras. Por ejemplo, si querés alternar las filas con un tono de azul claro y otro más oscuro, podrías hacer algo así:

```
#set table(
  fill: (_, y) => (none, rgb("EAF2F5"), rgb("DDEAEF")).at(calc.rem(y, 3)),
  stroke: frame(1pt + rgb("21222C")),
)
```

| Month | Title | Author | Genre |
|---|---|---|---|
| January | **The Great Gatsby** | F. Scott Fitzgerald | Classic |
| February | **To Kill a Mockingbird** | Harper Lee | Drama |
| March | **1984** | George Orwell | Dystopian |
| April | **The Catcher in the Rye** | J.D. Salinger | Coming-of-Age |

Este ejemplo muestra una forma alternativa de escribir nuestra función de relleno. La función usa un array con tres colores y cicla entre sus valores en cada fila indexando el array con el resto de dividir y por 3.

Por último, un ejemplo extra que usa el *trazo* para lograr filas alternadas:

```
#set table(
  stroke: (x, y) => (
    y: 1pt,
    left: if x > 0 { 0pt } else if calc.even(y) { 1pt },
    right: if calc.even(y) { 1pt },
  ),
)
```

| Month | Title | Author | Genre |
|---|---|---|---|
| January | **The Great Gatsby** | F. Scott Fitzgerald | Classic |
| February | **To Kill a Mockingbird** | Harper Lee | Drama |
| March | **1984** | George Orwell | Dystopian |
| April | **The Catcher in the Rye** | J.D. Salinger | Coming-of-Age |

## /1 Cómo sobrescribir manualmente el color de relleno de una celda

A veces, el relleno de una celda no tiene que variar según su posición en la tabla, sino según su contenido. Podemos usar el elemento `table.cell`[3.6.19/4.1] en la lista de parámetros de `table` para envolver el contenido de una celda y sobrescribir su relleno.

Por ejemplo, esta es una lista de todos los presidentes de Alemania, con los bordes de las celdas coloreados con el color de su partido.

```
#set text(font: "Roboto")

#let cdu(name) = ([CDU], table.cell(fill: black, text(fill: white, name)))
#let spd(name) = ([SPD], table.cell(fill: red, text(fill: white, name)))
#let fdp(name) = ([FDP], table.cell(fill: yellow, name))

#table(
  columns: (auto, auto, 1fr),
  stroke: (x: none),

  table.header[Tenure][Party][President],
  [1949–1959], ..fdp[Theodor Heuss],
  [1959–1969], ..cdu[Heinrich Lübke],
  [1969–1974], ..spd[Gustav Heinemann],
  [1974–1979], ..fdp[Walter Scheel],
  [1979–1984], ..cdu[Karl Carstens],
  [1984–1994], ..cdu[Richard von Weizsäcker],
  [1994–1999], ..cdu[Roman Herzog],
  [1999–2004], ..spd[Johannes Rau],
  [2004–2010], ..cdu[Horst Köhler],
  [2010–2012], ..cdu[Christian Wulff],
  [2012–2017], [n/a], [Joachim Gauck],
  [2017–],     ..spd[Frank-Walter-Steinmeier],
)
```

| Tenure | Party | President |
|---|---|---|
| 1949–1959 | FDP | Theodor Heuss |
| 1959–1969 | CDU | Heinrich Lübke |
| 1969–1974 | SPD | Gustav Heinemann |
| 1974–1979 | FDP | Walter Scheel |
| 1979–1984 | CDU | Karl Carstens |
| 1984–1994 | CDU | Richard von Weizsäcker |
| 1994–1999 | CDU | Roman Herzog |
| 1999–2004 | SPD | Johannes Rau |
| 2004–2010 | CDU | Horst Köhler |
| 2010–2012 | CDU | Christian Wulff |
| 2012–2017 | n/a | Joachim Gauck |
| 2017– | SPD | Frank-Walter-Steinmeier |

En este ejemplo usamos variables porque solo hubo tres partidos cuyos miembros llegaron a ser presidentes (y un presidente independiente). Sus colores se repiten varias veces, así que guardamos una función que produce un array con el nombre del partido y una celda de tabla con el color de ese partido y el nombre del

presidente (`cdu`, `spd` y `fdp`). Después usamos estas funciones en la lista de argumentos de `table` en lugar de agregar directamente el nombre. Usamos el operador spread[3.5.1/2] `..` para convertir los elementos de los arrays en celdas individuales. También podríamos escribir algo como `[FDP], table.cell(fill: yellow) [Theodor Heuss]` para cada celda directamente en la lista de argumentos de `table`, pero se vuelve ilegible, sobre todo para los partidos con colores oscuros, que requieren texto blanco. También eliminamos los trazos verticales y configuramos la fuente en Roboto.

La columna del partido y el color de la celda comunican información redundante a propósito: comunicar datos importantes solo con color es una mala práctica de accesibilidad. Perjudica a los usuarios con discapacidad visual y viola los estándares de acceso universal, como el criterio de éxito 1.4.1 de WCAG 2.1[A]. Para mejorar esta tabla, agregamos una columna que imprime el nombre del partido. Como alternativa, podrías haber elegido una paleta amigable para el daltonismo y marcado tus celdas con una etiqueta adicional que los lectores de pantalla puedan leer en voz alta. Esta última función todavía no está soportada por Typst, pero se va a agregar en una versión futura. Podés comprobar cómo se ven los colores para lectores daltónicos con esta extensión de Chrome[B], Photoshop[C] o GIMP[D].

### 4.3.5 ¿Cómo ajusto las líneas de una tabla?

De forma predeterminada, Typst agrega trazos entre cada fila y columna de una tabla. Podés ajustarlos de varias maneras. La más práctica depende de la modificación que quieras hacer y de tu intención:

- ¿Querés darles estilo a todas las tablas de tu documento, sin importar su tamaño y contenido? Usá el argumento stroke[3.6.19/3.9] de la función `table` en una regla set.
- ¿Querés personalizar todas las líneas de una sola tabla? Usá el argumento stroke[3.6.19/3.9] de la función `table` al llamarla.
- ¿Querés cambiar, agregar o quitar el trazo alrededor de una sola celda? Usá el elemento `table.cell` en la lista de argumentos de tu llamada a table.
- ¿Querés cambiar, agregar o quitar una única línea horizontal o vertical en una sola tabla? Usá los elementos `table.hline`[3.6.19/4.2] y `table.vline`[3.6.19/4.3] en la lista de argumentos de tu llamada a table.

¡A continuación vamos a repasar todas estas opciones con ejemplos! Primero nos ocupamos del argumento stroke[3.6.19/3.9] de la función `table`. Ahí podés ajustar tanto cómo se dibujan las líneas de la tabla como cuáles se dibujan.

Empecemos modificando el color y el grosor del trazo:

```
#table(
  columns: 4,
  stroke: 0.5pt + rgb("666675"),
  [*Monday*], [11.5], [13.0], [4.0],
  [*Tuesday*], [8.0], [14.5], [5.0],
  [*Wednesday*], [9.0], [18.5], [13.0],
)
```

| | | | |
|---|---|---|---|
| **Monday** | 11.5 | 13.0 | 4.0 |
| **Tuesday** | 8.0 | 14.5 | 5.0 |
| **Wednesday** | 9.0 | 18.5 | 13.0 |

Esto hace que las líneas de la tabla sean un poco más finas y usa un gris azulado. Podés ver que le sumamos un ancho en puntos a un color para lograr nuestro trazo personalizado. Esta suma da como resultado un valor del tipo trazo[3.11.12]. Como alternativa, podés usar la representación como diccionario para los trazos, que te da acceso a funciones avanzadas como las líneas punteadas.

El ejemplo anterior mostró cómo usar el argumento stroke en la invocación de la función table. Como alternativa, podés especificar el argumento stroke en la regla set de `table`. Esto tiene exactamente el mismo efecto en todas las llamadas posteriores a `table` que si el argumento se hubiera especificado en la lista de argumentos. Es útil si estás escribiendo una plantilla o querés darle estilo a todo tu documento.

[A] https://www.w3.org/WAI/WCAG21/Understanding/use-of-color.html
[B] https://chromewebstore.google.com/detail/colorblindly/floniaahmccleoclneebhhmnjgdfijgg
[C] https://helpx.adobe.com/photoshop/using/proofing-colors.html
[D] https://docs.gimp.org/2.10/en/gimp-display-filter-dialog.html

```
// Renders the exact same as the last example
#set table(stroke: 0.5pt + rgb("666675"))

#table(
  columns: 4,
  [*Monday*], [11.5], [13.0], [4.0],
  [*Tuesday*], [8.0], [14.5], [5.0],
  [*Wednesday*], [9.0], [18.5], [13.0],
)
```

En las tablas chicas, a veces querés suprimir todos los trazos porque agregan demasiado ruido visual. Para eso, simplemente poné el argumento stroke en none:

```
#table(
  columns: 4,
  stroke: none,
  [*Monday*], [11.5], [13.0], [4.0],
  [*Tuesday*], [8.0], [14.5], [5.0],
  [*Wednesday*], [9.0], [18.5], [13.0],
)
```

| | | | |
|---|---|---|---|
| **Monday** | 11.5 | 13.0 | 4.0 |
| **Tuesday** | 8.0 | 14.5 | 5.0 |
| **Wednesday** | 9.0 | 18.5 | 13.0 |

Si querés un control más fino de dónde se colocan las líneas de tu tabla, también podés pasar un diccionario con las claves `top`, `left`, `right`, `bottom` (que controlan los lados respectivos de la celda), `x`, y (que controlan los trazos verticales y horizontales) y `rest` (que abarca todos los trazos a los que no les dio estilo otra entrada del diccionario). Todas las claves son opcionales; las claves omitidas usan el valor establecido anteriormente o el valor predeterminado si nunca se estableció. Por ejemplo, para obtener una tabla con solo líneas horizontales, podés hacer esto:

```
#table(
  columns: 2,
  stroke: (x: none),
  align: horizon,
  [☒], [Close cabin door],
  [☐], [Start engines],
  [☐], [Radio tower],
  [☐], [Push back],
)
```

| | |
|---|---|
| ☒ | Close cabin door |
| ☐ | Start engines |
| ☐ | Radio tower |
| ☐ | Push back |

Esto desactiva todos los trazos verticales y deja los horizontales. Para lograr el efecto inverso (solo trazos verticales), poné en cambio el argumento stroke en `(y: none)`.

Más adelante en la guía[4.3.5/3] vemos cómo usar una función en el argumento stroke para personalizar todos los trazos individualmente. Así se logran patrones de trazos más complejos.

## /1 Cómo agregar líneas individuales en la tabla

Si querés agregar una única línea horizontal o vertical a tu tabla, por ejemplo para separar un grupo de filas, podés usar los elementos `table.hline`[3.6.19/4.2] y `table.vline`[3.6.19/4.3] para líneas horizontales y verticales, respectivamente. Agregalos a la lista de argumentos de la función `table` igual que agregarías celdas individuales y un encabezado.

Veamos el siguiente ejemplo de la referencia:

```
#set table.hline(stroke: 0.6pt)

#table(
  stroke: none,
  columns: (auto, 1fr),
  // Morning schedule abridged.
  [14:00], [Talk: Tracked Layout],
  [15:00], [Talk: Automations],
  [16:00], [Workshop: Tables],
  table.hline(),
  [19:00], [Day 1 Attendee Mixer],
)
```

| | |
|---|---|
| 14:00 | Talk: Tracked Layout |
| 15:00 | Talk: Automations |
| 16:00 | Workshop: Tables |
| 19:00 | Day 1 Attendee Mixer |

En este ejemplo podés ver que pusimos una llamada a `table.hline` entre las celdas, lo que produce una línea horizontal en ese lugar. También usamos una regla set sobre el elemento para reducir el grosor de su trazo y que combine mejor con el peso de la fuente.

De forma predeterminada, Typst coloca las líneas horizontales y verticales después de la fila o columna actual, según su posición en la lista de argumentos. También podés moverlas manualmente a otra posición agregando el argumento `y` (para `hline`) o `x` (para `vline`). Por ejemplo, el siguiente código produciría el mismo resultado:

```
#set table.hline(stroke: 0.6pt)

#table(
  stroke: none,
  columns: (auto, 1fr),
  // Morning schedule abridged.
  table.hline(y: 3),
  [14:00], [Talk: Tracked Layout],
  [15:00], [Talk: Automations],
  [16:00], [Workshop: Tables],
  [19:00], [Day 1 Attendee Mixer],
)
```

Imaginemos que trabajás con una plantilla que no muestra ningún trazo de tabla salvo uno entre la primera y la segunda fila. Ahora bien, como tenés una tabla que además tiene etiquetas en la primera columna, querés agregarle una línea vertical extra. Sin embargo, no querés que esa línea vertical cruce la fila superior. Podés lograrlo con el argumento `start`:

```
// Base template already configured tables, but we need some
// extra configuration for this table.
#{
  set table(align: (x, _) => if x == 0 { left } else { right })
  show table.cell.where(x: 0): smallcaps
  table(
    columns: (auto, 1fr, 1fr, 1fr),
    table.vline(x: 1, start: 1),
    table.header[Trainset][Top Speed][Length][Weight],
    [TGV Réseau], [320 km/h], [200m], [383t],
    [ICE 403], [330 km/h], [201m], [409t],
    [Shinkansen N700], [300 km/h], [405m], [700t],
  )
}
```

| Trainset | Top Speed | Length | Weight |
|---|---|---|---|
| TGV Réseau | 320 km/h | 200m | 383t |
| ICE 403 | 330 km/h | 201m | 409t |
| Shinkansen N700 | 300 km/h | 405m | 700t |

En este ejemplo agregamos `table.vline` al comienzo de nuestra lista de argumentos posicionales. Pero como la línea no tiene que ir a la izquierda de la primera columna, especificamos el argumento `x` como `1`. También configuramos el argumento `start` en `1` para que la línea empiece recién después de la primera fila.

El ejemplo también contiene dos cosas más: usamos el argumento align con una función para alinear a la derecha los datos de todas las columnas salvo la primera, y usamos una regla show para que la primera columna de celdas aparezca en versalitas. Como estos estilos son específicos de esta tabla, ponemos todo en un bloque de código[3.3.2], para que el estilo no afecte a otras tablas.

## /2 Cómo sobrescribir los trazos de una sola celda

Imaginá que querés cambiar el trazo alrededor de una sola celda. ¡Quizás tu celda es muy importante y necesita destacarse! Para este caso existe la función `table.cell`[3.6.19/4.1]. En lugar de agregar tu contenido directamente en la lista de argumentos de la tabla, lo envolvés en una llamada a `table.cell`. Ahora podés usar la lista de argumentos de `table.cell` para sobrescribir las propiedades de la tabla, como el trazo, solo para esa celda.

Este es un ejemplo con una matriz de dos de los cinco grandes factores de personalidad, con una intersección resaltada.

```
#table(
  columns: 3,
  stroke: (x: none),

  table.header[][*High Neuroticism*][*Low Neuroticism*],

  [*High Agreeableness*],
  table.cell(stroke: orange + 2pt)[
    _Sensitive_ \ Prone to emotional distress but very empathetic.
  ],
  [_Compassionate_ \ Caring and stable, often seen as a supportive figure.],

  [*Low Agreeableness*],
  [_Contentious_ \ Competitive and easily agitated.],
  [_Detached_ \ Independent and calm, may appear aloof.],
)
```

| | High Neuroticism | Low Neuroticism |
|---|---|---|
| **High Agreeableness** | *Sensitive*<br>Prone to emotional distress but very empathetic. | *Compassionate*<br>Caring and stable, often seen as a supportive figure. |
| **Low Agreeableness** | *Contentious*<br>Competitive and easily agitated. | *Detached*<br>Independent and calm, may appear aloof. |

Arriba podés ver que usamos el elemento `table.cell` en la lista de argumentos de la tabla y le pasamos el contenido de la celda. Usamos su argumento `stroke` para establecer un trazo naranja más ancho. A pesar de que desactivamos los trazos verticales de la tabla, el trazo naranja apareció en todos los lados de la celda modificada, lo que muestra que la configuración de trazos de la tabla fue sobrescrita.

/3 ## Personalización compleja de trazos para todo el documento

Esta sección explica cómo personalizar todas las líneas a la vez en una o varias tablas. Esto te permite dibujar solo la primera línea horizontal u omitir las líneas exteriores, sin saber cuántas celdas tiene la tabla. Se logra pasando una función al parámetro `stroke` de la tabla. La función debería devolver un trazo dadas las posiciones x e y (desde cero) de la celda actual. Solo deberías necesitar estas funciones si sos autor de plantillas, no usás una plantilla o necesitás personalizar mucho tus tablas. En los demás casos, tu plantilla debería establecer trazos de tabla predeterminados adecuados.

Por ejemplo, esta es una regla set que dibuja todas las líneas horizontales salvo la primera y la última.

```
#show table.cell.where(x: 0): set text(style: "italic")
#show table.cell.where(y: 0): set text(style: "normal", weight: "bold")
#set table(stroke: (_, y) => if y > 0 { (top: 0.8pt) })

#table(
  columns: 3,
  align: center + horizon,
  table.header[Technique][Advantage][Drawback],
  [Diegetic], [Immersive], [May be contrived],
  [Extradiegetic], [Breaks immersion], [Obtrusive],
  [Omitted], [Fosters engagement], [May fracture audience],
)
```

| Technique | Advantage | Drawback |
|---|---|---|
| *Diegetic* | Immersive | May be contrived |
| *Extradiegetic* | Breaks immersion | Obtrusive |
| *Omitted* | Fosters engagement | May fracture audience |

En la regla set pasamos una función que recibe dos argumentos, asigna la coordenada vertical a `y` y descarta la coordenada horizontal. Luego devuelve un diccionario de trazo con un trazo superior de `0.8pt` para todas las líneas menos la primera. Las celdas de la primera línea reciben en cambio `none` implícitamente como valor de retorno. Podés modificar fácilmente esta función para dibujar solo las líneas verticales interiores, así: `(x, _) => if x > 0 { (left: 0.8pt) }`.

Probemos algunas funciones de trazo más. La siguiente solo dibuja una línea debajo de la primera fila:

```
#set table(stroke: (_, y) => if y == 0 { (bottom: 1pt) })

// Table as seen above
```

| Technique | Advantage | Drawback |
|---|---|---|
| *Diegetic* | Immersive | May be contrived |
| *Extradiegetic* | Breaks immersion | Obtrusive |
| *Omitted* | Fosters engagement | May fracture audience |

Si entendiste el primer ejemplo, lo que pasa acá resulta obvio. Comprobamos si estamos en la primera fila. Si es así, devolvemos un trazo inferior. De lo contrario, devolvemos none implícitamente.

El siguiente ejemplo muestra cómo dibujar todas las líneas salvo las exteriores:

```
#set table(stroke: (x, y) => (
  left: if x > 0 { 0.8pt },
  top: if y > 0 { 0.8pt },
))

// Table as seen above
```

| Technique | Advantage | Drawback |
|---|---|---|
| *Diegetic* | Immersive | May be contrived |
| *Extradiegetic* | Breaks immersion | Obtrusive |
| *Omitted* | Fosters engagement | May fracture audience |

Este ejemplo usa las coordenadas x e y. Omite el trazo izquierdo en la primera columna y el trazo superior en la primera fila. Las líneas derecha e inferior no se dibujan.

Por último, esta es una tabla que dibuja todas las líneas salvo las verticales de la primera fila y las horizontales del cuerpo de la tabla. Se parece un poco a un calendario.

```
#set table(stroke: (x, y) => (
  left: if x == 0 or y > 0 { 1pt } else { 0pt },
  right: 1pt,
  top: if y <= 1 { 1pt } else { 0pt },
  bottom: 1pt,
))

// Table as seen above
```

| Technique | Advantage | Drawback |
|---|---|---|
| *Diegetic* | Immersive | May be contrived |
| *Extradiegetic* | Breaks immersion | Obtrusive |
| *Omitted* | Fosters engagement | May fracture audience |

Este ejemplo es un poco más complejo. Empezamos dibujando todos los trazos a la derecha de las celdas. Pero eso significa que también dibujamos trazos en la fila superior, ¡y no los necesitamos! Aprovechamos que `left` sobrescribe a `right` y solo dibujamos la línea izquierda si no estamos en la primera fila o si estamos en la primera columna. En todos los demás casos, quitamos explícitamente la línea izquierda. Por último, dibujamos las líneas horizontales estableciendo primero la línea inferior y después, con la clave `top`, las primeras dos filas, suprimiendo todas las demás líneas superiores. La última línea aparece porque no hay ninguna línea `top` que pueda suprimirla.

### /4 ¿Cómo logro una línea doble?

Typst todavía no tiene una forma nativa de dibujar trazos dobles, pero hay varias maneras de emularlos, por ejemplo con mosaicos[3.11.13]. En esta sección mostramos otra solución alternativa: la separación entre celdas de la tabla.

Las tablas pueden separar sus celdas con el argumento `gutter`. Cuando se aplica una separación, se dibuja un trazo en cada una de las celdas ahora separadas. Podemos agregar separación de forma selectiva entre las filas o columnas en las que queremos dibujar una línea doble. Los argumentos `row-gutter` y `column-gutter` nos permiten hacerlo. Aceptan arrays de valores de separación. Veamos un ejemplo:

```
#table(
  columns: 3,
  stroke: (x: none),
  row-gutter: (2.2pt, auto),
  table.header[Date][Exercise Type][Calories Burned],
  [2023-03-15], [Swimming], [400],
  [2023-03-17], [Weightlifting], [250],
  [2023-03-18], [Yoga], [200],
)
```

| Date | Exercise Type | Calories Burned |
|---|---|---|
| 2023-03-15 | Swimming | 400 |
| 2023-03-17 | Weightlifting | 250 |
| 2023-03-18 | Yoga | 200 |

Vemos que usamos un array para `row-gutter` que especifica un espacio de `2.2pt` entre la primera y la segunda fila. Luego continúa con `auto` (que es el valor predeterminado, en este caso una separación de `0pt`), que será la separación entre todas las demás filas, ya que es la última entrada del array.

## 4.3.6 ¿Cómo alineo el contenido de las celdas de mi tabla?

Podés usar varios mecanismos para alinear el contenido de tu tabla. Podés usar el argumento `align` de la función `table` para establecer la alineación de toda la tabla (o usarlo en una regla set para establecerla en las tablas de todo tu documento), o bien la función `align`[3.10.1] (o el argumento `align` de `table.cell`) para sobrescribir la alineación de una sola celda.

Al usar el argumento align de la función `table`, podés elegir entre tres métodos para especificar una alineación[3.10.2]:

- Simplemente especificar una única alineación, como `right` (alinea en la esquina superior derecha) o `center + horizon` (centra todo el contenido de las celdas). Esto cambia la alineación de todas las celdas.
- Proporcionar un array. Typst cicla por este array en cada columna.
- Proporcionar una función a la que se le pasan las coordenadas horizontal `x` y vertical `y` de una celda y que devuelve una alineación.

Por ejemplo, este itinerario de viaje alinea a la derecha la columna del día y a la izquierda todo lo demás, proporcionando un array en el argumento `align`:

```
#set text(font: "IBM Plex Sans")
#show table.cell.where(y: 0): set text(weight: "bold")

#table(
  columns: 4,
  align: (right, left, left, left),
  fill: (_, y) => if calc.odd(y) { green.lighten(90%) },
  stroke: none,

  table.header[Day][Location][Hotel or Apartment][Activities],
  [1], [Paris, France], [Hôtel de l'Europe], [Arrival, Evening River Cruise],
  [2], [Paris, France], [Hôtel de l'Europe], [Louvre Museum, Eiffel Tower],
  [3], [Lyon, France], [Lyon City Hotel], [City Tour, Local Cuisine Tasting],
  [4], [Geneva, Switzerland], [Lakeview Inn], [Lake Geneva, Red Cross Museum],
  [5], [Zermatt, Switzerland], [Alpine Lodge], [Visit Matterhorn, Skiing],
)
```

| Day | Location | Hotel or Apartment | Activities |
|---|---|---|---|
| 1 | Paris, France | Hôtel de l'Europe | Arrival, Evening River Cruise |
| 2 | Paris, France | Hôtel de l'Europe | Louvre Museum, Eiffel Tower |
| 3 | Lyon, France | Lyon City Hotel | City Tour, Local Cuisine Tasting |
| 4 | Geneva, Switzerland | Lakeview Inn | Lake Geneva, Red Cross Museum |
| 5 | Zermatt, Switzerland | Alpine Lodge | Visit Matterhorn, Skiing |

Sin embargo, este ejemplo todavía no se ve perfecto: las celdas del encabezado deberían estar alineadas abajo. Usemos una función para lograrlo:

```
#set text(font: "IBM Plex Sans")
#show table.cell.where(y: 0): set text(weight: "bold")

#table(
  columns: 4,
  align: (x, y) =>
    if x == 0 { right } else { left } +
    if y == 0 { bottom } else { top },
  fill: (_, y) => if calc.odd(y) { green.lighten(90%) },
  stroke: none,

  table.header[Day][Location][Hotel or Apartment][Activities],
  [1], [Paris, France], [Hôtel de l'Europe], [Arrival, Evening River Cruise],
  [2], [Paris, France], [Hôtel de l'Europe], [Louvre Museum, Eiffel Tower],
```

```
  // ... remaining days omitted
)
```

| Day | Location | Hotel or Apartment | Activities |
|---|---|---|---|
| 1 | Paris, France | Hôtel de l'Europe | Arrival, Evening River Cruise |
| 2 | Paris, France | Hôtel de l'Europe | Louvre Museum, Eiffel Tower |
| 3 | Lyon, France | Lyon City Hotel | City Tour, Local Cuisine Tasting |
| 4 | Geneva, Switzerland | Lakeview Inn | Lake Geneva, Red Cross Museum |
| 5 | Zermatt, Switzerland | Alpine Lodge | Visit Matterhorn, Skiing |

En la función calculamos una alineación horizontal y otra vertical según si estamos en la primera columna (`x == 0`) o en la primera fila (`y == 0`). Luego aprovechamos que podemos sumar alineaciones horizontales y verticales con + para obtener una única alineación bidimensional.

Podés encontrar un ejemplo de uso de `table.cell` para cambiar la alineación de una sola celda en su página de referencia[3.6.19/4.1].

### 4.3.7 ¿Cómo combino celdas?

Cuando una tabla contiene agrupaciones lógicas o los mismos datos en varias celdas adyacentes, puede ser conveniente combinar varias celdas en una sola celda más grande. Otro caso de uso para los grupos de celdas son los encabezados de tabla con varias filas: así, por ejemplo, podés agrupar una tabla de datos de ventas por trimestre en la primera fila y por mes en la segunda.

Una celda combinada abarca varias filas y/o columnas. Lo lográs con los argumentos `rowspan` y `colspan` de la función `table.cell`[3.6.19/4.1]: simplemente especificá cuántas filas o columnas querés que abarque tu celda.

El siguiente ejemplo contiene un calendario de asistencia de una oficina, con los días presenciales y remotos de cada integrante del equipo. Para que la tabla se entienda de un vistazo, combinamos las celdas adyacentes con el mismo valor:

```
#let ofi = [Office]
#let rem = [_Remote_]
#let lea = [*On leave*]

#show table.cell.where(y: 0): set text(
  fill: white,
  weight: "bold",
)

#table(
  columns: 6 * (1fr,),
  align: (x, y) => if x == 0 or y == 0 { left } else { center },
  stroke: (x, y) => (
    // Separate black cells with white strokes.
```

```
    left: if y == 0 and x > 0 { white } else { black },
    rest: black,
  ),
  fill: (_, y) => if y == 0 { black },

  table.header(
    [Team member],
    [Monday],
    [Tuesday],
    [Wednesday],
    [Thursday],
    [Friday]
  ),
  [Evelyn Archer],
    table.cell(colspan: 2, ofi),
    table.cell(colspan: 2, rem),
    ofi,
  [Lila Montgomery],
    table.cell(colspan: 5, lea),
  [Nolan Pearce],
    rem,
    table.cell(colspan: 2, ofi),
    rem,
    ofi,
)
```

| Team member | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| Evelyn Archer | Office | | *Remote* | | Office |
| Lila Montgomery | **On leave** | | | | |
| Nolan Pearce | *Remote* | Office | | *Remote* | Office |

En el ejemplo, primero definimos variables con "Office", "Remote" y "On leave" para no tener que escribir estas etiquetas cada vez. Después podemos usar estas variables en el cuerpo de la tabla, ya sea directamente o en una llamada a `table.cell` si la persona pasa varios días seguidos en la oficina, en remoto o de licencia.

El ejemplo también contiene un encabezado negro (creado con el argumento `fill` de `table`) con trazos blancos (argumento `stroke` de `table`) y texto blanco (establecido por la regla set de `table.cell`). Por último, alineamos al centro todo el contenido de todas las celdas del cuerpo. Si querés saber más sobre las funciones que se le pasan a `align`, `stroke` y `fill`, podés consultar las secciones sobre alineación[3.10.2], trazos[4.3.5/3] y tablas con filas alternadas[4.3.4].

¡Esta tabla sería una excelente candidata para generarse de forma totalmente automática a partir de una fuente de datos externa! Mirá la sección sobre importación de datos[4.3.10] para aprender más al respecto.

### 4.3.8 ¿Cómo roto una tabla?

Cuando las tablas tienen muchas columnas, la orientación vertical del papel puede quedar chica enseguida. Por eso, a veces vas a querer pasar tus tablas a orientación horizontal. Hay dos maneras de lograrlo en Typst:

- Si querés rotar solo la tabla, pero no el resto del contenido de la página ni la página en sí, usá la función `rotate`[3.10.23] con el argumento `reflow` en `true`.
- Si querés rotar toda la página en la que está la tabla, podés usar la función `page`[3.10.17] con su argumento `flipped` en `true`. El encabezado, el pie de página y el número de página también aparecen ahora en el borde largo de la página. Esto tiene la ventaja de que la tabla se ve derecha cuando se lee en una computadora, pero también implica que una página de tu documento tiene dimensiones distintas de las demás, lo que puede resultarles chocante a tus lectores.

A continuación vamos a demostrar ambas técnicas con una tabla de calificaciones de estudiantes.

Primero vamos a rotar la tabla en la página. El ejemplo también coloca algo de texto a la derecha de la tabla.

```
#set page("a5", columns: 2, numbering: "- 1 -")
#show table.cell.where(y: 0): set text(weight: "bold")

#rotate(
  -90deg,
  reflow: true,

  table(
    columns: (1fr,) + 5 * (auto,),
    inset: (x: 0.6em,),
    stroke: (_, y) => (
      x: 1pt,
      top: if y <= 1 { 1pt } else { 0pt },
      bottom: 1pt,
    ),
    align: (left, right, right, right, right, left),

    table.header(
      [Student Name],
      [Assignment 1], [Assignment 2],
      [Mid-term], [Final Exam],
      [Total Grade],
    ),
    [Jane Smith], [78%], [82%], [75%], [80%], [B],
    [Alex Johnson], [90%], [95%], [94%], [96%], [A+],
    [John Doe], [85%], [90%], [88%], [92%], [A],
    [Maria Garcia], [88%], [84%], [89%], [85%], [B+],
    [Zhang Wei], [93%], [89%], [90%], [91%], [A-],
    [Marina Musterfrau], [96%], [91%], [74%], [69%], [B-],
  ),
)

#lorem(80)
```

| Student Name | Assignment 1 | Assignment 2 | Mid-term | Final Exam | Total Grade |
|---|---|---|---|---|---|
| Jane Smith | 78% | 82% | 75% | 80% | B |
| Alex Johnson | 90% | 95% | 94% | 96% | A+ |
| John Doe | 85% | 90% | 88% | 92% | A |
| Maria Garcia | 88% | 84% | 89% | 85% | B+ |
| Zhang Wei | 93% | 89% | 90% | 91% | A- |
| Marina Musterfrau | 96% | 91% | 74% | 69% | B- |

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et.



Lo que tenemos acá es un documento de dos columnas en papel ISO A5 con números de página al pie. La tabla tiene seis columnas y algunas personalizaciones de trazo[4-3-5], alineación y espaciado. Pero lo más importante es que la tabla está envuelta en una llamada a la función `rotate` con el argumento `reflow` en `true`. Esto hace que la tabla gire 90 grados en sentido antihorario. El argumento reflow es necesario para que la rotación de la tabla afecte al diseño. Si se omitiera, Typst diagramaría la página como si la tabla no estuviera rotada (`true` podría pasar a ser el valor predeterminado en el futuro).

El ejemplo también muestra cómo producir muchas columnas del mismo tamaño: a la columna inicial de `1fr` le sumamos un array con cinco elementos `auto`, que creamos multiplicando por cinco un array con un solo elemento `auto`. Tené en cuenta que los arrays con un solo elemento necesitan una coma final para distinguirlos de simples expresiones entre paréntesis.

El segundo ejemplo muestra cómo rotar toda la página para que la tabla quede derecha:

```
#set page("a5", numbering: "– 1 –")
#show table.cell.where(y: 0): set text(weight: "bold")

#page(flipped: true)[
  #table(
    columns: (1fr,) + 5 * (auto,),
    inset: (x: 0.6em,),
    stroke: (_, y) => (
      x: 1pt,
      top: if y <= 1 { 1pt } else { 0pt },
      bottom: 1pt,
    ),
    align: (left, right, right, right, right, left),

    table.header(
      [Student Name],
      [Assignment 1], [Assignment 2],
      [Mid-term], [Final Exam],
      [Total Grade],
    ),
    [Jane Smith], [78%], [82%], [75%], [80%], [B],
    [Alex Johnson], [90%], [95%], [94%], [96%], [A+],
    [John Doe], [85%], [90%], [88%], [92%], [A],
    [Maria Garcia], [88%], [84%], [89%], [85%], [B+],
    [Zhang Wei], [93%], [89%], [90%], [91%], [A-],
    [Marina Musterfrau], [96%], [91%], [74%], [69%], [B-],
  )

  #pad(x: 15%, top: 1.5em)[
    = Winter 2023/24 results
    #lorem(80)
  ]
]
```

| Student Name | Assignment 1 | Assignment 2 | Mid-term | Final Exam | Total Grade |
|---|---|---|---|---|---|
| Jane Smith | 78% | 82% | 75% | 80% | B |
| Alex Johnson | 90% | 95% | 94% | 96% | A+ |
| John Doe | 85% | 90% | 88% | 92% | A |
| Maria Garcia | 88% | 84% | 89% | 85% | B+ |
| Zhang Wei | 93% | 89% | 90% | 91% | A- |
| Marina Musterfrau | 96% | 91% | 74% | 69% | B- |

**Winter 2023/24 results**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et.

— 1 —

Acá tomamos la misma tabla y el otro contenido que queremos componer junto con ella y los ponemos en una llamada a la función page[3.10.17], pasándole `true` al argumento `flipped`. Esto le indica a Typst que cree páginas nuevas con el ancho y el alto intercambiados y que coloque el contenido de la llamada en una página nueva. Fijate que el número de página también está ahora en el borde largo del papel. Al pie de la página usamos la función pad[3.10.16] para acotar el ancho del párrafo y lograr una longitud de línea cómoda y legible.

### 4.3.9 ¿Cómo divido una tabla en varias páginas?

Lo mejor es que una tabla entre en una sola página. Sin embargo, algunas tablas tienen muchísimas filas y dividirlas entre varias páginas se vuelve inevitable. Por suerte, Typst soporta dividir tablas entre páginas de forma predeterminada. Si usás las funciones `table.header`[3.6.19/4.4] y `table.footer`[3.6.19/4.5], su contenido se repite en cada página como primera y última fila, respectivamente. Si querés desactivar este comportamiento, podés poner `repeat` en `false` en cualquiera de ellas.

Si pusiste tu tabla dentro de una figura[3.6.7], por defecto no puede dividirse entre páginas. Sin embargo, podés cambiar este comportamiento. Veamos cómo:

```
#set page(width: 9cm, height: 6cm)
#show table.cell.where(y: 0): set text(weight: "bold")
#show figure: set block(breakable: true)

#figure(
  caption: [Training regimen for Marathon],
  table(
    columns: 3,
    fill: (_, y) => if y == 0 { gray.lighten(75%) },

    table.header[Week][Distance (km)][Time (hh:mm:ss)],
    [1], [5],  [00:30:00],
    [2], [7],  [00:45:00],
    [3], [10], [01:00:00],
    [4], [12], [01:10:00],
    [5], [15], [01:25:00],
```

```
    [6], [18], [01:40:00],
    [7], [20], [01:50:00],
    [8], [22], [02:00:00],
    [...], [...], [...],
    table.footer[_Goal_][_42.195_][_02:45:00_],
  )
)
```

| Week | Distance (km) | Time (hh:mm:ss) |
|---|---|---|
| 1 | 5 | 00:30:00 |
| 2 | 7 | 00:45:00 |
| 3 | 10 | 01:00:00 |
| 4 | 12 | 01:10:00 |
| 5 | 15 | 01:25:00 |
| 6 | 18 | 01:40:00 |
| *Goal* | *42.195* | *02:45:00* |

| Week | Distance (km) | Time (hh:mm:ss) |
|---|---|---|
| 7 | 20 | 01:50:00 |
| 8 | 22 | 02:00:00 |
| … | … | … |
| *Goal* | *42.195* | *02:45:00* |

Tabla 1: Training regimen for Marathon

Una figura produce automáticamente un bloque[3.10.4] que, por defecto, no puede dividirse. Sin embargo, podemos reconfigurar el bloque de la figura con una regla show para hacerlo `breakable`. Ahora la figura abarca varias páginas y los encabezados y pies se repiten.

### 4.3.10 ¿Cómo importo datos a una tabla?

Muchas veces necesitás poner en una tabla datos que obtuviste de otro lado. A veces vienen de Microsoft Excel o Google Sheets, a veces de un conjunto de datos de la web o de tu experimento. Por suerte, Typst puede cargar muchos formatos de archivo comunes[3.13], así que podés usar scripting para incluir sus datos en una tabla.

El formato de archivo más común para datos tabulares es CSV. Podés obtener un archivo CSV desde Excel eligiendo "Guardar como" en el menú *Archivo* y seleccionando el formato "CSV UTF-8 (delimitado por comas) (.csv)". Guardá el archivo y, si usás la app web, subilo a tu proyecto.

En nuestro caso, vamos a construir una tabla sobre la Ley de Moore. Para eso usamos una estadística sobre cuántos transistores tiene en promedio un microprocesador por año, de Our World in Data[A]. Empecemos presionando el botón "Download" para obtener un archivo CSV con los datos sin procesar.

[A] https://ourworldindata.org/grapher/transistors-per-microprocessor

Asegurate de mover el archivo a tu proyecto o a algún lugar que Typst pueda ver, si usás la CLI. Una vez que lo hiciste, podemos abrir el archivo para ver cómo está estructurado:

```
Entity,Code,Year,Transistors per microprocessor
World,OWID_WRL,1971,2308.2417
World,OWID_WRL,1972,3554.5222
World,OWID_WRL,1974,6097.5625
```

El archivo empieza con un encabezado y contiene cuatro columnas: Entity (a quién se aplica la métrica), Code, el año y la cantidad de transistores por microprocesador. Solo las dos últimas columnas cambian entre una fila y otra, así que podemos ignorar "Entity" y "Code".

Primero, carguemos este archivo con la función `csv`[3.13.3]. Acepta como argumento de tipo cadena de texto el nombre del archivo que queremos cargar:

```
#let moore = csv("moore.csv")
```

Cargamos nuestro archivo (suponiendo que lo llamamos `moore.csv`) y lo asociamos[3.3.3] a la nueva variable `moore`. Esto no produce ninguna salida, así que todavía no hay nada para ver. Si queremos examinar lo que Typst cargó, podemos pasar el mouse sobre el nombre de la variable en la app web o imprimir algunos elementos del array:

```
#let moore = csv("moore.csv")

#moore.slice(0, 3)
```

```
(
  (
    "Entity",
    "Code",
    "Year",
    "Transistors per microprocessor",
  ),
  ("World", "OWID_WRL", "1995",
"9646616"),
  ("World", "OWID_WRL", "1996",
"9646616"),
)
```

Con los argumentos `(0, 3)`, el método `slice`[3.5.2/3.9] devuelve los tres primeros elementos del array (con los índices 0, 1 y 2). Vemos que cada fila es su propio array con un elemento por celda.

Ahora escribamos un bucle que transforme estos datos en un array de celdas que podamos usar con la función table.

```
#let moore = csv("moore.csv")

#table(
  columns: 2,
  ..for (.., year, count) in moore {
    (year, count)
  }
)
```

| Year | Transistors per microprocessor |
|---|---|
| 1995 | 9646616 |
| 1996 | 9646616 |
| 1997 | 9646616 |
| 1998 | 15261378 |
| 1999 | 21673922 |
| 2000 | 37180264 |
| 2001 | 42550656 |
| 2002 | 220673400 |
| 2003 | 220673400 |
| 2004 | 273842000 |
| 2005 | 305052770 |
| 2006 | 582941600 |
| 2007 | 805842200 |
| 2008 | 805842200 |
| 2009 | 2308241400 |
| 2010 | 2308241400 |
| 2011 | 2600000000 |
| 2012 | 2600000000 |
| 2013 | 5000000000 |
| 2014 | 5700000000 |
| 2016 | 8000000000 |
| 2017 | 19200000000 |
| 2018 | 21100000000 |
| 2019 | 39500000000 |
| 2020 | 39500000000 |
| 2021 | 58200000000 |

El ejemplo anterior usa un bucle for que itera sobre las filas de nuestro archivo CSV y devuelve un array en cada iteración. Usamos la capacidad de desestructuración[3.3.3] del bucle for para descartar todos los elementos de cada fila salvo los dos últimos. Luego creamos un array nuevo con solo esos dos. Como Typst concatena los arrays resultantes de todas las iteraciones del bucle, obtenemos un array unidimensional en el que se alternan la columna del año y la cantidad de transistores. Después podemos insertar el array como celdas. Para eso usamos el operador spread[3.5.1/2] (`..`). Al anteponer dos puntos a un array o, en nuestro caso, a una expresión que produce un array, le indicamos a Typst que los elementos del array deben usarse como argumentos posicionales.

Como alternativa, también podemos usar los métodos de array `map`[3.5.2/3.15], `slice`[3.5.2/3.9] y `flatten`[3.5.2/3.23] para escribir esto en un estilo más funcional:

```
#let moore = csv("moore.csv")

#table(
   columns: 2,
   ..moore.map(m => m.slice(2, 4)).flatten(),
)
```

Este ejemplo se renderiza igual que el anterior, pero primero cargamos el CSV y luego transformamos cada fila con `map`. La función que le pasamos a `map` se aplica a cada fila de los datos y devuelve un array nuevo que reemplaza a la fila original. Acá usamos `.slice(2, 4)` para extraer solo la tercera y la cuarta columna, ya que son las que queremos conservar. Como `moore` es un array bidimensional (cada fila es a su vez un array),

el resultado del mapeo sigue siendo un array anidado. La función `flatten` convierte esta estructura anidada en un array unidimensional, que es lo que se necesita al expandir los datos en la función `table`. Por último, especificamos explícitamente `columns: 2` porque conservamos exactamente dos columnas de cada fila.

Ahora que tenemos un buen código para nuestra tabla, ¡tratemos de que la tabla en sí también quede linda! La cantidad de transistores pasa de millones en 1995 a billones en 2021 y con tantos dígitos cuesta ver los cambios. Podríamos intentar presentar nuestros datos en escala logarítmica para hacerlos más digeribles:

```
#let moore = csv("moore.csv")
#let moore-log = moore.slice(1).map(m => {
  let (.., year, count) = m
  let log = calc.log(float(count))
  let rounded = str(calc.round(log, digits: 2))
  (year, rounded)
})

#show table.cell.where(x: 0): strong

#table(
   columns: moore-log.first().len(),
   align: right,
   fill: (_, y) => if calc.odd(y) { rgb("D7D9E0") },
   stroke: none,

   table.header[Year][Transistor count ($log_10$)],
   table.hline(stroke: rgb("4D4C5B")),
   ..moore-log.flatten(),
)
```

| Year | Transistor count ($\log_{10}$) |
|---|---|
| **1995** | 6.98 |
| **1996** | 6.98 |
| **1997** | 6.98 |
| **1998** | 7.18 |
| **1999** | 7.34 |
| **2000** | 7.57 |
| **2001** | 7.63 |
| **2002** | 8.34 |
| **2003** | 8.34 |
| **2004** | 8.44 |
| **2005** | 8.48 |
| **2006** | 8.77 |
| **2007** | 8.91 |
| **2008** | 8.91 |
| **2009** | 9.36 |
| **2010** | 9.36 |
| **2011** | 9.41 |
| **2012** | 9.41 |
| **2013** | 9.7 |
| **2014** | 9.76 |
| **2016** | 9.9 |
| **2017** | 10.28 |
| **2018** | 10.32 |
| **2019** | 10.6 |
| **2020** | 10.6 |
| **2021** | 10.76 |

En este ejemplo, primero descartamos la fila de encabezado de los datos, ya que agregamos la nuestra. Luego descartamos todas las columnas salvo las dos últimas, como antes. Lo hacemos desestructurando[3.3.3] el array `m` y descartando todo menos los dos últimos elementos. Después convertimos la cadena de `count` en un número de punto flotante, calculamos su logaritmo y lo guardamos en la variable `log`. Finalmente, lo redondeamos a dos dígitos, lo convertimos en cadena de texto y lo guardamos en la variable `rounded`. Luego devolvemos un array con `year` y `rounded` que reemplaza a la fila original. En nuestra tabla agregamos un encabezado propio que le indica al lector que aplicamos un logaritmo a los valores. Después expandimos los datos aplanados, como antes.

También le dimos estilo a la tabla con rayas[4.3.4], una línea horizontal[4.3.5/1] debajo de la primera fila, todo alineado[4.3.6] a la derecha y la primera columna en negrita. ¡Hacé clic en los enlaces para ir a las secciones de la guía correspondientes y ver cómo se hace!

### 4.3.11 ¿Qué hago si necesito la función table para algo que no es una tabla?

Las disposiciones tabulares del contenido pueden ser útiles no solo para matrices de datos estrechamente relacionados, como los que se muestran en los ejemplos de esta guía, sino también con fines de presentación. Typst distingue entre las grillas, que sirven solo para maquetar y presentar, y las tablas, en las que la disposición de las celdas transmite información en sí misma.

Para dejar clara esta diferencia a otros programas y permitir que las plantillas les den mucho estilo a las tablas, Typst tiene dos funciones para la disposición en grilla y en tabla:

- La función `table`[3.6.19], explicada a lo largo de esta guía, pensada para datos tabulares.
- La función `grid`[3.10.10], pensada para fines de presentación y de maquetación de páginas.

Ambos elementos funcionan de la misma manera y tienen los mismos argumentos. Todo lo que aprendiste sobre tablas en esta guía se aplica a las grillas. Solo hay tres diferencias:

- Vas a tener que usar los elementos `grid.cell`[3.10.10/6.1], `grid.vline`[3.10.10/6.3] y `grid.hline`[3.10.10/6.2] en lugar de `table.cell`[3.6.19/4.1], `table.vline`[3.6.19/4.3] y `table.hline`[3.6.19/4.2].
- La grilla tiene valores predeterminados distintos: no dibuja trazos por defecto y no tiene espaciado (`inset`) dentro de sus celdas.
- Elementos como `figure` no reaccionan a las grillas, ya que se supone que no tienen incidencia semántica en la estructura del documento.

## 4.4 Guía de accesibilidad

Hacer accesible un documento significa que todos puedan usarlo y entenderlo. Eso no incluye solo a las personas con discapacidades permanentes o temporales, sino también a quienes usan otros dispositivos o tienen otras preferencias. Para subrayar por qué es importante la accesibilidad, pensá que tu documento puede leerse en más contextos de los que esperabas:

- Alguien puede imprimir el documento en papel
- Alguien puede leer tu documento en un celular, con el reflujo activado en su lector de PDF
- Alguien puede hacer que su computadora le lea el documento en voz alta
- Alguien puede pedirle a una inteligencia artificial que le resuma tu documento
- Alguien puede convertir tu documento a otro formato, como HTML, que le resulte más accesible

Para atender a todas estas personas y escenarios, tendrías que diseñar tu documento para el **Acceso universal.** El Acceso universal es un principio simple pero poderoso: en lugar de adaptar un proyecto para que sea accesible una vez terminado, diseñalo desde el principio para que funcione con la mayor variedad posible de personas y situaciones. ¡Esto mejora la experiencia de todos los lectores!

Typst puede ayudarte a crear archivos accesibles que se lean bien con lectores de pantalla, que se vean bien incluso al reflujarse para otro tamaño de pantalla y que pasen los verificadores automáticos de accesibilidad. Sin embargo, para crear archivos accesibles vas a tener que tener presentes algunas reglas. Esta guía te ayuda a conocer qué problemas afectan la accesibilidad, cómo diseñar para el Acceso universal y qué herramientas te da Typst para lograrlo. Gran parte de las recomendaciones valen para todos los formatos de exportación, pero la guía se centra en la exportación a PDF. Las diferencias importantes con la exportación a HTML se señalan aparte.

### 4.4.1 Conceptos básicos de accesibilidad

Los archivos accesibles permiten que el software haga más que simplemente renderizarlos. Tu computadora puede entender qué representa cada parte del documento y usar esa información para presentárselo al usuario.

Distintos programas consumen esta información para dar acceso. Al exportar un PDF desde Typst, el *visor de PDF* (a veces llamado también lector) muestra las páginas del documento tal como las diseñaste en la vista previa de Typst. Algunas personas dependen de *tecnologías de asistencia* (TA), como lectores de pantalla, pantallas braille, magnificadores de pantalla y más, para consumir archivos PDF. En ese caso, la información semántica del archivo se usa para adaptar su contenido a texto hablado o escrito, o a otra representación visual. Otras personas hacen que el visor de PDF reflujee el archivo para crear un diseño similar al de una página web: el contenido se ajusta al ancho de la ventana y se desplaza de forma continua. Por último, algunos usuarios reutilizan el PDF en otro formato, por ejemplo texto plano para dárselo a un Modelo de Lenguaje Grande (LLM) o HTML. Una forma especial de reutilización es copiar y pegar, donde los usuarios usan el portapapeles para extraer contenido de un archivo y usarlo en otra aplicación.

El soporte de accesibilidad varía según el visor y la TA. Algunas combinaciones funcionan mejor que otras. En nuestras pruebas, Adobe Acrobat[A] junto con NVDA[B] en Windows y VoiceOver[C] en macOS ofrecieron el soporte de accesibilidad más completo. Con la exportación a HTML, los navegadores brindan una base de accesibilidad más consistente que los lectores de PDF.

Solo la exportación a PDF y a HTML produce archivos accesibles. Ni los PNG ni los SVG son accesibles por sí solos. Ambos formatos pueden usarse dentro de una obra mayor que sea accesible si les proporcionás una representación textual[4.4.7].

### 4.4.2 Mantener la semántica

Para agregar a un archivo la información semántica correcta para las TA y la reutilización, Typst necesita saber qué rol semántico cumple cada parte. Por ejemplo, esto significa que un título en un PDF compilado

[A] https://www.adobe.com/acrobat.html
[B] https://www.nvaccess.org/download/
[C] https://support.apple.com/guide/voiceover/welcome/mac

no debería ser simplemente texto grande y en negrita. En cambio, el archivo debería contener la información explícita (conocida como *etiqueta*, o *tag*) de que cierto texto constituye un título. Entonces un lector de pantalla lo anunciará como título y le permitirá al usuario navegar entre títulos.

Si usás Typst de forma idiomática, con el marcado y los elementos integrados, Typst agrega automáticamente etiquetas con información semántica rica a tus archivos. Veamos dos ejemplos de código:

```
// ❌ Don't do this
#text(
  size: 16pt,
  weight: "bold",
)[Heading]
```

**Heading**

```
// ✅ Do this
#show heading: set text(size: 16pt)
= Heading
```

**Heading**

Los dos ejemplos se ven igual. Ambos contienen el texto "Heading" en negrita, con tamaño de 16 puntos. Sin embargo, solo el segundo es accesible. Al usar el marcado de título, Typst sabe que el significado semántico de este texto es el de un título y puede propagar esa información al PDF final. En el primer ejemplo, solo sabe que tiene que usar negrita y un cuerpo más grande en un texto que, por lo demás, es normal, y no puede asumir que querías un título y no una decisión de estilo u otro elemento, como una cita.

Usar semántica no se limita a los títulos. Estos son algunos ejemplos más de elementos que tenés que usar:

- Usá guiones bajos / `emph`[3.6.6] en lugar de la función `text`[3.7.12] para poner texto en énfasis
- Usá asteriscos / `strong`[3.6.18] en lugar de la función text para dar énfasis fuerte al texto
- Usá listas (`list`[3.6.2], `enum`[3.6.11], `terms`[3.6.20]) en lugar de texto normal con saltos de línea cuando trabajes con contenido enumerado u ordenado
- Usá `quote`[3.6.16] para las citas en línea y en bloque
- Usá las funciones integradas `bibliography`[3.6.1] y `cite`[3.6.3] en lugar de imprimir la bibliografía a mano
- Usá etiquetas y `ref`[3.6.17] o `@references` para referenciar otras partes de tus documentos en lugar de escribir la referencia a mano
- Usá el argumento `caption` del elemento `figure`[3.6.7/6.1] para agregar epígrafes en lugar de ponerlos como texto debajo de la llamada a la función

Si querés cambiar la apariencia predeterminada de un elemento, no lo reemplaces por tu propia función. En cambio, usá reglas set[3.2.1], reglas show-set y reglas show[3.2.2] para personalizar su aspecto. Este es un ejemplo de cómo cambiar el aspecto del énfasis fuerte en tu documento:

```
// Change how text inside of strong emphasis looks
#show strong: set text(tracking: 0.2em, fill: blue, weight: "black")

When setting up your tents, *never forget* to secure the pegs.
```

When setting up your tents, **n e v e r f o r g e t** to secure the pegs.

La regla show-set cambia por completo la apariencia predeterminada del elemento `strong`[3.6.18], pero conserva su significado semántico. Si necesitás todavía más personalización, podés darles a las reglas show código de diseño totalmente propio y Typst seguirá reteniendo el propósito semántico del elemento.

### 4.4.3 Orden de lectura

Para que las TA lean el contenido de un documento en el orden correcto, y para las aplicaciones de reutilización, los archivos accesibles deben hacer explícito su orden de lectura. Esto se debe a que el orden lógico

de lectura puede diferir del orden del diseño. Las figuras flotantes son un ejemplo común de esa diferencia: una figura puede ser relevante para un párrafo en el centro de la página pero aparecer en el borde superior o inferior. En los archivos no accesibles, los lectores de PDF y las TA tienen que asumir que el orden del diseño es igual al orden lógico de lectura, lo que suele confundir a los usuarios de TA. Cuando el orden de lectura está bien definido, los lectores de pantalla leen una nota al pie o una figura flotante justo donde tiene sentido.

Por suerte, el marcado de Typst ya implica un único orden de lectura. Podés asumir que los documentos de Typst se leen en el orden en que el contenido fue colocado en el marcado. Para la mayoría de los documentos, eso alcanza. Sin embargo, cuando usás las funciones `place`[3.10.19] y `move`[3.10.15] o figuras flotantes[3.6.7/5.3], tenés que prestar especial atención a poner la llamada a la función en un lugar apropiado del orden lógico de lectura en el marcado, aunque eso no tenga consecuencias en el diseño. Simplemente preguntate dónde querrías que un lector de pantalla anuncie el contenido que estás colocando.

### 4.4.4 Contenedores de diseño

Typst ofrece algunos contenedores de diseño, como `grid`[3.10.10], `stack`[3.10.28], `box`[3.10.5], `columns`[3.10.7] y `block`[3.10.4], para organizar visualmente tu contenido. Ninguno de estos contenedores tiene significado semántico asociado. Typst conserva algunos de ellos durante el reflujo del PDF y descarta otros.

Al diseñar para el Acceso universal, tenés que tener presente que los usuarios de TA a menudo no pueden ver el diseño visual que crea el contenedor. En cambio, la TA solo lee su contenido, así que lo mejor es pensar en estos contenedores como transparentes en términos de accesibilidad. Por ejemplo, el contenido de una grilla se leerá de forma plana, en el orden en que agregaste las celdas en el código fuente. Si el diseño que creaste es meramente visual y decorativo, no hay problema. Pero si el diseño tiene un significado semántico evidente para quien ve el archivo en un lector de PDF común, entonces no es accesible. En ese caso, creá una representación alternativa de tu contenido que aproveche el texto, o envolvé tu contenedor en el elemento `figure`[3.6.7] para dar una descripción textual alternativa.

No uses el contenedor grid para representar datos tabulares. En su lugar, usá `table`[3.6.19]. Las tablas son accesibles para los usuarios de TA: su TA les permite navegar la tabla en dos dimensiones. Las tablas se conservan durante el reflujo y la reutilización. Al crear tablas, usá los elementos `table.header`[3.6.19/4.4] y `table.footer`[3.6.19/4.5] para marcar los roles semánticos de cada fila. La documentación de table contiene una sección de accesibilidad[3.6.19/2] con más información sobre cómo hacer accesibles tus tablas. Tené en cuenta que, aunque los usuarios de TA pueden acceder a las tablas, muchas veces les resulta engorroso: las tablas están optimizadas para el consumo visual. Que te lean el contenido de un conjunto de celdas y tener que recordar su fila y su columna genera una carga mental adicional. Considerá hacer accesible la conclusión principal de la tabla como texto o como epígrafe en otro lugar.

Del mismo modo, si usás funciones como `rotate`[3.10.23], `scale`[3.10.24] y `skew`[3.10.25], fijate de que esa transformación no tenga significado semántico o de que el significado esté disponible en otro lado para los usuarios de TA, por ejemplo en el texto alternativo[4.4.7] de una figura o en un epígrafe.

### 4.4.5 Artefactos

Algunas cosas de una página no tienen significado semántico y son irrelevantes para el contenido del documento. A estos elementos los llamamos *artefactos*. Los artefactos se ocultan a las TA y a la reutilización, y desaparecen durante el reflujo. Algunos ejemplos de artefactos:

- Los guiones que inserta la separación silábica automática al final de una línea
- Los encabezados y pies de página de cada página
- Una imagen de fondo de página puramente decorativa

En general, para que un documento se considere accesible, cada elemento de una página tiene que tener alguna forma de ser anunciado por la TA o ser un artefacto.

Typst marca automáticamente como artefactos muchos elementos de diseño, como encabezados, pies de página, fondos y primeros planos de página, y la separación silábica automática. Sin embargo, si querés agregar contenido puramente decorativo a tu documento, podés usar la función `pdf.artifact`[3.14.4/4] para marcar un fragmento de contenido como artefacto. Si no estás seguro de si tenés que marcar un elemento como artefacto, preguntate lo siguiente: ¿sería simplemente molesto que un lector de pantalla te anunciara

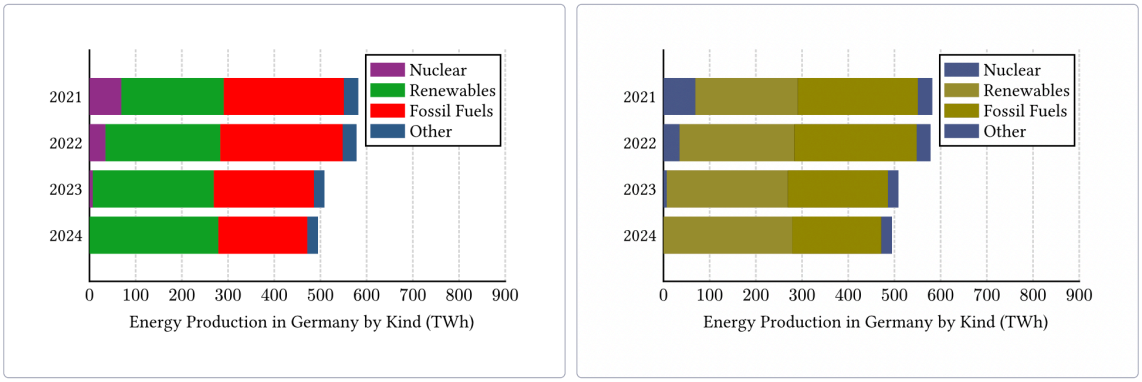
el elemento? Entonces puede ser un artefacto. Si, en cambio, podría ser útil que se anuncie, entonces no es un artefacto.

Por razones técnicas, una vez dentro de un artefacto, el contenido no puede volver a ser semántico. Para apilar artefactos y contenido semántico, usá `place`[3.10.19] para superponer los contenidos.

Tené en cuenta que Typst marca como artefactos las formas y trazados como `square`[3.11.11] y `circle`[3.11.2], mientras que su contenido sigue siendo semánticamente relevante y accesible para las TA. Si tus formas tienen un significado semántico, envolvelas en el elemento `figure`[3.6.7] para dar una descripción textual alternativa.

### 4.4.6 Uso del color y contraste

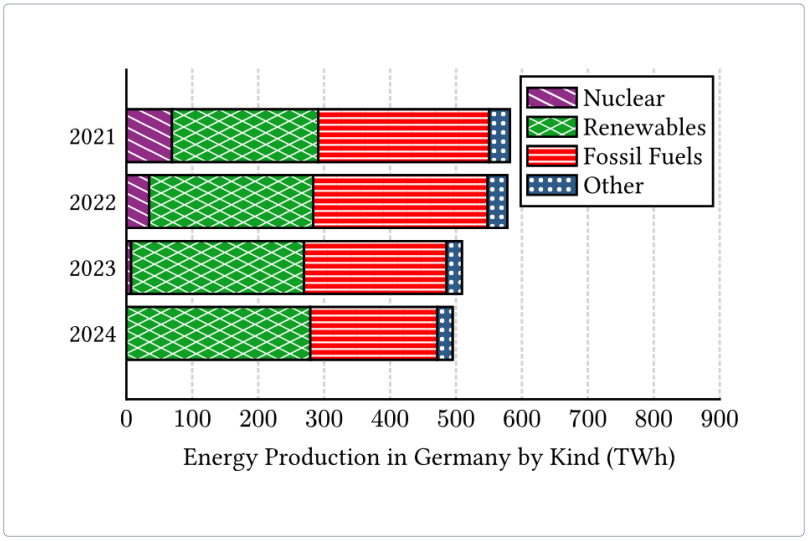
El Acceso universal no solo significa que tus documentos funcionen con TA, reflujo y reutilización, sino también que el acceso visual sea posible para todos, incluidas las personas con problemas de visión. No solo es frecuente que la vista empeore con la edad: una parte significativa de la población tiene problemas para diferenciar colores. Alrededor del 8 % de los hombres y el 0,5 % de las mujeres son daltónicos.




Esto significa que el color no debe ser la única manera de hacer accesible la información para los usuarios videntes en tus documentos. Como ejemplo, pensá en un gráfico de barras apiladas con varios segmentos de colores por barra. Nuestro ejemplo muestra un gráfico de la producción doméstica de energía en Alemania por tipo[A]. En la imagen podés ver el gráfico tal como se vería normalmente y una simulación de cómo lo verían las personas con daltonismo de tipo deuteranopía. Se ve que los dos pares formados por el primer y el último segmento parecen azules y el par central parece amarillento. El primer desafío para el usuario daltónico es, entonces, distinguir el límite entre la barra "Renovables" y la de "Combustibles fósiles". Después, tiene que llevar la cuenta de cuál es cuál solo por su orden, lo que suma carga mental. Una forma de hacer este gráfico todavía menos accesible sería que el orden de los segmentos no coincidiera con el de la leyenda.

¿Cómo podemos mejorar el gráfico? Primero, asegurarnos de que ninguna información se comunique únicamente mediante el color. Una posibilidad es agregarle un patrón a cada barra. Después, podemos ayudar al usuario a distinguir los límites de cada segmento agregando un borde de alto contraste. Nuestro gráfico podría quedar así:

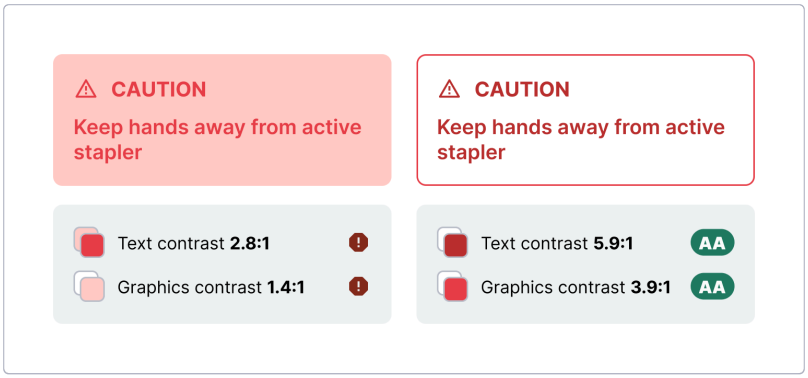
[A] Conjunto de datos de la Oficina Federal de Estadística de Alemania (Statistisches Bundesamt, Destatis). "Bruttostromerzeugung nach Energieträgern in Deutschland ab 1990", 2025, disponible bajo la *Data licence Germany – attribution – version 2.0*.




Se podría mejorar todavía más eligiendo colores que se diferencien para las personas con los tipos más comunes de daltonismo. También podrías seguir iterando el diseño con patrones de dos tonos, alineándolos con las barras o cambiando el uso de la tipografía.

Podés revisar tu diseño en la app web usando el simulador de daltonismo integrado. Para usarlo, abrí el menú "View" y elegí el modo deseado en el menú "Simulate color blindness". Si no usás nuestra app web, también podés usar otras herramientas de la web para simular la percepción del color de los distintos tipos de daltonismo[A].

Considerá también el contraste de color entre el fondo y el primer plano. Por ejemplo, si usás texto gris claro para las notas al pie, pueden volverse difíciles de leer. Otra situación que a menudo lleva a un contraste bajo es superponer texto sobre una imagen.




En nuestro ejemplo, vemos dos diseños de cuadros de aviso. Como estos cuadros buscan ayudar al usuario a evitar un peligro, es fundamental que realmente pueda leerlos. Sin embargo, en el primer cuadro el fondo es bastante claro, lo que dificulta distinguirlo. Peor todavía: el texto rojo es difícil de leer sobre el fondo rojo claro. El texto tiene una relación de contraste de 2,8:1, por debajo del mínimo de 4,5:1 que fijan las Pautas de Accesibilidad para el Contenido Web (WCAG). Del mismo modo, el cuadro tiene una relación de contraste de 1,4:1 con el fondo blanco de la página, por debajo del umbral de 3:1 para objetos gráficos.

Los colores del segundo ejemplo se ajustaron para cumplir los umbrales de contraste WCAG nivel AA. ¡Tendría que ser notablemente más fácil leer el texto del cuadro, incluso si tenés buena vista!

Existen herramientas para comparar cuánto contraste tiene un par de colores[B] como primer plano y fondo. La más común es la relación de contraste de color de WCAG. Para un tamaño de fuente dado, un par de colores puede no pasar la prueba, alcanzar el nivel AA o llegar al nivel AAA, más alto. Apuntá a un contraste AA como mínimo para todas tus combinaciones de colores.

[A] https://daltonlens.org/colorblindness-simulator
[B] https://webaim.org/resources/contrastchecker/

| Contenido | Relación AA | Relación AAA |
|---|---|---|
| Texto grande (≥18 pt, o en negrita y ≥14 pt) | 3:1 | 4.5:1 |
| Texto pequeño | 4.5:1 | 7:1 |
| Contenido que no es texto | 3:1 | 3:1 |

Tené en cuenta que los marcos de accesibilidad comunes, como WCAG, hacen una excepción para el texto puramente decorativo y los logotipos: por su carácter gráfico, pueden tener relaciones de contraste que no alcancen el contraste AA.

### 4.4.7 Representaciones textuales

Para dar soporte al uso de TA y a algunos flujos de reutilización, todos los elementos con significado semántico deben tener una representación textual. Pensalo en términos de Acceso universal: si un elemento no es un artefacto[4.4.5], tiene significado semántico. Pero si la TA no puede procesarlo, el significado semántico completo del documento no está disponible para los usuarios de TA. Por lo tanto, para brindar Acceso universal, usá los mecanismos integrados en Typst para proporcionar representaciones alternativas.

Cuando agregues una imagen, asegurate de usar el argumento `alt` de la función image[3.11.7/3.5] para describir lo que se ve en ella. Esta descripción alternativa (conocida a veces como texto alternativo) tiene que transmitir lo esencial de la imagen: pensá cómo se la describirías a un amigo si lo llamaras por teléfono. Para escribir buenas descripciones alternativas, considerá el contexto en el que aparece la imagen:

```
#image("heron.jpg", alt: "?")

Herons have feet with interdigital
webbing, allowing for good mobility
when swimming, and wings that span
up to 2.3 m.
```



Herons have feet with interdigital webbing, allowing for good mobility when swimming, and wings that span up to 2.3 m.

¿Cuál podría ser una buena descripción alternativa para esta imagen[A]? Veamos algunos ejemplos de lo que *no* hay que hacer:

- `"Imagen de una garza"`
  ❌ El lector de pantalla ya anuncia la imagen por su cuenta, así que decir que es una imagen es redundante. En este ejemplo, el usuario de TA escucharía "Imagen, Imagen de una garza".
- `"Un pájaro"`
  ❌ La descripción alternativa no es lo bastante específica. Por ejemplo, es relevante para el usuario que la imagen muestra una garza y que se ven tanto sus patas como sus alas.
- `"Garza gris en vuelo. Foto de Makasch1966 en Wikimedia Commons, licencia CC Attribution 4.0 International"`
  ❌ La descripción alternativa no debe incluir detalles que no se ven en la imagen, como atribuciones, chistes o metadatos. Tené presente que eso no es accesible para los usuarios videntes. Esa información va en otro lugar.
- `"Garza gris volando bajo, de derecha a izquierda. Tiene las patas extendidas y apuntando levemente hacia abajo, y toca un horizonte desenfocado donde se distingue un bosque`

[A] https://commons.wikimedia.org/wiki/File:Reiher_im_Flug.jpg

`oscuro. Las alas del ave están extendidas y forman un arco hacia arriba. En la esquina inferior izquierda de la imagen se ven ramas fuera de foco."`

❌ La descripción alternativa es demasiado extensa. Usá tu criterio y determiná qué tan importante es la imagen para el contenido. Pensá cuánto tiempo la miraría realmente un usuario vidente; tu texto alternativo debería demandar un esfuerzo parecido para "consumirlo". Por ejemplo, la descripción anatómica de arriba podría ser apropiada para una discusión más extensa en un libro de texto de zoología, mientras que la información de composición es útil al escribir sobre fotografía. El contexto que acompaña a la imagen de ejemplo es relativamente breve, así que escribí una descripción más corta.

En cambio, en el ejemplo dado, podrías usar este texto alternativo:

- `"Garza en vuelo con las patas y las alas extendidas"`

  ✅ Esta descripción alternativa describe la imagen, es relevante para el contexto y guarda proporción con su brevedad.

Hay recursos disponibles en la web para aprender más sobre cómo escribir buenas descripciones alternativas[A]. El requisito de agregar texto alternativo a las imágenes se aplica a todos los formatos de imagen. Actualmente, Typst no conserva en el documento compilado las etiquetas de una imagen PDF, incluso si el archivo PDF de la imagen era accesible por sí solo.

No uses imágenes de texto; del mismo modo, no uses operaciones de trazado para dibujar texto a mano. Typst no va a poder procesar el texto de las imágenes para hacerlo accesible de la misma manera que el texto nativo. Hay una excepción a esta regla: usá una imagen de texto cuando la apariencia del texto sea esencial para el significado semántico del documento y no pueda reproducirse de forma nativa con Typst. En ese caso, tenés que describir en la descripción alternativa tanto el contenido textual como las características visuales esenciales.

Igual que la función image, la función figure tiene un atributo `alt`[3.6.7/5.2]. Cuando usás este atributo, muchos lectores de pantalla y otras TA no anuncian el contenido de la figura y leen solo la descripción alternativa. Tu descripción alternativa tiene que ser lo bastante completa como para que el usuario de TA no necesite acceder al cuerpo de la figura. Usá la descripción alternativa solo si el contenido de la figura no es accesible de otra manera. Por ejemplo, no uses el atributo `alt` de una figura si contiene un elemento `table`, pero sí usalo si dentro usaste formas que tienen un significado semántico. Si especificás tanto `alt` como `caption`, las TA leerán ambos. Cuando tu figura contiene una imagen, poné la descripción alternativa en la imagen misma[3.11.7/3.5], no en la figura. No pongas ambas, porque la descripción de la figura reemplazaría a la de la imagen.

```
#figure(
  alt: "Star with a blue outline",
  curve(
    stroke: blue,
    curve.move((25pt, 0pt)),
    curve.line((10pt, 50pt)),
    curve.line((50pt, 20pt)),
    curve.line((0pt, 20pt)),
    curve.line((40pt, 50pt)),
    curve.close(),
  ),
)
```

Por último, podés especificar una descripción alternativa para las fórmulas con `math.equation`[3.8.16]. Describí tu fórmula como si la leyeras en voz alta en lenguaje natural. Actualmente, agregar una descripción alternativa es obligatorio para tener matemática accesible en todos los formatos de exportación. Si no agregás una descripción alternativa a tu fórmula, la exportación a PDF/UA-1 falla. En el futuro, Typst va a hacer accesible la matemática automáticamente en HTML y PDF 2.0 gracias a la tecnología MathML.

```
#math.equation(
  alt: "a squared plus b squared equals c squared",
```

[A] https://webaim.org/techniques/alttext/

```
  block: true,
  $ a^2 + b^2 = c^2 $,
)
```

Otro elemento que se representa a sí mismo como texto son los enlaces. Lo mejor es evitar textos de enlace poco descriptivos, como *acá* o *ir*. Estos textos también perjudican el posicionamiento en buscadores (SEO), si es algo que te importa en tu documento. En cambio, intentá que el enlace contenga texto sobre adónde apunta. Tené en cuenta que, a menos que busques el máximo nivel de accesibilidad, también está bien que el enlace no sea descriptivo en sí mismo mientras su propósito pueda entenderse por el contenido que lo rodea inmediatamente.

### 4.4.8 Idioma natural

Para que los lectores de pantalla pronuncien bien tu documento y el software de traducción funcione correctamente, tenés que indicar en qué idioma natural está escrito. Usá la regla `#set text(lang: "..")`[3.7.12/2.16] al comienzo de tu documento, o la capacidad de tu plantilla para configurar el idioma. Si no lo hacés, Typst va a asumir que tu contenido está en inglés. El idioma natural que elijas no solo afecta la accesibilidad, sino también cómo Typst aplica la separación silábica, qué convenciones tipográficas se aplican, las etiquetas de figuras y referencias y, en la app web, qué idioma se usa para el corrector ortográfico.

Si usás un idioma con variaciones importantes entre regiones, como el chino o el inglés, usá también el argumento `region`[3.7.12/2.17]. Por ejemplo, el chino hablado en Hong Kong se vería así:

```
#set text(lang: "zh", region: "HK")
```

Para especificar tu idioma, usá los códigos ISO 639. Para las regiones, usá el código ISO 3166-1 alpha-2[A]. ISO 639 contiene tres variantes: una para códigos de idioma de dos letras, como "de" para alemán (ISO 639-1)[B], y dos para códigos de tres letras, como "deu" (ISO 639-2[C] e ISO 639-3[D]). Si tu idioma tiene un código ISO 639-1 de dos letras, preferilo siempre. ISO 639-2 e ISO 639-3 comparten la mayoría de los códigos, pero hay algunas diferencias. Cuando el código de tu idioma difiere entre los dos estándares, usá ISO 639-2 al exportar a PDF 1.7 (el predeterminado de Typst) y anteriores, e ISO 639-3 para exportar a PDF 2.0 y HTML.

Hay tres códigos de idioma especiales definidos tanto por ISO 639-2 como por ISO 639-3 que podés usar cuando es difícil indicar un código de idioma normal:

- `zxx` para texto que no está en un idioma natural
- `und` para texto cuyo idioma natural no podés determinar
- `mis` para texto en idiomas a los que no se les asignó un código de idioma

Si tu documento contiene texto en varios idiomas, podés usar la función text o una regla set de text con alcance acotado para encerrar los fragmentos en otros idiomas:

```
This is #text(lang: "fr")[français].

#[
  #set text(lang: "es")
  Este es un fragmento más largo
  del texto en español.
]
```

This is français.

Este es un fragmento más largo del texto en español.

[A] https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2
[B] https://en.wikipedia.org/wiki/List_of_ISO_639_language_codes
[C] https://en.wikipedia.org/wiki/List_of_ISO_639-2_codes
[D] https://en.wikipedia.org/wiki/List_of_ISO_639-3_codes

### 4.4.9 Título del documento y títulos de sección

Ponerle título a tu documento facilita encontrarlo y navegar entre él y otros documentos, tanto para los usuarios de TA como para los usuarios comunes de visores de PDF. Por eso los estándares de accesibilidad como WCAG y PDF/UA exigen que establezcas un título legible por máquina para tu documento.

Para hacerlo en Typst, colocá esta regla set en tu documento antes de cualquier contenido:

```
#set document(title: [GlorboCorp Q1 2023 Revenue Report])
```

Esto establece el título en los metadatos del documento[3.6.5/3.3] y en la barra de título del visor de PDF o del navegador web. Si esto produce un error al usar una plantilla, fijate si tu plantilla ofrece otra forma de establecer el título del documento.

Lo más probable es que también quieras incluir el título de forma visible en tu documento. Para eso, usá el elemento `title`[3.6.21]. Cuando llamás al elemento title sin argumentos, imprime el contenido que estableciste como título del documento. Como alternativa, podés personalizar el título pasando contenido como argumento posicional de cuerpo. No uses el elemento title más de una vez en tu documento.

Nunca uses un título de sección (heading) para el título del documento; usá el elemento title. Si tenés experiencia con HTML, es importante recordar que la semántica del elemento heading en Typst difiere de la de los títulos de HTML. En Typst se recomienda usar varios títulos de primer nivel para las secciones. Al exportar a HTML, un title[3.6.21] se serializa como una etiqueta `h1`, mientras que un título de primer nivel[3.6.9/5.1] se serializa como una etiqueta `h2`. En la exportación a PDF, el título y los títulos de sección se etiquetan correctamente según la versión de PDF elegida.

Es importante que la secuencia de títulos sea correlativa: nunca saltees un nivel al ir más profundo. Esto significa que un título de tercer nivel tiene que ir seguido de uno de nivel cuatro o inferior, pero nunca de uno de nivel cinco o superior.

```
// ❌ Don't do this:
= First level heading
=== Third level heading
```

Tené en cuenta que, para pasar la verificación automática de accesibilidad de Adobe Acrobat[A], los documentos de 21 páginas o más deben contener títulos en el esquema (marcadores).

### 4.4.10 Estándares y legislación de accesibilidad

Typst puede ayudarte a afirmar que tu documento es accesible verificándolo contra estándares internacionales. Para la exportación a PDF hay varios estándares de archivos accesibles, sobre todo el estándar PDF/UA. Su primera parte (PDF/UA-1) ya está soportada por Typst, mientras que el soporte para la segunda (PDF/UA-2) está planeado para el futuro. A continuación encontrás una explicación de todos los estándares relevantes:

- **PDF etiquetado (Tagged PDF):** Los PDF etiquetados contienen datos legibles por máquina sobre la estructura semántica de un documento, que las TA pueden interpretar. Typst escribe PDF etiquetados de forma predeterminada, pero tené presente que solo puede escribir las etiquetas adecuadas si conoce la estructura semántica de tu documento. Consultá la sección *Mantener la semántica*[4.4.2] para aprender a usar los elementos de Typst para comunicar semántica. Para ofrecer Acceso universal, también sos responsable de proporcionar por tu cuenta una representación textual del contenido que no es texto.
- **PDF/UA-1:** El estándar PDF/UA explica cómo escribir un archivo PDF 1.7 optimizado para el Acceso universal. Implica PDF etiquetado, exige descripciones alternativas para imágenes y matemática, requiere un título del documento e introduce reglas sobre cómo estructurar los contenidos, como las tablas. Si seguís esta guía, ya estás evitando la mayoría de los errores del compilador que pueden ocurrir durante la exportación a PDF/UA-1.

[A] https://helpx.adobe.com/acrobat/using/create-verify-pdf-accessibility.html#Bookmarks

- **PDF/UA-2:** También existe la parte más reciente, PDF/UA-2, que apunta a archivos PDF 2.0. Mejora la accesibilidad de la matemática y de algunos elementos semánticos. El soporte para PDF/UA-2 todavía no está disponible en Typst, pero está planeado.
- **Well Tagged PDF (WTPDF):** Es un estándar de la industria muy similar a PDF/UA-2. Igual que PDF/UA-2, Typst no lo soporta actualmente. Originalmente se redactó porque las dos partes de la especificación PDF/UA solo estaban disponibles a un costo alto a través de la Organización Internacional de Normalización. Por eso, WTPDF[A] se diseñó de modo que todos los archivos conformes también puedan declarar conformidad con PDF/UA-2. Hoy, ambas partes de la especificación PDF/UA están disponibles sin cargo[B], lo que reduce la relevancia de WTPDF.
- **PDF/A-1a:** El estándar PDF/A describe cómo producir archivos PDF aptos para archivo a largo plazo. Las partes uno a tres del estándar PDF/A tienen varios niveles de conformidad. El nivel más estricto, A, contiene reglas de accesibilidad, ya que solo los archivos que las cumplen siguen siendo utilizables para la mayor variedad de personas en un futuro lejano. El nivel A implica conformidad con PDF etiquetado y te obliga a proporcionar descripciones alternativas para las imágenes. También se aplican otras reglas de PDF/A no relacionadas con la accesibilidad, por ejemplo sobre transparencia, colores y más. Esta parte del estándar PDF/A se basa en la desactualizada especificación PDF 1.4. Usala solo si tu ámbito lo exige o si necesitás un archivo muy compatible. De lo contrario, PDF/UA-1 y la segunda y tercera parte de PDF/A son mejores alternativas.
- **PDF/A-2a** y **PDF/A-3a:** Igual que la primera parte de PDF/A, estos estándares se enfocan en crear archivos aptos para archivo y almacenamiento a largo plazo. Ambos apuntan a la versión más nueva, PDF 1.7, en lugar de PDF 1.4. También acá el nivel de conformidad más estricto, A, contiene reglas de accesibilidad. Además de las reglas de PDF/A-1a, estos estándares prohíben el uso de caracteres de la Zona de uso privado de Unicode[C], cuyo significado no está definido universalmente. Las mejoras respecto de PDF/A-1 incluyen la posibilidad de usar transparencia y un mejor reflujo. Al elegir entre estas dos partes del estándar PDF/A, elegí PDF/A-2a a menos que necesites adjuntar[3.14.4/5] otros archivos. Tené en cuenta que el nivel de conformidad A se eliminó de PDF/A-4 a favor del estándar PDF/UA específico.

La página de referencia de PDF[3.14.4/3.2] contiene más información sobre cada estándar soportado. Para habilitar PDF/UA, PDF/A-2a o PDF/A-3a, usá el indicador correspondiente en la CLI[3.14.4/1.1] o usá el menú desplegable de exportación y hacé clic en PDF en la app web. Podés combinar estándares PDF/A y PDF/UA que sean compatibles. Para documentos centrados en la accesibilidad, recomendamos combinar PDF/UA-1 con PDF/A-2a o PDF/A-3a.

Cuando seleccionás uno de estos estándares para la exportación a PDF, Typst detecta si estás infringiendo sus reglas y hace fallar la exportación con un mensaje de error descriptivo. Para la verificación de accesibilidad más estricta disponible actualmente, elegí PDF/UA-1. No desactives el etiquetado a menos que tengas una buena razón, ya que las etiquetas aportan una base de accesibilidad en todos los documentos que exportás.

Quizás ya notaste que algunos de los factores del Acceso universal son difíciles de verificar automáticamente. Por ejemplo, Typst actualmente no comprueba automáticamente que tus contrastes de color sean suficientes ni que el idioma natural configurado coincida con el idioma natural real (aunque la cantidad de errores del corrector ortográfico debería darte una pista si usás la app web). Hay dos estándares internacionales que abordan con más detalle algunos de estos factores humanos:

- Las **Pautas de Accesibilidad para el Contenido Web (WCAG)**[D]: Diseñadas por el W3C, un gran consorcio internacional detrás de las tecnologías que hacen funcionar internet, describen cómo hacer accesible un sitio web. Todas estas reglas son aplicables a la salida HTML de Typst, y muchas se aplican a su salida PDF. WCAG separa sus reglas en tres niveles: A, AA y AAA. Se recomienda que los documentos normales apunten a AA. Si tenés altas exigencias de Acceso universal, también podés considerar los criterios de éxito AAA. Sin embargo, Typst todavía no expone todas las funciones de PDF necesarias para cumplir AAA, por ejemplo una forma accesible para las TA de definir las expansiones de abreviaturas.

[A] https://pdfa.org/wtpdf/
[B] https://pdfa.org/sponsored-standards/
[C] https://en.wikipedia.org/wiki/Private_Use_Areas
[D] https://www.w3.org/TR/WCAG21/
[A] https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf

- La **Norma Europea EN 301 549**[A]: Su sección 9 describe cómo crear sitios web accesibles y su sección 10 describe qué reglas se aplican a los documentos que no son web, incluidos los PDF creados por Typst. Señala qué cláusulas de WCAG son también aplicables a los PDF. Cumplir esta norma es un buen comienzo para cumplir las leyes de accesibilidad de la UE y nacionales.

Tené en cuenta que, para cumplir con EN 301 549 y las disposiciones relevantes de WCAG, tu documento debe estar etiquetado. Si apuntás a la conformidad, te sugerimos firmemente usar PDF/UA-1 para la exportación, de modo de automatizar muchas de las comprobaciones de los criterios de éxito que contiene.

Muchos territorios tienen legislación de accesibilidad que en algunas circunstancias exige crear archivos accesibles. Estos son solo algunos ejemplos:

- **Ley Europea de Accesibilidad (EAA, UE 2019/882)**[B]: Este reglamento se aplica a libros electrónicos, servicios bancarios para consumidores, servicios de comercio electrónico y más. Exige que los archivos distribuidos en estas aplicaciones sean accesibles.
- **Ley de Estadounidenses con Discapacidades (ADA)**: El Departamento de Justicia va a exigir a las organizaciones del sector público que proporcionen archivos[C] conformes con WCAG bajo el Título II de la ADA a partir de 2026. Asimismo, las organizaciones privadas pueden ser consideradas responsables[D] por servicios digitales inaccesibles según la ADA y las leyes estatales.

Usar esta guía puede ayudarte a cumplir con cualquiera de las dos regulaciones.

### 4.4.11 Pruebas de accesibilidad

Para probar si tu documento PDF es accesible, podés usar herramientas automáticas y pruebas manuales. Algunos estándares, como PDF/UA y PDF/A, pueden verificarse exclusivamente con herramientas automáticas, mientras que algunas reglas de WCAG y otros estándares requieren comprobaciones manuales. Typst supera automáticamente muchas de las comprobaciones automatizables cuando el PDF etiquetado está habilitado. Para muchas otras comprobaciones automatizables, podés habilitar la exportación a PDF/UA-1 para que Typst las ejecute. Las herramientas automáticas solo pueden ofrecer una base de accesibilidad. Para un Acceso universal de verdad, lo mejor es que pruebes el documento vos mismo con una TA.

Esta es una lista de verificadores automáticos para probar la conformidad:

- **veraPDF**[E]**:** Esta herramienta de código abierto puede comprobar si tu archivo PDF cumple con las partes de los estándares PDF/A y PDF/UA con las que declaró conformidad. Usala si elegiste uno de estos estándares al exportar. Las fallas se consideran errores de Typst y deberían reportarse en GitHub[F].
- **PDF Accessibility Checker (PAC)**[G]**:** El programa gratuito PAC comprueba si tu documento cumple las reglas de PDF/UA y WCAG. Cuando recibís un error grave en la pestaña PDF/UA, se considera un error de Typst y debería reportarse en GitHub[H]. Las advertencias en las pestañas PDF/UA y Quality pueden ser errores, problemas de tu documento o ninguna de las dos cosas. Consultá en el foro[I] o en Discord[J] si tenés dudas. Los errores y advertencias en la pestaña WCAG indican problemas de tu documento.
- **Accessibility Check en Adobe Acrobat Pro**[K]**:** El verificador de accesibilidad de la versión paga de Adobe Acrobat revisa todos los documentos PDF en busca de problemas. En lugar de comprobar el cumplimiento de un estándar internacional o de la industria conocido, Adobe creó su propio conjunto de pruebas. Como las reglas detrás de estas pruebas a veces contradicen estándares internacionales como PDF/UA, es esperable que algunas comprobaciones de Acrobat fallen para documentos de Typst[L]. Otras comprobaciones, como la de contraste, son útiles e indican problemas de tu documento.

[B] https://eur-lex.europa.eu/eli/dir/2019/882/oj
[C] https://www.ada.gov/law-and-regs/regulations/title-ii-2010-regulations/
[D] https://www.boia.org/blog/the-robles-v.-dominos-settlement-and-why-it-matters
[E] https://verapdf.org
[F] https://github.com/typst/typst/issues
[G] https://pac.pdf-accessibility.org/en
[H] https://github.com/typst/typst/issues
[I] https://forum.typst.app/
[J] https://discord.gg/2uDybryKPe
[K] https://helpx.adobe.com/acrobat/using/create-verify-pdf-accessibility.html
[L] Por ejemplo, al usar notas al pie, es esperable que falle la comprobación "Lbl and LBody must be children of LI" de la sección "List".

Para las comprobaciones manuales, podés empezar con una lista de verificación. Si tu organización hace hincapié en la accesibilidad, a veces tiene la propia. Si no, podés probar listas de universidades como la Universität Bremen (en inglés)[A] o de gobiernos, como la de Canadá[B] o la de la Administración del Seguro Social de EE. UU.[C]. Aunque estas listas difieren en su nivel de detalle, todas cubren las comprobaciones manuales más esenciales. Muchas de las comprobaciones técnicas que incluyen pueden omitirse si elegís la exportación a PDF/UA-1 en Typst. Si no sabés qué lista usar, elegí una de una organización culturalmente parecida a la tuya.

Sin embargo, para alcanzar el mayor nivel de accesibilidad en documentos de amplia circulación, considerá revisar tu documento con una TA. Aunque hay muchos productos de TA y visores de PDF, normalmente alcanza con probar una sola combinación. Cuál es la mejor depende de tu sistema operativo:

- Windows: Probá con Adobe Acrobat[D] y NVDA[E]. NVDA es software libre y de código abierto. Hay una versión gratuita de Acrobat disponible.
- macOS: Probá con Adobe Acrobat[F] y VoiceOver[G]. VoiceOver es el lector de pantalla integrado en macOS y otras plataformas de Apple.
- Linux: Probá con Evince[H] u Okular[I] y Orca[J]. Las tres herramientas son software libre y de código abierto. Sin embargo, el soporte de TA en las plataformas Linux está por detrás del disponible en Windows y macOS. Del mismo modo, Evince y Okular tienen menos soporte de accesibilidad que Acrobat. Te sugerimos firmemente probar con Acrobat.

Cuando recién empieces a hacer pruebas, considerá completar el programa de capacitación interactivo que ofrezca tu lector de pantalla, si lo tiene. Ganar confianza con un lector de pantalla te ayuda a vivir tu documento como lo haría un usuario habitual de lector de pantalla. Al revisar tu documento, verificá que no solo haga accesible toda la información disponible para un usuario vidente, sino que además sea fácil de navegar. La experiencia de tus usuarios va a variar según la combinación de visor de PDF y TA que usen.

### 4.4.12 Límites y consideraciones sobre los formatos de exportación

Aun cuando diseñes tu documento pensando en la accesibilidad, tenés que conocer las limitaciones de tu formato de exportación. En esencia, dar soporte de TA a los archivos PDF es más difícil que a otros formatos, como HTML. El PDF se concibió en 1993 para renderizar con fidelidad documentos impresos en una computadora. Las funciones de accesibilidad se agregaron por primera vez con PDF 1.4 en 2001 y mejoraron en PDF 1.5 (2003) y PDF 2.0 (2017). En cambio, HTML ofrece un modelo semántico más rico y más flexibilidad, por lo que el soporte de TA en los navegadores generalmente supera lo posible en los visores de PDF.

Tené en cuenta también que los archivos PDF son mayormente estáticos. Esto te permite ignorar muchas reglas de WCAG y EN 301 549 pensadas para contenido interactivo y multimedia. Sin embargo, la falta de interactividad también hace más difícil que los usuarios adapten el diseño de un documento a sus necesidades.

Por ejemplo, el criterio de éxito 1.4.12 de WCAG[K] (codificado en la cláusula 10.1.4.12 de EN 301 549) establece que el usuario debe poder aumentar el espaciado entre caracteres, letras, líneas y párrafos a valores muy amplios. Esto beneficia a usuarios con visión reducida o dislexia. El criterio de éxito no te exige diseñar tu documento con esos parámetros de diseño. Solo exige un mecanismo mediante el cual los usuarios puedan aumentarlos al leer el documento. En los archivos HTML es fácil cumplir este criterio porque el navegador le permite al usuario anular estos parámetros de espaciado en una página. En PDF, la situación es más matizada: en teoría, Typst agrega a un archivo etiquetas y atributos pensados para el reflujo. Un lector de PDF, al

[A] https://www.uni-bremen.de/fileadmin/user_upload/universitaet/Digitale_Transformation/Projekt_BALLON/Checklisten/2._Auflage_englisch/Checklist_for_accessible_PDF_ENG-US_ver2.pdf
[B] https://a11y.canada.ca/en/pdf-accessibility-checklist/
[C] https://www.ssa.gov/accessibility/checklists/PDF_508_Compliance_Checklist.pdf
[D] https://www.adobe.com/acrobat.html
[E] https://www.nvaccess.org/download/
[F] https://www.adobe.com/acrobat.html
[G] https://support.apple.com/guide/voiceover/welcome/mac
[H] https://wiki.gnome.org/Apps/Evince/
[I] https://okular.kde.org/
[J] https://orca.gnome.org
[K] https://www.w3.org/WAI/WCAG21/Understanding/text-spacing.html

reflujar, podría permitirle al usuario aumentar los espaciados más allá de lo codificado en esas etiquetas. En la práctica, no conocemos ningún visor de PDF con esa función. En cambio, este criterio de éxito puede satisfacerse reutilizando el PDF como un archivo HTML y abriéndolo en un navegador.

En la práctica, incluso si tu archivo cumple técnicamente, no podés esperar que tus usuarios conozcan estas soluciones alternativas. Por lo tanto, si apuntás a cumplir los estándares más altos de Acceso universal, considerá distribuir una versión HTML de tu documento junto con el PDF. Exportá este archivo directamente con la exportación a HTML[3.14-3] de Typst (en vista previa). Aunque la exportación a HTML no conserva muchos aspectos de tu diseño visual, produce un archivo que aprovecha HTML semántico y tecnologías como Digital Publishing ARIA[A] para ofrecer Acceso universal. Va a tener una calidad superior a la de un PDF reutilizado como HTML.

Por último, tené presente que los PDF están pensados para imprimirse. Por lo tanto, no des por sentado que las funciones interactivas, como los enlaces, estén disponibles para los usuarios que decidan imprimir tu documento.

Como se mencionó antes, los archivos creados con la exportación a PNG y SVG no son accesibles.

[A] https://www.w3.org/TR/dpub-aria-1.1/

# Changelog

Learn what has changed in the latest Typst releases and move your documents forward. This section documents all changes to Typst since its initial public release.

INFO

Some changelog items contain references like this one: #5017. These refer to a *pull request* or *issue* on GitHub[A] related to the change. If the reference refers to another repository than `typst/typst`, the repository is explicitly listed like this: typst/hayagriva#350.

## Versions


0.15.1 ..... 653
0.15.0 ..... 654
0.14.2 ..... 666
0.14.1 ..... 667
0.14.0 ..... 669
0.13.1 ..... 677
0.13.0 ..... 678
0.12.0 ..... 684
0.11.1 ..... 691
0.11.0 ..... 692
0.10.0 ..... 697
0.9.0 ..... 699
0.8.0 ..... 702
0.7.0 ..... 704
0.6.0 ..... 707
0.5.0 ..... 709
0.4.0 ..... 711
0.3.0 ..... 713
0.2.0 ..... 715
0.1.0 ..... 716
Earlier ..... 718


[A] https://github.com/typst/typst

# Typst 0.15.1 (July 17, 2026)

## Library

### Text

- Updated New Computer Modern fonts to version 8.1.1, fixing a bug where the regular weight of the math font was still using the old calligraphic letterforms[3.8.27/1.6] that were supposed to live in stylistic set[3.7.12/2.24] 6 #8552

### Math

- Fixed a regression where alignment points did not work correctly when placed within `lr`[3.8.20/2.1] / matched delimiters #8566
- Fixed a regression where op[3.8.26] elements could be vertically misaligned #8546

### Layout

- Fixed a bug where gaps could appear in multi-page lists with `number-align`[3.6.11/3.9] / `marker-align`[3.6.2/3.6] set to an alignment with a vertical component #8649

## Export

### SVG

- Fixed that inline SVGs resulting from `html.frame`[3.14.3/5] elements did not respect pretty printing #8535

### Bundle

- In the watch server, the automatic selection of appropriate `Content-Type` headers based on file extensions was expanded to include more commonly used file types (e.g. `json` files) #8650
- The error for PNG/SVG documents with multiple pages is now only raised if it persists until the final document iteration #8618

## Command Line Interface

- The `typst eval` subcommand will now exit with code `1` when the provided expression fails to evaluate #8623

# Typst 0.15.0 (June 15, 2026)

This section documents all changes to the Typst language and compiler between Typst 0.14.2 and 0.15.0. If you are migrating an existing document to Typst 0.15, make sure to check out the Migration guide. It walks you through changes you may need to make to your existing documents to ensure compatibility with Typst 0.15.

## Highlights

- Typst now supports variable fonts[3.7.12/2.33]
- HTML export now supports equations out of the box via MathML
- With the new, experimental *bundle*[3.14.2] export target, a single Typst project can output multiple files (e.g. a multi-page website)
- A single document can now contain multiple bibliographies[3.6.1/4.5]
- Typst can now target multiple PDF standards[3.14.4/3.2] at once
- The new `within`[3.5.26/3.5] selector simplifies many introspection use cases
- The new `divider`[3.6.4] element represents a thematic break that templates can style
- Spot colors[3.11.3/5.1] enable use of custom pigments in offset printing
- With the new file `path`[3.5.22] type, project-relative paths can be passed to packages
- The new, more general `typst eval` CLI subcommand supersedes `typst query`
- Layout convergence[3.4.4] issues now result in detailed diagnostics
- Two long-standing list layout issues with marker alignment and centering were fixed
- Paragraph handling in HTML export is improved, preventing unexpected paragraphs from appearing
- This documentation now has a print version[A]

## Language

### Syntax

- File paths[3.5.22] (e.g. in imports or `image`[3.11.7] function calls) may not contain backslashes anymore; instead forward slashes must be used #7688 **(Breaking change)**
- Added hints for invalid characters in code mode #7752
- Added hint when trying to use a unary operator directly in an embedded expression using a hash (e.g. `#-30deg`) #8069
- Fixed potential stack overflow crashes by enforcing a maximum parsing depth #7207
- Fixed incremental parsing of unclosed strings #8067

### Styling

- Text show rules now have tracebacks that include the matched text #8088
- Fixed a crash with text show rules that match on multi-character symbols #8011

### Scripting

- Extended hint when built-in definitions are shadowed to set and show rules #7131
- Added hint when trying to spread one or multiple dictionaries into an array #7798
- Improved diagnostics for invalid method calls #7865
- Improved hint for unknown variables in math that are available in `std` #7810
- Fixed a misleading error message when trying to assign to a temporary return value #8212

## Library

### Foundations

- Added file `path`[3.5.22] type that is now accepted in all places where paths were previously only represented as strings #7555

[A] https://github.com/typst/typst/releases/download/v0.15.0/typst-documentation.pdf

  - A path constructed in one file can be used in another file, but will be resolved relative to its original file
  - Likewise, paths can be passed across package boundaries
  - The initial path type is very minimal, but additional features like file existence checks or directory walking are planned
- Collections
  - Added `map` and `filter` functions on dictionaries[3.5.12/3.8] and arguments[3.5.1/4.5] #7284
  - Named arguments on `arguments`[3.5.1] values are now accessible with field access syntax #8407
  - The `slice` functions on strings[3.5.28/4.5] and arrays[3.5.2/3.9] will now error if passing both an `end` and a `count` #7238 **(Minor breaking change)**
  - Added `inclusive` parameter[3.5.2/3.13.3] to `range` function #8345
- Calculation
  - Added `asinh`[3.5.9/2.16], `acosh`[3.5.9/2.17], `atanh`[3.5.9/2.18], and `erf`[3.5.9/2.21] functions to `calc` module #6967 #7655
  - Added `int.min` and `int.max` constants for the minimum and maximum representable integer, respectively #8445
  - Fixed behavior of `quo`[3.5.9/2.40] for negative integers #7392
  - Fixed potential overflows in `norm`[3.5.9/2.41], `abs`[3.5.9/2.1], `gcd`[3.5.9/2.25], and `lcm`[3.5.9/2.26] #8156 #8352
  - Floating-point calculations are now consistently deterministic across platforms #7712
- Date & time handling
  - The `offset` parameter[3.5.10/5.1.1] of `datetime.today` now accepts durations[3.5.13] as an alternative to integers, allowing for sub-hour precision offsets #7691
  - Addition and subtraction of datetimes[3.5.10] and durations[3.5.13] now retains precision instead of clamping to full days in some cases #7843
  - The `datetime`[3.5.10] constructor now emits more precise errors when components are missing #7371
- Conversions
  - Added `base` parameter[3.5.17/3.2] to `int` constructor to configure in which base to parse a string #7386
  - Fixed that the `base` parameter[3.5.28/3.2] of the `str` constructor was accepted for non-integer values if its value was `10` #7237 **(Minor breaking change)**
  - Added hint when trying to construct a string[3.5.28/3] with base 1 #7237
- The `panic`[3.5.21] function now displays strings as-is instead of showing their `repr`[3.5.25], making it more suitable for friendly, user-facing messages #8211
- Changed `repr`[3.5.25] of styles and locations[3.12.4] to be more distinct #7364

## Model

- Added `divider`[3.6.4] element representing a thematic break that templates can style #7982
- Bundle-related elements
  - The `document`[3.6.5] element can now be constructed to produce individual documents in bundle export #7964
  - Added `path`[3.6.5/3.1] and `format`[3.6.5/3.2] parameters to `document` element #7964
  - Added experimental `asset`[3.14.2/5] element #7964
- Bibliography management
  - A single document can now contain multiple bibliographies #8307
  - Added `target` parameter[3.6.1/4.5] to `bibliography` element to configure which citation is picked up by which bibliography #8307
  - Added `group` parameter[3.6.1/4.6] to `bibliography` element to configure how numbers are shared/reset across bibliographies #8307
  - Added support for numeric values for the `month` key in `.bib` files typst/biblatex#104
  - Added support for BibLaTeX name options in `.bib` files typst/biblatex#90
  - Added support for propagating non-numeric `volume` fields in `.bib` files to bibliographies typst/hayagriva#478
  - Improved sorting in bibliographies to take into account language conventions typst/hayagriva#314
  - Improved interoperability with CSL styles; for a full listing of changes, review the Hayagriva 0.10.0 changelog[A]
  - Added support for setting directors on videos without a parent in Hayagriva YAML files typst/hayagriva#450

[A] https://github.com/typst/hayagriva/blob/v0.10.0/CHANGELOG.md

  - Improved handling of `Anthos` entries in Hayagriva YAML files by treating them as `chapters` in CSL typst/hayagriva#427
- Footnotes
  - The link of a `footnote`[3.6.8] is now within the superscript instead of around it, improving PDF tagging and HTML output #7637
  - The thickness of the default footnote separator[3.6.8/4.1.2] is now specified in font-relative instead of absolute units #7950
- Numbering
  - Added `numbering`[3.6.12] support for Armenian numerals #7529, Arabic Abjad numerals typst/codex#170, and Tibetan numerals typst/codex#163
  - Greek `numbering`[3.6.12] now uses the modern Greek style #7632
- The `par.first-line-indent`[3.6.14/4.6] property will now *fold*, meaning that partial dictionaries across different set rules or `par`[3.6.14] calls are combined #7747
- Added `list.marker-align`[3.6.2/3.6] property for defining how to align list markers #7895
  - When omitted, it will default to the new baseline alignment (vertically), combined with `end` alignment (horizontally)

## Text

- Added support for variable fonts #8425
  - The well-known variation axes `ital`, `slnt`, `wght`, `wdth`, and `opsz` are automatically set based on text `weight`[3.7.12/2.4], `stretch`[3.7.12/2.5], `style`[3.7.12/2.3], and `size`[3.7.12/2.6]
  - Custom variations can be configured via the new `variations` parameter[3.7.12/2.33] of the `text` function
  - When using a variable font with Typst, the suffixes "Variable", "Var", and "VF" should be omitted as Typst trims them to unify static and variable fonts into a single family #8444 **(Minor breaking change)**
- Font features
  - The `text.alternates`[3.7.12/2.23] parameter now accepts an integer in addition to a boolean to select stylistic alternates other than `0` and `1` #7423
  - Parsing of tag names in `text.features`[3.7.12/2.32] is now more strict #7423 **(Minor breaking change)**
- Fixed that `context text.font` did not reflect the `covers` field #7354
- Fixed uneven CJK-Latin spacing[3.7.12/2.11] in justified paragraphs #7606
- Fixed a bug where the `lorem`[3.7.3] function would not produce the exact number of requested words #7877
- Improved translations for Swedish #7166, Portuguese #7088, Czech #7318, Latvian #7701, Slovak #7734, Polish #7734, Vietnamese #7774, Finnish #7988 #7989, and Welsh #7811
- Added font exception to avoid *SimSun-ExtB* being incorrectly merged with *SimSun* #8042
- Updated New Computer Modern fonts to version 8.1.0 #7597 #7663 #8164 #8330 #8435
  - This update changes the default look of calligraphic letterforms[3.8.27/1.6] in the math font; the previous style can be restored through `show math.equation: set text(stylistic-set: 6)`
- Updated Unicode components #8406
  - In particular, this fixed an issue with linebreaking of guillemets

## Math

- Layout
  - Improved layout of under/over elements like `underbrace`[3.8.15/1.3] #7299
  - Slightly improved spacing around `math.op`[3.8.26] elements #7429
  - In which situations *cramped* styles (with tighter spacing) are applied is now fully consistent with TeX and MathML Core #8082
  - The `size` parameter[3.8.20/2.1.1] of the `lr` function now consistently applies to middle delimiters[3.8.20/2.2] in the same way it does to outer delimiters #7435
  - The `size` parameter[3.8.20/2.1.1] of the `lr` function now resolves relatively to the height of just the inner content; it does not take the delimiters into account anymore #7605 **(Minor breaking change)**
  - Glyph stretching[3.8.24] is now always relative to the base glyph rather than a potentially already scaled version (e.g. due to display sizing) #7435 **(Minor breaking change)**
  - Fixed left/right alignment not being applied correctly due to spacing next to alignment points #7473 #7435
  - The `binom`[3.8.11] element now uses different OpenType constants for layout; though this does not lead to visible changes with most fonts #7685

  - The default length and stroke width of `math.cancel`[3.8.12] lines is now specified in font-relative instead of absolute units #7241
  - Fixed potential misalignment in `cases`[3.8.13] function #7662
- Text handling
  - Improved handling of multi-character symbols in math #6929
  - Fixed that some glyphs did not stretch correctly in script sizes #8171
  - Fraction, root, and under/over lines now respect `text.stroke`[3.7.12/2.8] #7540
  - Accents in math are now always rendered in front of their base if they overlap #7733
- The `class`[3.8.14] function now applies the class only to its direct body rather than recursively #8328 **(Minor breaking change)**
- More delimiter symbols (e.g. `chevron.l`) are now callable to produce an `lr`[3.8.20/2.1] element #7228 **(Minor breaking change)**
- Fixed various bugs with rendering of mathematical expressions that look like function calls but in reality aren't (e.g. `$pi(1, 2)$`, since `pi` is not a function) #7865
- Fixed a bug with ordering of primes and nested attachments #7647

## Symbols

- Added many new symbols and variants. View the codex 0.3.0 changelog[A] for a full listing.

## Layout

- Baseline information is now retained in many more parts of the layout engine #8150 **(Breaking change)**
  - In particular, text contained in a `box`[3.10.5] with an inset is now aligned with the text surrounding the `box`
  - This also fixes a bug where wrapping an inline equation in a `box` would shift its baseline
  - Similarly, using a `block`[3.10.4] in an equation will keep the baseline intact
  - Last but not least, the marker/number and item of a `list`[3.6.2] or `enum`[3.6.11] are now properly baseline-aligned with the first line of the item even if the item is vertically larger than a normal line #7895
- Centering something in a list now centers based on the full available width rather than based on the maximum width of other list content #7895
- Page layout
  - Added `bleed` parameter[3.10.17/4.6] to `page` element to set up bleed margins #6357
  - Fixed the size of `"us-executive"` paper[3.10.17/4.1] #7869
  - Added warning for `show page` rules as they are unsupported #7445
- Paragraph layout
  - Fixed a bug where justified text could accidentally protrude into the margin when it ends with certain kinds of characters (e.g. a zero-width space) #8415
  - Fixed a bug where first-line indent[3.6.14/4.6] was applied at the start of a column even if `all: false` is set #7722
- Added support for spacing that is both weak[3.10.27/2.2] and fractional[3.10.9] #6833

## Visualize

- Added support for spot colors[3.11.3/5.1] (also called separation colors) #7629
- Tilings
  - Added `offset` parameter[3.11.13/3.3] for shifting the starting position of a tiling #7506
  - Fixed parent-relative placement for `stack`[3.10.28] and `polygon`[3.11.9] #8324
- Gradients
  - Added `color.map.coolwarm`[3.11.3/3.1] for use with gradients #7489
  - Fixed interpolation of gradient stops[3.11.6/7.7] in Oklab color space #7326
  - Fixed gradient angle[3.11.6/7.10] handling for negative-size shapes #7826
  - Fixed gradient strokes for lines[3.11.8] and curves[3.11.4] #7863
  - Fixed parent-relative placement for `stack`[3.10.28] and `polygon`[3.11.9] #8324
- Fixed various bugs with rectangle strokes in combination with radii[3.11.10/2.5] #7357 #8081
- Fixed a potential deadlock with font fallback in SVGs #7766
- Various improvements to SVG image handling (see the resvg 0.46 and 0.47 changelogs[B]); in particular:

[A] https://github.com/typst/codex/blob/v0.3.0/CHANGELOG.md

  - Added support for SVGs without top-level `xmlns` attribute
  - Added support for variable fonts in SVG using the `font-variation-settings` CSS property
- Various improvements to PDF image handling (see the commits between `d0b540f` and `3483462` on hayro[A]); in particular:
  - Added support for JPEG2000 (`JPXDecode`) and JBIG2 (`JBIG2Decode`) images
  - Improved parsing robustness for non-compliant files
  - Added support for blend modes

### Introspection

- Layout convergence issues now result in detailed diagnostics that help pin down the cause #7364
- Added `within` selector[3.5.26/3.5] that matches elements that are contained within any elements matching an ancestor selector #8250 #7964 #8114
- Added `at`[3.12.1/11.2.2] parameter to `counter.display` function #6781
- Improved how `counter.display`[3.12.1/11.2] auto-selects the numbering to use #7446

### Data Loading

- Added support for namespaces to `xml`[3.13.7] function #7899
- Added hint when trying to read from a path that looks like a URL #7682
- Diagnostics for binary file loading failures now include file paths #8259
- The `json`[3.13.4] function now emits a friendly error when the loaded JSON has a leading UTF-8 BOM #7488

## Export

### Bundle

- Added new, experimental *bundle* export target[3.14.2] #7964
  - With bundle export, you can emit multiple output files from a single Typst project
  - Bundles can contain any combination of HTML pages[3.14.3], PDFs[3.14.4], PNGs[3.14.5], SVGs[3.14.6], and arbitrary assets[3.14.2/5]

### HTML

- Mathematical equations are now automatically exported to MathML *(thanks to @mkorje[B])* #7436
  - MathML defines how to render an equation, but also preserves its semantics
  - If you've previously relied on show rules that use `html.frame`[3.14.3/5] to render equations to SVG, try MathML output. Switching to it will improve the accessibility of your document (though rendering will be somewhat less consistent across browsers).
- The `box`[3.10.5] and `block`[3.10.4] elements' purpose is now aligned with paged export #8181 **(Breaking change)**
  - `box`[3.10.5] is used to bring block-level content inline
  - `block`[3.10.4] ensures inline-level content becomes block-level
  - Depending on the contained content, this may be achieved by setting the CSS `display` property or by wrapping in an additional `<span>` or `<div>`.
- The rules of how paragraphs are grouped in HTML have been adjusted to fix cases where paragraphs would appear unexpectedly #8181 #7505 **(Breaking change)**
  - The list of HTML elements that can be grouped into paragraphs was tweaked (it now includes all *phrasing content* with the exception of elements that default to `display: none`)
  - This default can be controlled by wrapping an element in a `box`[3.10.5] or `block`[3.10.4] as appropriate
  - HTML elements that cannot be part of paragraphs (like a `<div>`) do not immediately force adjacent inline-level Typst content to be wrapped in a paragraph; rather, they are considered *neutral* for paragraph grouping
  - Paragraph creation is only forced by *block-level Typst elements* (as opposed to HTML elements). Built--in block-level elements like headings or images wrap the HTML elements they create in `block`[3.10.4]

[B] https://github.com/linebender/resvg/blob/v0.47.0/CHANGELOG.md
[A] https://github.com/LaurenzV/hayro/compare/d0b540fc9ab8e18b4a7a000d1404139af8e9d023...34834627c0b4afa9c83c9b64d4d978b127030c77
[B] https://github.com/mkorje

elements to force adjacent inline content into paragraphs. Package authors should do the same to ensure paragraph creation is consistent between HTML and paged export.
- The `target`[3.5.30] function can now be used without the `html` feature flag (the rest of HTML export remains feature-flagged) #8248
- DOM structure and built-in show rules
  - The root `<html>` element now receives a `lang` attribute respecting what was configured for `text.lang`[3.7.12/2.16] #7208
  - The Typst `image`[3.11.7] element now always emits `width` and `height` attributes on the generated HTML `<img>` element #8118
  - Code in `<pre>` tags will now prefer raw newlines over `<br>` elements to encode line breaks #7675
  - Fixed generated HTML for quote attributions[3.6.16/2.3] #8181
  - Fixed table cell show rules not working in HTML export #7821
- Serialization
  - HTML is now minified by default; use the `--pretty` CLI flag or the checkbox in the web app to pretty-print it #8371
- HTML elements
  - The `attrs` parameter[3.14.3/4.1.2] on `html.elem` will now *fold*, meaning that partial attributes across different set rules or `html.elem` calls are combined #8182
  - `html.script`[3.14.3/6.2.83] and `html.style`[3.14.3/6.2.92] only accept a string and not arbitrary content anymore #7784 **(Breaking change)**
- Whitespace handling
  - HTML `<br>` elements now collapse adjacent Typst spaces #8166
  - Fixed spans being emitted to protect whitespace from collapsing unnecessarily in some cases #8166

## PDF

- Typst can now target multiple (compatible) PDF standards at once, e.g. PDF/UA-1 and PDF/A-2a #8294
- PDFs are now a bit more space-optimized at the cost of being harder to inspect with a text editor; use the `--pretty` CLI flag or the checkbox in the web app to pretty-print them #8294 #8371 #8430
- Labelled headings now result in named destinations even if they are not referenced #7964
- Graphics
  - Added support for more compositing features of COLRv1 fonts LaurenzV/krilla#358
  - Fixed rendering of gradients[3.11.6] in LinearRGB, CMYK, and Luma color spaces #8149
  - Fixed excessive sampling of linear gradients #7818
- Tagging
  - Added support for more specific artifact kinds in `pdf.artifact`[3.14.4/4]; these are now internally used when appropriate #8416
  - Fixed "invalid document structure" errors with complex list markers[3.6.2/3.2] #7789
  - Fixed wrong PDF tagging order for inline content outside of paragraphs #7861
  - Fixed bounding box computations for stroked shapes in tagged PDFs #8322
- Standards compliance
  - Fixed potentially incompliant PDF files by emitting an error when PDF/UA-1 is requested but complying with PDF/UA-1 would require newer features than available in the current PDF version LaurenzV/krilla#278
  - Fixed potentially incompliant PDF 1.4 files by emitting errors when implementation limits are exceeded LaurenzV/krilla#348

## SVG

- SVGs are now minified by default; use the `--pretty` CLI flag or the checkbox in the web app to pretty-print them #8371
- Somewhat reduced the size of generated SVGs #7476 #7680 #7857
- SVG elements do not have the classes `typst-frame`, `typst-doc`, `typst-group`, `typst-shape`, and `typst-text` anymore #7680 **(Minor breaking change)**
- Graphics
  - Fixed a bug where tilings[3.11.13] could be incorrectly reused in multiple places #7837
  - Fixed rendering of gradients[3.11.6] in LinearRGB, CMYK, and Luma color spaces #8149

  - Fixed handling of conic gradient angles[3.11.6/7.3.2] #7952
  - Fixed excessive sampling of linear gradients #7818
- Fixed positioning and sizing of color bitmap glyphs #7679 #8440
- Fixed sources of non-determinism in SVG export #7680

### PNG

- Fixed handling of conic gradient angles[3.11.6/7.3.2] #7952
- Fixed positioning and sizing of color bitmap glyphs #8440
- Fixed that negatively scaled[3.10.24] text with equal `x` and `y` scale would turn invisible in PNG export #8111

## Command Line Interface

- Added new `typst eval` subcommand to evaluate a Typst code expression from the CLI; this command supersedes `typst query` #7362
- Tracebacks for diagnostics are now more compact and readable #8000
- Added `--pretty` flag for producing human-readable output; output is otherwise minified by default (applies to HTML, SVG, and PDF, but not PNG) #8371
- Fonts
  - The output of `typst fonts --variants` is now more readable and informative; in particular, it also displays the paths of font files and, for variable fonts, variation axes #7490 #8425
  - The CLI now discovers fonts lazily, saving time on operations that do not need fonts (like HTML export without frames[3.14.3/5]) #7380
  - Adobe Creative Cloud fonts are now discovered as system fonts #7716
- Dependency output
  - When writing to stdout while using `--deps-format make` (which is incompatible and thus fails), no empty Make dependency file will be emitted anymore #7246
  - The JSON dependency format now includes information about outputs in addition to inputs #7209
- Path handling fixes on Windows #8508
  - Reading from a path that contains a drive letter now results in an explicit error instead of the drive letter being silently ignored
  - Attempting to read from a path that contains a reserved filename (like `COM1`) now results in an explicit error
- Non-Unicode input paths are not supported anymore #7688 **(Breaking change)**
- The experimental `--timings` argument now requires an explicit file name instead of defaulting to `record-{n}.json` #8119 **(Breaking change)**
- Added colors to `--help`/`-h` and `typst info` output #7443 #7500
- Fixed inconsistency in environment variable handling between `typst info` and `typst compile` #8030
- Fixed a bug with the injection of a live reload script when using `typst watch` with HTML export #7770
- Fixed a bug where local timezone information was taken into account even if a fixed date is set via `--creation-timestamp` or `SOURCE_DATE_EPOCH`, leading to irreproducible results #7856

## Tooling

- Syntax highlighting
  - In math, parentheses used for grouping are now highlighted differently than ones intended for display #7894
- Autocomplete and tooltips
  - Autocompletion and tooltips are now aware of parameters of user-defined functions #7808
  - Autocomplete descriptions and tooltips for font families are now more detailed and, for variable fonts, include variation axes #8425
  - Function autocompletions in math mode now always prefer round parentheses over square brackets (which are not supported in math) #8417
  - Fixed autocompletion and tooltips not working with argument lists in math mode #8116

## Deprecations

- Certain unclear/ambiguous ways to write a raw language tag; these will now emit a warning in anticipation of an upcoming change to how they are parsed[3.7.6/3.2] #8257
- Fallback to Arabic numerals for `numbering`[3.6.12] systems that do not support the number zero (e.g. for "⓪") #7936
- Various symbols were renamed, see the deprecation section in the dedicated changelog[A] for a full listing
- Some citation styles that were renamed or superseded typst/hayagriva#404 typst/hayagriva#424 typst/hayagriva#453
  - `council-of-science-editors` is now called `cse-citation-sequence-brackets-8th-edition`
  - `council-of-science-editors-author-date` is now called `cse-name-year`
  - `modern-language-association-8` / `mla-8` is superseded by `modern-language-association` / `mla`
  - `vancouver` is now called `nlm-citation-sequence`
  - `vancouver-superscript` is now called `nlm-citation-sequence-superscript`
- Undocumented array forms of `enum`[3.6.11] and `terms`[3.6.20] items #7484

## Removals

- The `path` element, use `curve`[3.11.4] instead #7554 **(Breaking change)**
- The `pattern` type, use `tiling`[3.11.13] instead #8252 **(Breaking change)**
- The `pdf.embed` element, use `pdf.attach`[3.14.4/5] instead #8252 **(Breaking change)**
- The scoped functions `cbor.decode`, `csv.decode`, `json.decode`, `toml.decode`, `xml.decode`, `yaml.decode`, and `image.decode`; directly pass `bytes`[3.5.7] to the top-level functions instead #8252 **(Breaking change)**
- Various previously deprecated symbols, see the removals section in the dedicated changelog[B] for a full listing **(Breaking change)**

## Development

- The `typst-kit` crate was completely reworked to make it easier to create a Typst `World` implementation #7710 #8026
- Diagnostic hints can now have spans (though typically they will be *detached*, which just means there isn't a span) #7364
- Increased minimum supported Rust version to 1.92 #8236
- Moved Nix flake from typst/typst[C] to typst/typst-flake[D], where it is now maintained by the community as a best effort #7512

## Migration guide

This section walks you through changes you may need to make to your existing documents to ensure compatibility with Typst 0.15.

Typst 0.15 removes definitions from the Typst standard library that were already deprecated in previous Typst versions. If you are still relying on those, also refer to the Typst 0.14[E] and Typst 0.13[F] migration guides.

### Baselines

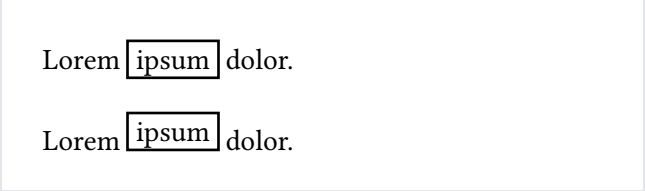
Typst now retains baseline information in more parts of the layout engine (most importantly for boxes[3.10.5], blocks[3.10.4], list items[3.6.2/4.1], and equations[3.8.16]). This may lead to silent layout shifts. These are generally improvements, but if you've previously counter-acted bad-looking output through explicit adjustments, you may need to revisit those.

[A] https://github.com/typst/codex/blob/v0.3.0/CHANGELOG.md#deprecations-in-sym
[B] https://github.com/typst/codex/blob/v0.3.0/CHANGELOG.md#removals-in-sym-breaking-change
[C] https://github.com/typst/typst
[D] https://github.com/typst/typst-flake
[E] https://typst.app/blog/2025/typst-0.14#migrating
[F] https://typst.app/blog/2025/typst-0.13#migrating

In particular, Typst will now set the baseline of a box[3.10.5] to the baseline of its first line of content and use it to align the box with its surrounding text. This yields better-looking output out of the box.

```
// Typst 0.15+
Lorem #box(inset: 0.3em, stroke: 1pt)[ipsum] dolor.

// Typst 0.14 and below
Lorem #box(inset: 0.3em, stroke: 1pt)[ipsum] dolor.
```




## File paths

If you are using Typst on Windows, you may have used backslashes in your file paths. This syntax is not supported anymore. Consistent use of forward slashes ensures your document compiles on all supported platforms.

```
// Typst 0.15+ ✅
#include "chapters/introduction.typ"

// Typst 0.14 and below ❌
#include "chapters\\introduction.typ"
```

Moreover, the Typst CLI does not support non-Unicode input paths anymore. Such paths are exceedingly rare today.

## Math glyph stretching

In mathematical typesetting, glyphs frequently need to be stretched to the size of some part of an equation. This primarily applies to brackets, but also accents, arrows, and more. In Typst, this happens implicitly through matched delimiters or explicitly through the `lr`[3.8.20/2.1.1] and `stretch`[3.8.24] functions.

Typst 0.15 makes some adjustments to how ratios passed to these functions are resolved. Previously, a size of `200%` passed to `lr.size`[3.8.20/2.1.1] or `stretch.size`[3.8.24/1.2] would be interpreted relative to a glyph size that was potentially already affected by display sizing[3.8.25/1.1]. Now, ratios are always resolved relative to the base glyph size. In practice, this means that, to retain the same appearance, you may need to increase the target sizes you've configured for glyphs that are already larger out of the box in display style (e.g. integrals).

## Math delimiters as functions

In Typst's math mode, various symbols defined in Símbolos generales[3.9.2] can be called like a function, either to be applied as an accent or to delimit body content. Typst 0.15 extends this handling to additional delimiters like `chevron.l`. This can change the output of existing calls, which would previously fall back to displaying the parentheses (which would rarely have produced desirable output in the case of delimiters).

```
// Typst 0.15+ ✅
$ chevron.l(x) $

// Typst 0.14 and below ⚠️
$ chevron.l(x) $
```



## Math classes

The `class`[3.8.14] function defines how part of an equation should be laid out (primarily the spacing around it). In Typst 0.14 and below, the `class` function applied recursively: If applied to a larger piece of content, all items in it would receive the class. In Typst 0.15, the class is only applied to the directly wrapped content. This is mostly a bug fix, but can lead to subtle layout changes.

## Calligraphic letterforms in the default math font

With the update to *New Computer Modern Math* 8.1.0, the default calligraphic letterforms[3.8.27/1.6] were changed. The previous letterforms can still be accessed by selecting stylistic set[3.7.12/2.24] 6.

```
// Typst 0.15, default style.
$ R != cal(R) $

// Typst 0.15, with stylistic set 6.
// Reproduces the default style from Typst 0.14 and below.
#show math.equation: set text(stylistic-set: 6)
$ R != cal(R) $
```

$$R \neq \mathcal{R}$$

$$R \neq \mathcal{R}$$

## Paragraphs, boxes, and blocks in HTML

Typst automatically collects inline-level content into paragraphs[3.6.14/2]. In Typst 0.14 and below, the exact same rules were used to collect inline-level HTML elements (e.g. a ) into Typst paragraphs, which in turn result in <p> elements. This had the unfortunate effect that even use of the low-level typed HTML API[3.14.3] could result in <p> elements appearing automatically. Consider the example below:

```
#html.div({
  html.span[Hello]
  html.div[World]
})
```

Where Typst 0.14 would insert an unexpected additional <p> element, Typst 0.15 yields the expected result:

```
<!-- Typst 0.14 and below ⚠️ -->
<div>
  <p><span>Hello</span></p>
  <div>World</div>
</div>

<!-- Typst 0.15+ ✅ -->
<div>
  <span>Hello</span>
  <div>World</div>
</div>
```

This change in output is achieved through a few new rules for how paragraphs are collected in HTML export. Previously, Typst would always categorize elements into either inline-level or block-level. It would then force inline-level elements into a paragraph whenever at least one block-level element was present in the same flow of content. In Typst 0.15, elements can instead be considered inline-level, block-level, or *neutral*:

- Text, Typst box[3.10.5] elements and HTML *phrasing content* (elements that are allowed as children of <p> elements) are considered inline-level[A]

- All other HTML elements are considered neutral
- Typst `block`[3.10.4] elements are considered block-level

Neutral elements don't force adjacent inline-level content into paragraphs. They can co-exist with inline-level content in a *mixed flow*. This ensures that usage of the typed HTML API does not result in extraneous paragraphs.

Meanwhile, Typst elements that are block-level out of the box (e.g. `heading`[3.6.9]) now explicitly use a `block`[3.10.4] in their default show rule to ensure that they still force adjacent inline-level elements into paragraphs. This ensures consistency in paragraph grouping across paged and HTML export. Package authors should also explicitly use `blocks`[3.10.4] when creating HTML components that Typst should consider block-level.

To make this change possible, the way `boxes`[3.10.5] and `blocks`[3.10.4] behave in HTML export has been adjusted: Instead of unconditionally wrapping their contents in a `<span>` or `<div>`, they now do just what is necessary to ensure their contents are considered inline- or block-level by browsers. If they contain multiple children, they still create a wrapper element, but for just a single child, they instead configure the CSS `display` property. And if the single child already has the appropriate `display` property by default, it is fully omitted. Then, the effect of the `box` or `block` is limited to influencing Typst's paragraph grouping.

## HTML `script` and `style` elements

The typed HTML functions for the HTML `script`[3.14.3/6.2.83] and `style`[3.14.3/6.2.92] element previously accepted arbitrary body content and then only failed during HTML serialization if the content resulted in non-textual elements. In Typst 0.15, they instead only accept strings in the first place. Note that you can also use them with `raw`[3.7.6] syntax by accessing the `.text` field.

```
// Typst 0.15+ ✅
#html.style("a { color: red }")
#html.style(
  ```css
  a { color: red }
  ```.text
)

// Typst 0.14 ❌
#html.style[
  a { color: red }
]
```

## Classes in SVG export

Typst 0.14 and below would generate the `class` attributes `typst-frame`, `typst-doc`, `typst-group`, `typst-shape`, and `typst-text` on SVG elements in SVG export. These are no longer emitted. If you are using the `typst-frame` class to style `html.frame`[3.14.3/5] elements in HTML export, you'll need to adapt your style sheet.

## Variable font family names

Typst aims to unify different fonts from the same family under a single family name. To that effect, it automatically trims common style suffixes like "Bold" or "Condensed" from font family names. Instead of selecting these through the name, they should be accessed through Typst's built-in mechanisms (such as the `weight`[3.7.12/2.4] and `stretch`[3.7.12/2.5] parameters).

In Typst 0.15, the additional suffixes "Variable", "Var", and "VF" are trimmed to unify static and variable fonts into a single family. If you've previously used a font with any of these suffixes in its family name, you should now omit the suffix when specifying the `font`[3.7.12/2.1].

[A] There are a few exceptions: Some elements like `<script>` are considered phrasing content by the HTML specifications, but don't make sense in paragraphs as they default to `display: none`.

## Upcoming raw language tag changes

In the next Typst version, we will make a change to how raw language tags are parsed[3.7.6/3.2]. Should this upcoming change have an effect on your document, the compiler will already warn you today and suggest how to adjust your document to future-proof it against the change.

## Numbering fallback for the number zero

Not all `numbering`[3.6.12] systems can express the number zero. In Typst 0.14 and below, those would silently fall back to Arabic numerals for `0`. This behavior is deprecated in Typst 0.15. Using zero with these systems will become a hard error in the future.

## Tightened validation in some functions

The `array.slice`[3.5.2/3.9] function, `str`[3.5.28] constructor, and `text.features`[3.7.12/2.32] parameter now perform stricter validation on their inputs. If you've previously passed invalid or nonsensical input, you will now receive an error.

## Renamed citation styles

Typst 0.15 renames some citation styles to stay aligned with upstream CSL changes. If you are relying on any of these, the compiler will warn you and suggest the new name.

## Renamed symbols

Typst 0.15 also renames[A] a few symbols in Símbolos generales[3.9.2]. If you are using any of these, the compiler will warn you and suggest the new name.

[A] https://github.com/typst/codex/blob/v0.3.0/CHANGELOG.md#deprecations-in-sym

# Typst 0.14.2 (December 12, 2025)

## Security

- Updated the WebAssembly runtime used for executing plugins[3.5.23]. The version used in Typst 0.14.0 and 0.14.1 suffers from a memory handling vulnerability. Based on our investigation, the vulnerability would be very hard to exploit in practice, but an exploit could theoretically be feasible. In any case, **we recommend upgrading to Typst 0.14.2.** This holds in particular for local users. In the web app, the bug is not critical as the browser offers an extra layer of protection.

  Typst 0.13.1 and below are not affected by this vulnerability.

  *Technical details:* The `wasmi`[A] WebAssembly runtime versions used in 0.14.0 and 0.14.1 have a *use-after-free* memory handling bug in certain memory growth situations. Specifically, the bug occurs when the plugin tries to grow its memory, but allocating the requested amount of memory fails. Based on our investigation, the bug is hard to trigger in practice as the WebAssembly linear memory is always limited to 4GB on a technical level and modern operating systems rarely fail to serve a 4GB memory allocation request (typically not even under RAM pressure). Once the bug is triggered, it would also still be very challenging to turn it into an actual exploit. Regardless, we recommend upgrading to Typst 0.14.2 for protection against a potential exploit.

## Diagnostics

- Added a hint when `array.sorted`[3.5.2/3.30] fails because a pair of elements could not be compared. This hint aids with fixing bugs in user code that were surfaced by a change in internal sorting behavior in Typst 0.14.1.

[A] https://github.com/wasmi-labs/wasmi/

# Typst 0.14.1 (December 03, 2025)

## PDF export

- Fixed regression where links to labelled elements would sometimes not work correctly LaurenzV/krilla#304
- Fixed bug where PDF text attributes could be written incorrectly #7478
- Fixed crash in link handling #7471
- Fixed crash for zero-sized pages #7454
- Fixed crash when a table `vline`[3.6.19/4.3] or `hline`[3.6.19/4.2] has an out-of-bounds index #7448
- Fixed crash in formatting of font-related PDF export errors #7460
- Fixed crash when a `footnote`[3.6.8] or `place`[3.10.19] element was queried and reinserted into the document #7216
- Fixed crash for PNGs with invalid metadata LaurenzV/krilla#286, LaurenzV/krilla#287
- Fixed bug where text in SVGs with `fill-and-stroke` paint order could be exported incorrectly LaurenzV/krilla#292
- Fixed bug with layer isolation in SVGs where blending/masking is used LaurenzV/krilla#295
- Fixed that table headers could be tagged incorrectly in some scenarios LaurenzV/krilla#289
- Fixed issues where generated PDFs could differ between 32-bit and 64-bit systems LaurenzV/krilla#317, LaurenzV/krilla#316, LaurenzV/krilla#312
- Upgraded JPEG decoder used during PDF export for improved compatibility, fixing a case where a valid JPEG was rejected LaurenzV/krilla#288
- A PDF document information dictionary that would be empty is now fully omitted instead LaurenzV/krilla#280
- A rare crash in PDF tagging was turned into a compiler error #7450

## HTML export

- Fixed export of table headers[3.6.19/4.4] and footers[3.6.19/4.5] with gutter #7332
- A page[3.10.17] set rule in HTML export is now a warning instead of a hard error, in line with how unsupported elements are generally treated #7513

## Math

- Fixed regression where `arrow.l.r` could not be used as an accent[3.8.9] anymore #7481
- Fixed that single-letter strings did not react to spaces around them like multi-letter strings do #7276
- Fixed that spacing around `mat`[3.8.21] and `vec`[3.8.28] with a fence delimiter was whitespace-dependent #7309
- Fixed height calculation for horizontally stretched glyphs #7327

## Model

- Fixed regression where Typst would error in heading numbering functions that don't handle the counter state (`0,`). This can occur in the first layout iteration. Such errors are usually automatically caught by Typst, which was not the case here. #7459

## Text

- Fixed regression where Typst would synthesize superscripts for some fonts even when typographic[3.7.11/2.1] glyphs were available #7462
- Fixed regression where some oblique fonts would be classified as italic #7483

## Scripting

- Fixed crash due to violated invariants in `array.sorted`[3.5.2/3.30] #7520
- Fixed crashes due to overflow in `calc.rem`[3.5.9/2.37], `calc.rem-euclid`[3.5.9/2.39], `calc.div-euclid`[3.5.9/2.38], `calc.quo`[3.5.9/2.40], and `calc.gcd`[3.5.9/2.25] #7419
- Upgraded WebAssembly runtime, fixing a bug that plugins[3.5.23] could run into #7438

## Command Line Interface

- Compiling to standard output in combination with `--deps -` `--deps-format=make` (which results in an error) will not produce an empty Make dependency file as a side effect anymore #7246

## Development

- Increased minimum supported Rust version to 1.89 #7363

# Typst 0.14.0 (October 24, 2025)

## Highlights

- Typst now produces *accessible* PDFs[4.4] out of the box, with opt-in support for stricter checks and conformance to PDF/UA-1
- Typst now supports all PDF/A standards[3.14.4/3.2.3]
- PDFs can now be used as images[3.11.7/3.2] (thanks to @LaurenzV[A])
- Added support for character-level justification[3.6.14/4.4] (can significantly improve the appearance of justified text)
- Added support for many more built-in elements in HTML export
- Added typed HTML API (e.g. `html.div`[3.14.3/6.2.29]) with individually typed attributes
- Added support for multiple headers[3.6.19/4.4] and subheaders in tables
- Added `title`[3.6.21] element for displaying the document title
- Added `frac.style`[3.8.19/3.3] property for producing skewed and inline fractions

## PDF export

PDF export was fully rewritten to use the new `krilla`[B] library, fixing various bugs and enabling many improvements. Known fixes are listed below, but there will likely be other changes in how the output behaves. If you spot any regressions, please report them on GitHub[C]. *(Thanks to @LaurenzV[D] for creating krilla!)* #5420

- Typst now produces *accessible* PDFs out of the box. Such documents are suitable for consumption in a wide range of circumstances. That not only includes consumption by people with permanent or temporary disabilities, but also by those with different devices or preferences.
  - Typst PDFs are now *tagged* by default. *Tags* are rich metadata that PDF viewers can use to make the document consumable in other ways than visually (e.g., through a screen reader).
  - In addition, Typst can now emit documents conforming to the PDF/UA-1 standard. (PDF/UA-2 is not yet supported, but planned.)
  - There is an increasing amount of existing and upcoming legislation requiring documents to be accessible, for instance, the European Accessibility Act and the Americans with Disabilities Act.
  - For more details on all of this, read the new Accessibility Guide[4.4].
- Typst now supports all PDF/A standards: PDF/A-1b, PDF/A-1a, PDF/A-2b, PDF/A-2u, PDF/A-2a, PDF/A-3b, PDF/A-3u, PDF/A-3a, PDF/A-4, PDF/A-4f, and PDF/A-4e. See the expanded PDF/A documentation[3.14.4/3.2.3] for guidance on how to select a suitable standard. #5420 #7038
- Typst now supports the PDF versions 1.4, 1.5, 1.6, and 2.0 in addition to PDF 1.7. See the relevant section of the PDF documentation[3.14.4/3.2.1] for details. #5420
- Added `pdf.artifact`[3.14.4/4] function for marking content as not semantically meaningful #6619
- Added experimental `pdf.header-cell`[3.14.4/7], `pdf.data-cell`[3.14.4/6], and `pdf.table-summary`[3.14.4/8] functions for enhancing accessibility of documents with complex tables. These functions are guarded by the `a11y-extras` feature. They do not have a final interface and will be removed in the future, either through integration into table functions or through full removal. #6619
- PDF heading bookmarks now contain the heading's numbering #6622
- Attachments[3.14.4/5]
  - Renamed `pdf.embed` to `pdf.attach`[3.14.4/5] (the old name will remain as a deprecated alias until Typst 0.15) #6705
  - The `mime`[3.14.4/5.3.4] property of `pdf.attach` is now checked for syntactical correctness **(Minor breaking change)**
  - Fixed parsing of the `data`[3.14.4/5.3.2] argument of `pdf.attach` #6435
  - Attachments now smartly determine whether they should be compressed #6256
- Text extraction (i.e. copy paste)

[A] https://github.com/LaurenzV
[B] https://github.com/LaurenzV/krilla
[C] https://github.com/typst/typst/issues
[D] https://github.com/LaurenzV

  - Now works correctly even when multiple different characters result in the same glyph #5420
  - Spaces between words at which a natural line break occurred are now correctly retained for text extraction #6866
  - Fixed mapping of hyphenation artifacts to Unicode text #6799
- Images
  - CMYK images now work properly in PDF export #5420
  - Improved export of text in SVG images with a filter #5420
  - Improved compatibility of SVG images with Quartz rendering engine (the engine used in Apple Preview) #5420
  - Improved handling of SVG images with high group nesting depth #5420
- Fixed a bug with text in patterns #5420
- Fixed gradients with transparency #5420

## HTML export

- Added support for many more built-in elements (the *Model* category[3.6] is now fully covered)
  - The `image`[3.11.7] element #6578
  - The `footnote`[3.6.8] and `footnote.entry`[3.6.8/4.1] element #6917
  - The `outline`[3.6.13] and `outline.entry`[3.6.13/5.1] element #6606
  - The `bibliography`[3.6.1] element #6952
  - The `smartquote`[3.7.8] element #6710
  - The `sub`[3.7.10] and `super`[3.7.11] elements #6422
  - The `underline`[3.7.13], `overline`[3.7.5], `strike`[3.7.9], and `highlight`[3.7.1] elements #6510
  - The `smallcaps`[3.7.7] element #6600
  - The `lower`[3.7.4] and `upper`[3.7.14] functions #6585
- Added typed HTML API (e.g. `html.div`[3.14.3/6.2.29]) with individually typed attributes #6476
  - For example, to generate a `video` element you can now write `#html.video(width: 400, src: "sunrise.mp4")` instead of `#html.elem("video", attrs: (width: "400", src: "sunrise.mp4"))`. Note how the `width` attribute takes an integer instead of a string.
- Added support for intra-doc `link`[3.6.10] targets #6602
- The `raw`[3.7.6] element
  - Added syntax highlighting support #6691
  - Block-level `raw` elements now emit both a `<code>` and a `<pre>` tag #6701
  - The language tag[3.7.6/4.3] of `raw` elements is now preserved as a `data-lang` attribute on the `<code>` tag #6702
- The `authors`[3.6.5/3.4] and `keywords`[3.6.5/3.6] properties of the `document` function now yield corresponding HTML `<meta>` tags #6134
- The `html.elem`[3.14.3/4] function now supports custom HTML element names #6676
- Improved encoding of `html.frame`[3.14.3/5] #6605
- Empty attributes are now encoded with shorthand syntax (e.g. `<div hidden></div>`) #6479
- Zero-sized horizontal weak spacing (`h(0pt, weak: true)`) does not cause a "was ignored during HTML export" warning anymore, so it can be used to destruct surrounding spaces without producing any output, as in paged export #6917
- Fixed encoding of `<pre>` and `<textarea>` elements that start with a newline #6487 #6497
- Fixed encoding of raw text elements[A] #6487 #6720
- Fixed sizing of `html.frame`[3.14.3/5] #6505
- Fixed `measure`[3.10.14] in HTML export #7186
- Fixed nested `html.frames`[3.14.3/5] #6509
- Fixed that a `box`[3.10.5] without a body was ignored in HTML export #6709
- Fixed encoding of whitespace in HTML #6750

## SVG export

- Added support for COLR-flavored color glyphs #6693

[A] https://html.spec.whatwg.org/#raw-text-elements

- Reduced amount of <g> grouping elements that are generated #6247

## PNG export

- Fixed crash when text size[3.7.12/2.6] is negative #7004

## Visualize

- Added support for using PDFs as images[3.11.7] using the new hayro[A] library. PDFs will be embedded directly in PDF export, rasterized in PNG export, and turned into SVGs in SVG and HTML export. *(Thanks to @LaurenzV[B] for creating hayro!)* #6623
- Added support for WebP images #6311
- Various minor improvements for SVG images (see the resvg 0.44 and 0.45 changelogs[C])
- SVG images can now refer to external image files #6794
- Clip paths are now properly anti-aliased #6570
- Fixed gradients on curves where the last segment is `curve.line`[3.11.4/3.2] #6647
- Fixed stroke cap handling of shapes with partial strokes #5688
- Fixed corner radius handling of shapes with partial strokes #6976
- Fixed crash when sampling across two coinciding gradient stops #6166

## Layout

- Added opt-in support for character-level justification in addition to word-level justification, configured via the new `par.justification-limits`[3.6.14/4.4] property. This is an impactful microtypographical technique that can significantly improve the appearance of justified text. #6161
- Fixed wrong linebreak opportunities related to object replacement characters #6251
- Fixed an issue where a breakable block would still produce an empty segment even if nothing fit into the first segment, leading to various undesirable behaviors in combination with fills, strokes, and stickiness[3.10.4/2.13] #6335
- Fixed crash with set rule for column or rowspan on a grid cell #6401
- Fixed CJK-Latin-spacing[3.7.12/2.11] at manual line breaks #6700 and at sub- and superscript boundaries #7175

## Math

- Added `frac.style`[3.8.19/3.3] property with new options for skewed and inline fractions #6672
- Added `math.equation.alt`[3.8.16/3.5] property for setting an alternative description for an equation #6619
- Text handling
  - A single equation can now use multiple fonts #6365
  - Glyph layout in math now uses proper text shaping, leading to better handling of more complex Unicode features #6336
  - Generated characters in an equation (e.g. the √ produced by `sqrt`) can now be targeted by text show rules #6365
  - Added `scr`[3.8.27/1.7] function for roundhand script font style #6309
  - Added `dotless` parameter to `accent`[3.8.9] (typically for rendering a dotless accented i or j) #5939
  - Script-style glyphs are now preferred at reduced math sizes #6320
  - Fixed `text.stroke`[3.7.12/2.8] in math #6243
  - Broken glyph assemblies are now prevented even when font data is incorrect #6688
- Layout
  - Fixed a bug with vertical accent positioning #5941
  - Fixed positioning of bottom accents #6187
  - Fixed a bug with layout of roots #6021
  - Improved layout of `vec`[3.8.28] and `cases`[3.8.13], making it consistent with `mat`[3.8.21] #5934
  - Removed linebreak opportunity before closing bracket in inline math #6216

[A] https://github.com/LaurenzV/hayro
[B] https://github.com/LaurenzV
[C] https://github.com/linebender/resvg/blob/v0.45.1/CHANGELOG.md#0450---2025-02-26

- An `augment`[3.8.21/2.3] line can now exist at the start and end of a matrix, not only in between columns and rows #5806
- Shorthands and multi-character numbers do not bind more tightly than fractions anymore in cases like `$x>=(y)/z$` #5925 #5996 **(Minor breaking change)**
- Named arguments passed to symbols used as function now raise an error instead of being silently ignored #6192 **(Minor breaking change)**
- The `mid`[3.8.20/2.2] element does not force the `"large"` math class[3.8.14] upon its contents anymore and instead defaults to `"relation"` #5980
- Fixed error in math parsing when `..` isn't followed by anything #7105
- Fixed the default math class of ϑ, ∫, ∱, ∳, ∫, ∀, and ‹ #5949 #6537

*Thanks to @mkorje[A] for his work on math!*

## Model

- Tables
  - Added support for multiple `headers`[3.6.19/4.4] and subheaders in tables #6168
  - Table headers now force a rowbreak, i.e. an incomplete row before a header will not be filled with cells after the header #6687
  - Fixed a bug where `headers`[3.6.19/4.4] and footers could accidentally expand to contain non-header cells #5919
- Added `title`[3.6.21] element for displaying the document title #5618
- Added `figure.alt`[3.6.7/5.2] property for setting an alternative description for a figure #6619
- `Link`[3.6.10] hit boxes for text are now vertically a bit larger to avoid issues with automatic link detection in PDF viewers #6252
- The `link`[3.6.10] function will now produce an error when passed an empty string as a URL #7049 **(Minor breaking change)**
- The value of the `number`[3.6.11/4.1.1] argument of `enum.item` now takes `auto` instead of `none` for automatic numbering #6609 **(Minor breaking change)**
- Improved spacing of nested tight lists #6242
- Fixed that `quotes: false` was ignored for inline-level quotes[3.6.16/2.1] #5991
- Fixed `hanging indent`[3.6.9/5.8] for centered, numbered headings #6839
- Fixed `footnote.entry`[3.6.8/4.1] show rules breaking links from footnote to entry #6912
- Hebrew numbering does not add Geresh and Gershayim anymore #6122

## Bibliography

- Built-in styles
  - Updated styles to their latest upstream CSL versions typst/hayagriva#350
  - Renamed `"chicago-fullnotes"` to `"chicago-notes"` (the old name remains as a deprecated alias) #6920
  - Renamed `"modern-humanities-research-association"` to `"modern-humanities-research-association-notes"` (the old name remains as a deprecated alias) #6994
  - Added support for locator/supplement in alphanumeric style typst/hayagriva#307
- Hayagriva format
  - Added `chapter` field[B] corresponding to CSL `chapter-number` and BibLaTeX `chapter` typst/hayagriva#383
- BibLaTeX format
  - Fixed parsing of alphanumeric page ranges typst/biblatex#86
  - Added support for `%` comment syntax typst/biblatex#80
  - Fixed parsing of space-separated single character commands typst/biblatex#71
  - Added "primaryclass" alias for "eprintclass" field typst/biblatex#75
  - Added support for BibLaTeX `language` field typst/hayagriva#317
  - Improved translation of BibLaTeX fields to `genre` and `serial-number` typst/hayagriva#296 typst/hayagriva#369

[A] https://github.com/mkorje
[B] https://github.com/typst/hayagriva/blob/v0.9.1/docs/file-format.md#chapter

- CSL handling
  - The bibliography rendering now uses `strong`[3.6.18], `emph`[3.6.6], and `smallcaps`[3.7.7] to express CSL font styling instead of directly adjusting the `text`[3.7.12] style, making styling easier #6984
  - Added support for date seasons, which are displayed when the month is missing typst/hayagriva#391
  - Terms for "AD" and "BC" are now correctly used from the chosen locale typst/hayagriva#364
  - Fixed how subsequent citations with differing supplements translate into CSL `ibid` and `ibid-with-locator` positions #6171
  - Fixed handling of `ibid` and `ibid-with-locator` positions in styles typst/hayagriva#301
  - Fixed the `location` conditional in CSL styles for citations with no locator typst/hayagriva#399
  - Fixed accesses of the year suffix resulting in wrong CSL renders typst/hayagriva#400
  - Fixed regression where page variables were no longer supported in styles' `<number>` elements typst/hayagriva#289
  - Fixed sorting and formatting of name parts typst/hayagriva#287 typst/hayagriva#313
  - Fixed year suffix collapsing typst/hayagriva#367
  - Fixed delimiters in locale-specific date formatting typst/hayagriva#385
  - Fixed rendering of date ordinals typst/hayagriva#366
  - Fixed rendering and sorting of dates with BC years typst/hayagriva#334 typst/hayagriva#368
  - Fixed sorting for empty sort values typst/hayagriva#390

## Text

- The `sub`[3.7.10] and `super`[3.7.11] functions now use the `subs` and `sups` OpenType font features instead of special Unicode characters for typographic scripts, fixing semantical, sizing, and positioning issues #5777
- The `raw`[3.7.6] element
  - Tweaked default syntax-highlighting color scheme of `raw`[3.7.6] text to make the colors more accessible #6754
  - JSON keys and string values now use different colors in the default `raw` syntax highlighting `theme`[3.7.6/4.6] #6873
  - Fixed a crash when a `raw` `syntax`[3.7.6/4.5] contains an unescaped trailing backslash #6883
  - Fixed tab indentation in `raw`[3.7.6] text with CRLF line terminators #6961
- Translations
  - Added term translations (a term being the "Section" in "Section 1") for many new languages by importing pre-existing translations from LaTeX packages #6852
  - Added term translations for Indonesian #6108, Latvian #6348, Croatian #6413, Lithuanian #6587, French #7010, French (Canada) #7098, Galician #7019, Dutch #7026, Danish #7031, Slovenian #7032, Italian #7050, Chinese #7044, Spanish #7051, and Irish (Gaeilge) #7024
  - Improved term translations for Czech #6101, Swedish #6519, Galician #7019, Dutch #7026, Chinese #7044, Spanish #7051, Irish (Gaeilge) #7024, and German #7082
  - Improved smart quotes for French #5976, Ukrainian #6372, Russian #6331, and Arabic #6626
- An empty font list is not allowed anymore in `text.font`[3.7.12/2.1] #6049 **(Minor breaking change)**
- Added a warning when using a variable font as those are not currently supported #6425
- Fixed usage of the same font with different `coverage`[3.7.12/2.1] settings #6604
- Fixed hyphens not showing up when hyphenating at specific positions (where invisible metadata exists) #6807
- Fixed styling of repeated hyphens in languages with hyphen repetition #6798
- Last resort font fallback does not consider default ignorable characters anymore during font selection #6805
- Updated New Computer Modern fonts to version 7.0.4 #6376
- Updated data and shaper to Unicode 16.0.0 #5407

## Scripting

- The `in` operator can now be used to check whether a definition is present in a module #6498
- Added `default` parameter to `array.first`[3.5.2/3.2], `array.last`[3.5.2/3.3] #5970, `array.join`[3.5.2/3.26] #6932, `str.first`[3.5.28/4.2], and `str.last`[3.5.28/4.3] #6554 methods
- Added `by` parameter to `array.sorted`[3.5.2/3.30] for sorting with a comparison function #5627

- Added `str.normalize`[3.5.28/4.10] function for Unicode normalization #5631
- Added `from`[3.10.8/2.1], `to`[3.10.8/2.2], and `sign`[3.10.8/2.4] methods to `direction` type #5893
- Labels[3.5.18] cannot be empty anymore #6332 **(Minor breaking change)**
- The WebAssembly runtime used by the `plugin`[3.5.23] system was updated and now supports SIMD #6997
- Numberings and counters now use 64-bit numbers instead of platform-dependent numbers for consistency across platforms #6026
- Improved consistency of how large numbers are handled in data loading functions #6836
- The `toml`[3.13.6] function is now guaranteed to return a dictionary[3.5.12] and `toml.encode`[3.13.6/4.1] must receive a dictionary (it already errored before when passed something else, but the new function signature makes the error clearer) #6743
- Serialization of `bytes`[3.5.7] in human-readable formats now uses `repr`[3.5.25] #6743
- Fixed slicing of last N elements of an array using the `count`[3.5.2/3.9.3] parameter #6838
- Fixed crash when the expression wrapped in a `context` expression is an anonymous function #6975
- Fixed equality check between `raw.line`[3.7.6/5.1] elements #6560
- Fixed `repr`[3.5.25] of labels[3.5.18] being potentially syntactically invalid #6456
- Fixed `repr`[3.5.25] of functions modified through `with`[3.5.16/8.1] #6773

## Introspection

- The following elements are newly locatable (i.e. they can be discovered with `query`[3.12.6], `locate`[3.12.3], etc. without having a label): `par`, `table`, `enum`, `list`, `terms`, `title`, `figure.caption`, `footnote.entry`, `outline.entry`, `image`, `emph`, `strong`, `link`, `cite`, `raw`, `underline`, `overline`, `strike`, and `highlight` #6619
- Fixed issues with logical order in bidirectional text #5887 #6796
- Fixed logical order of cells in right-to-left `grid`[3.10.10] #6232
- Fixed logical order of elements in `grid`[3.10.10] cells that span multiple pages #7198
- Fixed logical order of metadata at the starts and ends of paragraphs #6881 #6909
- Fixed introspection positions[3.12.4/2.3] of inline elements at the very start of a paragraph (previously the Y position of an element at the very start would differ from one in the middle of the first line) #6881

## Styling

- Fixed rare infinite loop in show rule application #6259

## Performance

- Optimized incremental compilation with a new algorithm that, in particular, eliminates cases of very slow compilation with heavy context usage #6683

## Command Line Interface

- Added `typst info` subcommand for emitting build and environment information #6761
- Added `typst completions` subcommand for retrieving shell completions #6568
- Added `TYPST_IGNORE_SYSTEM_FONTS` environment variable #6092
- Added `--ignore-embedded-fonts` flag and `TYPST_IGNORE_EMBEDDED_FONTS` environment variable for disabling the use of fonts embedded into the Typst binary #7037
- Added `--no-pdf-tags` flag for disabling the automatic generation of accessibility tags. These are emitted when no particular standard like PDF/UA-1 is selected to provide a baseline of accessibility. #6619
- Added `--target` parameter to `typst query` #6405
- Added `--deps` and `--deps-format` parameters for emitting a list of files compilation depends on. Supports the three formats `json`, `zero`, and `make`. #7022
- Deprecated the `--make-deps` flag in favor of `--deps` with `--deps-format make` #7022
- On Linux, the font search will now fall back to known font directories if none were loaded via Fontconfig. RazrFalcon/fontdb#71
- The CLI will now warn when trying to watch the standard input #6381
- Fixed race condition when two separate Typst CLI processes concurrently try to install a package #5979

- Fixed incremental SVG export not writing output SVGs on changes that only affect the page (e.g. changing `page.fill`) #6810
- Fixed a rare potential crash when stack space couldn't be grown as expected #6969

## Tooling and Diagnostics

- Errors in many kinds of external text files (e.g., bibliographies, JSON files, etc.) are now annotated within these files instead of at the positions where the files are loaded from a Typst file #6308
- Warnings originating from within `eval`[3.5.14] are now correctly emitted #6100
- Diagnostic messages and hints
  - Improved error messages related to parsing of numbers #5969
  - The error message for an unsuitable CSL style now mentions the name of the style #6306
  - Added hints to various deprecated items with the removal timeline #6617
  - Added hint for the error message when an X/Y pair is expected #6476
  - Added hint for a label that appears in both the document and the bibliography #6457
  - Added additional hint for show rule recursion depth error #5856
  - Fixed inconsistent formatting of code points and strings in error messages #6487
- Autocompletion
  - Labels will now be deduplicated in completions #6516
  - Math font autocompletions are now based on the presence of an OpenType MATH table instead of the word "Math" in the name #6316
  - Autocompletion immediately after a comma in a parameter list is now supported for explicitly triggered completions (e.g. via Ctrl/Cmd+Space) #6550
  - Citation style aliases are now displayed as autocompletions #6696
  - Fixed autocompletion false positives with cursor after parameter list #6475
  - Fixed autocompletion after partial identifier in math #6415
  - Fixed which definitions are suggested in math #6415
  - Fixed inapplicable method autocompletions being listed #5824
- Tooltips
  - Fixed tooltip for scoped functions (e.g. `calc.round`) #6234
  - Fixed tooltip and details for figure references #6580
  - Expression tooltips now use × instead of `x` to indicate a repeated value #6163
- Fixed jump from click (jumping to the source panel with a click in the preview) in presence of transformations and clipping #6037

## Symbols

- Added many new symbols and variants; many more than could be listed here. View the dedicated changelog[A] for a full listing.
- Code points that have a symbol and emoji presentation now have the correct variation selector attached depending on whether they appear in `sym` or `emoji`. That said, they still don't render consistently in Typst due to how font fallback works. typst/codex#114
- The `symbol`[3.5.29] function can now be used to create symbols that comprise not just one character, but one full grapheme cluster #6489

## Deprecations

- The name `pdf.embed` in favor of the new name `pdf.attach`[3.14.4/5] #6705
- The `"chicago-fullnotes"` bibliography style in favor of `"chicago-notes"` #6920
- The `"modern-humanities-research-association"` bibliography style in favor of `"modern-humanities-research-association-notes"` #6994
- The `--make-deps` CLI flag in favor of `--deps` with `--deps-format make` #7022
- Various symbols, see the deprecation section in the dedicated changelog[B] for a full listing

[A] https://github.com/typst/codex/blob/v0.2.0/CHANGELOG.md#version-020-october-7-2025
[B] https://github.com/typst/codex/blob/v0.2.0/CHANGELOG.md#deprecated

## Removals

- Removed compatibility behavior of type/str comparisons (e.g. `int == "integer"`), which was temporarily introduced in Typst 0.8 and deprecated in Typst 0.13 **(Breaking change)**

## Development

- The `Default` impl for `Library` had to be removed for crate splitting and trait coherence reasons, but you can get a drop-in replacement via `use typst::LibraryExt` #6576
- The `PdfOptions` struct has a new `tagged` field, which defaults to `true` #6619 #7046
- Fixed a potential panic in `World::font` implementations. Downstream `World` implementations might need to apply the same fix[A]. #6117
- Increased minimum supported Rust version to 1.88 #6637
- The Docker container now has the optional non-root user `typst` #7058

[A] https://github.com/typst/typst/pull/6117

# Typst 0.13.1 (March 07, 2025)

## Command Line Interface

- Fixed high CPU usage for `typst watch` on Linux. Depending on the project size, CPU usage would spike for varying amounts of time. This bug appeared with 0.13.0 due to a behavioral change in the inotify file watching backend.

## HTML export

- Fixed export of tables with gutters[3.6.19/3.3]
- Fixed usage of `<html>` and `<body>` element within context[3.4]
- Fixed querying of metadata[3.12.5] next to `<html>` and `<body>` element

## Visualization

- Fixed curves[3.11.4] with multiple non-closed components

## Introspection

- Fixed a regression where labelled symbols[3.5.29] could not be queried[3.12.6] by label

## Deprecations

- Fixed false positives in deprecation warnings for type/str comparisons

# Typst 0.13.0 (February 19, 2025)

## Highlights

- There is now a distinction between proper paragraphs[3.6.14] and just inline-level content. This is important for future work on accessibility and means that first line indent[3.6.14/4.6] can now be enabled for all paragraphs instead of just consecutive ones.
- The `outline`[3.6.13] has a better out-of-the-box look and is more customizable
- The new `curve`[3.11.4] function (that supersedes the `path` function) provides a simpler and more flexible interface for creating Bézier curves
- The `image` function now supports raw pixel raster formats[3.11.7/3.2] for generating images from within Typst
- Functions that accept file paths now also accept raw bytes[3.5.7], for full flexibility
- WebAssembly plugins[3.5.23] are more flexible and automatically run multi-threaded
- Fixed a long-standing bug where single-letter strings in math (`$"a"$`) would be displayed in italics
- You can now specify which charset should be covered[3.7.12/2.1] by which font family
- The `pdf.embed`[3.14.4/5] function lets you embed arbitrary files in the exported PDF
- HTML export is currently under active development. The feature is still *very* incomplete, but already available for experimentation behind a feature flag.

## Model

- There is now a distinction between proper paragraphs[3.6.14] and just inline-level content **(Breaking change)**
  - All text at the root of a document is wrapped in paragraphs. Meanwhile, text in a container (like a block) is only wrapped in a paragraph if the container holds any block-level content. If all of the content is inline-level, no paragraph is created.
  - In the laid-out document, it's not immediately visible whether text became part of a paragraph. However, it is still important for accessibility, HTML export, and for properties like `first-line-indent`.
  - Show rules on `par` now only affect proper paragraphs
  - The `first-line-indent` and `hanging-indent` properties also only affect proper paragraphs
  - Creating a `par[..]` with body content that is not fully inline-level will result in a warning
  - The default show rules of various built-in elements like lists, quotes, etc. were adjusted to ensure they produce/don't produce paragraphs as appropriate
  - Removed support for booleans and content in `outline.indent`[3.6.13/4.4]
- The `outline`[3.6.13] function was fully reworked to improve its out-of-the-box behavior **(Breaking change)**
  - Outline entries[3.6.13/5.1] are now blocks[3.10.4] and are thus affected by block spacing
  - The `auto` indentation mode now aligns numberings and titles outline-wide for a grid-like look
  - Automatic indentation now also indents entries without a numbering
  - Titles wrapping over multiple lines now have hanging indent
  - The page number won't appear alone on its own line anymore
  - The link now spans the full entry instead of just the title and page number
  - The default spacing between outline leader dots was increased
  - The `fill`[3.6.13/5.1.3] parameter was moved from `outline` to `outline.entry` and can thus be configured through show-set rules
  - Removed `body` and `page` fields from outline entry
  - Added `indented`, `prefix`, `inner`, `body`, and `page` methods on outline entries to simplify writing of show rules
- Added configuration to `par.first-line-indent`[3.6.14/4.6] for indenting all paragraphs instead of just consecutive ones
- Added `form`[3.6.17/4.3] parameter to `ref` function. Setting the form to `"page"` will produce a page reference instead of a textual one.
- Added `document.description`[3.6.5/3.5] field, which results in corresponding PDF and HTML metadata
- Added `enum.reversed`[3.6.11/3.5] parameter
- Added support for Greek numbering[3.6.12]
- When the `link`[3.6.10] function wraps around a container like a block[3.10.4], it will now generate only one link for the whole block instead of individual links for all the visible leaf elements. This significantly reduces PDF file sizes when combining `link` and `repeat`[3.10.22].

- The `link`[3.6.10] function will now only strip one prefix (like `mailto:` or `tel:`) instead of multiple
- The link function now suppresses hyphenation via a built-in show-set rule rather than through its default show rule
- Displaying the page counter without a specified numbering will now take the page numbering into account

## Visualization

- Added new `curve`[3.11.4] function that supersedes the `path` function and provides a simpler and more flexible interface. The `path` function is now deprecated.
- The `image` function now supports raw pixel raster formats[3.11.7/3.2]. This can be used to generate images from within Typst without the need for encoding in an image exchange format.
- Added `image.scaling`[3.11.7/3.8] parameter for configuring how an image is scaled by PNG export and PDF viewers (smooth or pixelated)
- Added `image.icc`[3.11.7/3.9] parameter for providing or overriding the ICC profile of an image
- Renamed `pattern` to `tiling`[3.11.13]. The name `pattern` remains as a deprecated alias.
- Added `gradient.center`[3.11.6/7.11], `gradient.radius`[3.11.6/7.12], `gradient.focal-center`[3.11.6/7.13], and `gradient.focal-radius`[3.11.6/7.14] methods
- Fixed interaction of clipping and outset on `box`[3.10.5] and `block`[3.10.4]
- Fixed panic with `path` of infinite length
- Fixed non-solid (e.g. tiling) text fills in clipped blocks
- Fixed a crash for images with a DPI value of zero
- Fixed floating-point error in `gradient.repeat`[3.11.6/7.5]
- Auto-detection of image formats from a raw buffer now has support for SVGs

## Scripting

- Functions that accept file paths now also accept raw bytes[3.5.7]
  - `image`[3.11.7], `cbor`[3.13.2], `csv`[3.13.3], `json`[3.13.4], `toml`[3.13.6], `xml`[3.13.7], and `yaml`[3.13.8] now support a path string or bytes and their `.decode` variants are deprecated
  - `plugin`[3.5.23], `bibliography`[3.6.1], `bibliography.style`[3.6.1/4.4], `cite.style`[3.6.3/3.4], `raw.theme`[3.7.6/4.6], and `raw.syntaxes`[3.7.6/4.5] now accept bytes in addition to path strings. These did not have `.decode` variants, so this adds new flexibility.
  - The `path` argument/field of `image`[3.11.7] and `bibliography`[3.6.1] was renamed to `source` and `sources`, respectively **(Minor breaking change)**
- Improved WebAssembly plugins[3.5.23]
  - The `plugin` type is replaced by a `plugin` function[3.5.23] that returns a module[3.5.19] containing normal Typst functions. This module can be used with import syntax. **(Breaking change)**
  - Plugins now automatically run in multiple threads without any changes by plugin authors
  - A new `plugin.transition`[3.5.23/8.1] API is introduced which allows plugins to run impure initialization in a way that doesn't break Typst's purity guarantees
- The variable name bound by a bare import (no renaming, no import list) is now determined statically and dynamic imports without `as` renaming (e.g. `import "ot" + "her.typ"`) are a hard error **(Breaking change)**
- Values of the `arguments`[3.5.1] type can now be added with + and joined[3.3.2] in curly-braced code blocks
- Functions in an element function's scope can now be called with method syntax, bringing elements and types closer (in anticipation of a future full unification of the two). Currently, this is only useful for `outline.entry`[3.6.13/5.1] as no other element function defines methods.
- Added `calc.norm`[3.5.9/2.41] function
- Added support for 32-bit floats in `float.from-bytes`[3.5.15/4.4] and `float.to-bytes`[3.5.15/4.5]
- The `decimal`[3.5.11] constructor now also accepts decimal values
- Improved `repr` of symbols[3.5.29], arguments[3.5.1], and types[3.5.31]
- Duplicate symbol[3.5.29] variants and modifiers are now a hard error **(Breaking change)**

## Math

- Fixed a bug where single letter strings in math (`$"a"$`) would be displayed in italics

- Math function calls can now have hyphenated named arguments and support argument spreading[3.5.1/2]
- Better looking accents thanks to support for the `flac` (Flattened Accent Forms) and `dtls` (Dotless Forms) OpenType features
- Added `lcm` text operator[3.8.26]
- The `bold`[3.8.18/1.3] function now works with ſ and ẜ
- The `italic`[3.8.18/1.2] function now works with ħ
- Fixed a bug where the extent of a math equation was wrongly affected by internal metadata
- Fixed interaction of `lr`[3.8.20/2.1] and context[3.4] expressions
- Fixed weak spacing being unconditionally ignored in `lr`[3.8.20/2.1]
- Fixed sub/superscripts sometimes being in the wrong position with `lr`[3.8.20/2.1]
- Fixed multi-line annotations (e.g. overbrace) changing the math baseline
- Fixed merging of attachments when the base is a nested equation
- Fixed resolving of contextual (em-based) text sizes within math
- Fixed spacing around up tacks (⊥)

## Bibliography

- Prose and author-only citations now use editor names if the author names are unavailable
- Some non-standard but widely used BibLaTeX `editortypes` like `producer`, `writer`, `scriptwriter`, and `none` (defined by widespread style `biblatex-chicago` to mean performers within `music` and `video` entries) are now recognized
- CSL styles can now render affixes around the bibliography
- For BibTeX entries with `eprinttype = {pubmed}`, the PubMed ID will now be correctly processed
- Whitespace handling for strings delimiting initialized names has been improved
- Uppercase spelling after apostrophes used as quotation marks is now possible
- Fixed bugs around the handling of CSL delimiting characters
- Fixed a problem with parsing multibyte characters in page ranges that could prevent Hayagriva from parsing some BibTeX page ranges
- Updated CSL APA style
- Updated CSL locales for Finnish, Swiss German, Austrian German, German, and Arabic

## Text

- Added support for specifying which charset should be covered[3.7.12/2.1] by which font family
- Added `all`[3.7.7/4.1] parameter to `smallcaps` function that also enables small capitals on uppercase letters
- Added basic i18n for Basque and Bulgarian
- Justification[3.6.14/4.3] does not affect raw[3.7.6] blocks anymore
- CJK-Latin-spacing[3.7.12/2.11] does not affect raw[3.7.6] text anymore
- Fixed wrong language codes being used for Greek and Ukrainian
- Fixed default quotes for Croatian and Bulgarian
- Fixed crash in RTL text handling
- Added support for `raw`[3.7.6] syntax highlighting for a few new languages: CFML, NSIS, and WGSL
- New font metadata exception for New Computer Modern Sans Math
- Updated bundled New Computer Modern fonts to version 7.0.1

## Layout

- Fixed various bugs with footnotes
  - Fixed footnotes getting lost when multiple footnotes were nested within another footnote
  - Fixed endless loops with empty and overlarge footnotes
  - Fixed crash with overlarge footnotes within a floating placement
- Fixed sizing of quadratic shapes (`square`[3.11.11] and `circle`[3.11.2])
- Fixed `block.sticky`[3.10.4/2.13] not working properly at the top of a container
- Fixed crash due to consecutive weak spacing
- Fixed crash when a block[3.10.4] or text have negative sizes

- Fixed unnecessary hyphenations occurring in rare scenarios due to a bad interaction between padding and paragraph optimization
- Fixed lone citations[3.6.3] in `align`[3.10.1] not becoming their own paragraph

## Syntax

- Top-level closing square brackets that do not have a matching opening square bracket are now a hard error **(Minor breaking change)**
- Adding a space between the identifier and the parentheses in a set rule is not allowed anymore **(Minor breaking change)**
- Numbers with a unit cannot have a base prefix anymore, e.g. `0b100000pt` is not allowed anymore. Previously, it was syntactically allowed but always resolved to a value of zero. **(Minor breaking change)**
- Using `is` as an identifier will now warn as it might become a keyword in the future
- Fixed minor whitespace handling bugs
  - in math mode argument lists
  - at the end of headings
  - between a term list's term and description
- Fixed parsing of empty single line raw blocks with 3+ backticks and a language tag
- Fixed minor bug with parentheses parsing in math
- Markup that can only appear at the start of the line (headings, lists) can now also appear at the start of a list item
- A shebang `#!` at the very start of a file is now ignored

## PDF export

- Added `pdf.embed`[3.14.4/5] function for embedding arbitrary files in the exported PDF
- Added support for PDF/A-3b export
- The PDF timestamp will now contain the timezone by default

## HTML export

**Note:** HTML export is currently under active development. The feature is still *very* incomplete, but already available for experimentation behind a feature flag.

- Added HTML output support for some (but not all) of the built-in elements
- Added `html.elem`[3.14.3/4] function for outputting an arbitrary HTML element
- Added `html.frame`[3.14.3/5] function for integrating content that requires layout into HTML (by embedding an SVG)
- Added `target`[3.5.3⁰] function which returns either `"paged"` or `"html"` depending on the export target

## Tooling and Diagnostics

- Autocompletion improvements
  - Added autocompletion for file paths
  - Smarter autocompletion of variables: Completing `rect(fill: |)` will now only show variables which contain a valid fill (either directly or nested, e.g. a dictionary containing a valid fill)
  - Different functions will now autocomplete with different brackets (round vs square) depending on which kind is more useful
  - Positional parameters which are already provided aren't autocompleted again anymore
  - Fixed variable autocompletion not considering parameters
  - Added autocompletion snippets for common figure usages
  - Fixed autocompletion after half-completed import item
  - Fixed autocompletion for `cite` function
- Added warning when an unconditional return in a code block discards joined content
- Fixed error message when accessing non-existent label
- Fixed handling of nested imports in IDE functionality

## Command Line Interface

- Added `--features` argument and `TYPST_FEATURES` environment variable for opting into experimental features. The only feature so far is `html`.
- Added a live reloading HTTP server to `typst watch` when targeting HTML
- Fixed self-update not being aware about certain target architectures
- Fixed crash when piping `typst fonts` output to another command
- Fixed handling of relative paths in `--make-deps` output
- Fixed handling of multipage SVG and PNG export in `--make-deps` output
- Colons in filenames are now correctly escaped in `--make-deps` output

## Symbols

- New
  - `inter`, `inter.and`, `inter.big`, `inter.dot`, `inter.double`, `inter.sq`, `inter.sq.big`, `inter.sq.double`, `integral.inter`
  - `asymp`, `asymp.not`
  - `mapsto`, `mapsto.long`
  - `divides.not.rev`, `divides.struck`
  - `interleave`, `interleave.big`, `interleave.struck`
  - `eq.triple.not`, `eq.dots`, `eq.dots.down`, `eq.dots.up`
  - `smt`, `smt.eq`, `lat`, `lat.eq`
  - `colon.tri`, `colon.tri.op`
  - `dagger.triple`, `dagger.l`, `dagger.r`, `dagger.inv`
  - `hourglass.stroked`, `hourglass.filled`
  - `die.six`, `die.five`, `die.four`, `die.three`, `die.two`, `die.one`
  - `errorbar.square.stroked`, `errorbar.square.filled`, `errorbar.diamond.stroked`, `errorbar.diamond.filled`, `errorbar.circle.stroked`, `errorbar.circle.filled`
  - `numero`
- Renamed **(Breaking change)**
  - `ohm.inv` to `Omega.inv`
- Changed codepoint **(Breaking change)**
  - `angle.l.double` from 《 to ⟪
  - `angle.r.double` from 》 to ⟫
  - `angstrom` from U+212B (Å) to U+00C5 (Å)
- Deprecated
  - `sect` and all its variants in favor of `inter`
  - `integral.sect` in favor of `integral.inter`
- Removed **(Breaking change)**
  - `degree.c` in favor of `°C` (`$upright(°C)$` or `$upright(degree C)$` in math)
  - `degree.f` in favor of `°F` (`$upright(°F)$` or `$upright(degree F)$` in math)
  - `kelvin` in favor of just K (`$upright(K)$` in math)
  - `ohm` in favor of `Omega`

## Deprecations

- The `path` function in favor of the `curve`[3.11.4] function
- The name `pattern` for tiling patterns in favor of the new name `tiling`[3.11.13]
- `image.decode`, `cbor.decode`, `csv.decode`, `json.decode`, `toml.decode`, `xml.decode`, `yaml.decode` in favor of the top-level functions directly accepting both paths and bytes
- The `sect` and its variants in favor of `inter`, and `integral.sect` in favor of `integral.inter`
- The compatibility behavior of type/str comparisons (e.g. `int == "integer"`) which was temporarily introduced in Typst 0.8 now emits warnings. It will be removed in Typst 0.14.

## Removals

- Removed `style` function and `styles` argument of `measure`[3.10.14], use a `context`[3.4] expression instead **(Breaking change)**
- Removed `state.display` function, use `state.get`[3.12.7/6.1] instead **(Breaking change)**
- Removed `location` argument of `state.at`[3.12.7/6.2], `counter.at`[3.12.1/11.3], and `query`[3.12.6] **(Breaking change)**
- Removed compatibility behavior where `counter.display`[3.12.1/11.2] worked without `context`[3.4] **(Breaking change)**
- Removed compatibility behavior of `locate`[3.12.3] **(Breaking change)**

## Development

- The `typst::compile` function is now generic and can return either a `PagedDocument` or an `HtmlDocument`
- `typst-timing` now supports WebAssembly targets via `web-sys` when the `wasm` feature is enabled
- Increased minimum supported Rust version to 1.80
- Fixed linux/arm64 Docker image

# Typst 0.12.0 (October 18, 2024)

## Highlights

- Added support for multi-column floating placement[3.10.19/4.2] and figures[3.6.7/5.4]
- Added support for automatic line numbering[3.6.14/5.1] (often used in academic papers)
- Typst's layout engine is now multithreaded. Typical speedups are 2-3x for larger documents. The multithreading operates on page break boundaries, so explicit page breaks are necessary for it to kick in.
- Paragraph justification was optimized with a new two-pass algorithm. Speedups are larger for shorter paragraphs and go up to 6x.
- Highly reduced PDF file sizes due to better font subsetting (thanks to @LaurenzV[A])
- Emoji are now exported properly in PDF
- Added initial support for PDF/A. For now, only the PDF/A-2b profile is supported, but more is planned for the future.
- Added various options for configuring the CLI's environment (fonts, package paths, etc.)
- Text show rules now match across multiple text elements
- Block-level equations can now optionally break over multiple pages
- Fixed a bug where some fonts would not print correctly on professional printers
- Fixed a long-standing bug which could cause headings to be orphaned at the bottom of the page

## Layout

- Added support for multi-column floating placement and figures via `place.scope`[3.10.19/4.2] and `figure.scope`[3.6.7/5.4]. Two-column documents should now prefer `set page(columns: 2)` over `show: column.with(2)` (see the Page Setup Guide[4.2.4]).
- Added support for automatic line numbering[3.6.14/5.1] (often used in academic papers)
- Added `par.spacing`[3.6.14/4.2] property for configuring paragraph spacing. This should now be used instead of `show par: set block(spacing: ..)` **(Breaking change)**
- Block-level elements like lists, grids, and stacks now show themselves as blocks and are thus affected by all block properties (e.g. `stroke`) rather than just `spacing` **(Breaking change)**
- Added `block.sticky`[3.10.4/2.13] property which prevents a page break after a block
- Added `place.flush`[3.10.19/5.1] function which forces all floating figures to be placed before any further content
- Added `skew`[3.10.25] function
- Added `auto` option for `page.header`[3.10.17/4.13] and `page.footer`[3.10.17/4.15] which results in an automatic header/footer based on the numbering (which was previously inaccessible after a change)
- Added `gap` and `justify` parameters to `repeat`[3.10.22] function
- Added `width` and `height` parameters to the `measure`[3.10.14] function to define the space in which the content should be measured. Especially useful in combination with `layout`[3.10.12].
- The height of a `block`, `image`, `rect`, `square`, `ellipse`, or `circle` can now be specified in fractional units[3.10.9]
- The `scale`[3.10.24] function now supports absolute lengths for `x`, `y`, `factor`. This way an element of unknown size can be scaled to a fixed size.
- The values of `block.above` and `block.below` can now be retrieved in context expressions.
- Increased accuracy of conversions between absolute units (pt, mm, cm, in)
- Fixed a bug which could cause headings to be orphaned at the bottom of the page
- Fixed footnotes within breakable blocks appearing on the page where the breakable block ends instead of at the page where the footnote marker is
- Fixed numbering of nested footnotes and footnotes in floats
- Fixed empty pages appearing when a context[3.4] expression wraps whole pages
- Fixed `set block(spacing: x)` behaving differently from `set block(above: x, below: x)`
- Fixed behavior of `rotate`[3.10.23] and `scale`[3.10.24] with `reflow: true`
- Fixed interaction of `align(horizon)` and `v(1fr)`
- Fixed various bugs where floating placement would yield overlapping results
- Fixed a bug where widow/orphan prevention would unnecessarily move text into the next column

[A] https://github.com/LaurenzV

- Fixed weak spacing[3.10.26/4.2] not being trimmed at the start and end of lines in a paragraph (only at the start and end of paragraphs)
- Fixed interaction of weak page break and `pagebreak.to`[3.10.18/2.2]
- Fixed compilation output of a single weak page break
- Fixed crash when padding[3.10.16] by `100%`

## Text

- Tuned hyphenation: It is less eager by default and hyphenations close to the edges of words are now discouraged more strongly **(May lead to larger layout reflows)**
- New default font: Libertinus Serif. This is the maintained successor to the old default font Linux Libertine. **(May lead to smaller reflows)**
- Setting the font to an unavailable family will now result in a warning
- Implemented a new smart quote algorithm, fixing various bugs where smart quotes weren't all that smart
- Added `text.costs`[3.7.12/2.21] parameter for tweaking various parameters that affect the choices of the layout engine during text layout
- Added `typm` highlighting mode for math in raw blocks[3.7.6/4.3]
- Added basic i18n for Galician, Catalan, Latin, Icelandic, Hebrew
- Implemented hyphenation duplication for Czech, Croatian, Lower Sorbian, Polish, Portuguese, Slovak, and Spanish.
- The `smallcaps`[3.7.7] function is now an element function and can thereby be used in show(-set) rules.
- The `raw.theme`[3.7.6/4.6] parameter can now be set to `none` to disable highlighting even in the presence of a language tag, and to `auto` to reset to the default
- Multiple stylistic sets[3.7.12/2.24] can now be enabled at once
- Fixed the Chinese translation for "Equation"
- Fixed that hyphenation could occur outside of words
- Fixed incorrect layout of bidirectional text in edge cases
- Fixed layout of paragraphs with explicit trailing whitespace
- Fixed bugs related to empty paragraphs created via `#""`
- Fixed accidental trailing spaces for line breaks immediately preceding an inline equation
- Fixed `text.historical-ligatures`[3.7.12/2.27] not working correctly
- Fixed accidental repetition of Thai characters around line breaks in some circumstances
- Fixed smart quotes[3.7.8] for Swiss French
- New font metadata exceptions for Archivo, Kaiti SC, and Kaiti TC
- Updated bundled New Computer Modern fonts to version 6.0

## Math

- Block-level equations can now break over multiple pages if enabled via `show math.equation: set block(breakable: true)`.
- Matrix and vector sizing is now more consistent across different cell contents
- Added `stretch`[3.8.24] function for manually or automatically stretching characters like arrows or parentheses horizontally or vertically
- Improved layout of attachments on parenthesized as well as under- or overlined expressions
- Improved layout of nested attachments resulting from code like `#let a0 = $a_0$; $a0^1$`
- Improved layout of primes close to superscripts
- Improved layout of fractions
- Typst now makes use of math-specific height-dependent kerning information in some fonts for better attachment layout
- The `floor` and `ceil` functions in math are now callable symbols, such that `$ floor(x) = lr(floor.l x floor.r) $`
- The `mat.delim`[3.8.21/2.1], `vec.delim`[3.8.28/2.1], and `cases.delim`[3.8.13/2.1] parameters now allow any character that is considered a delimiter or "fence" (e.g. |) by Unicode. The `delim: "||"` notation is *not* supported anymore and should be replaced by `delim: bar.double` **(Minor breaking change)**
- Added `vec.align`[3.8.28/2.2] and `mat.align`[3.8.21/2.2] parameters
- Added `underparen`[3.8.15/1.7], `overparen`[3.8.15/1.8], `undershell`[3.8.15/1.9], and `overshell`[3.8.15/1.10]

- Added ~ shorthand for `tilde.op` in math mode **(Minor breaking change)**
- Fixed baseline alignment of equation numbers
- Fixed positioning of corner brackets (⌜, ⌝, ⌞, ⌟)
- Fixed baseline of large roots
- Fixed multiple minor layout bugs with attachments
- Fixed that alignment points could affect line height in math
- Fixed that spaces could show up between text and invisible elements like `metadata`[3.12.5] in math
- Fixed a crash with recursive show rules in math
- Fixed `lr.size`[3.8.20/2.1.1] not affecting characters enclosed in `mid`[3.8.20/2.2] in some cases
- Fixed resolving of em units in sub- and superscripts
- Fixed bounding box of inline equations when a text edge[3.7.12/2.14] is set to `"bounds"`

## Introspection

- Implemented a new system by which Typst tracks where elements end up on the pages. This may lead to subtly different behavior in introspections. **(Breaking change)**
- Fixed various bugs with wrong counter behavior in complex layout situations, through a new, more principled implementation
- Counter updates can now be before the first, in between, and after the last page when isolated by weak page breaks. This allows, for instance, updating a counter before the first page header and background.
- Fixed logical ordering of introspections within footnotes and figures
- Fixed incorrect `here().position()`[3.12.2] when `place`[3.10.19] was used in a context expression
- Fixed resolved positions of elements (in particular, headings) whose show rule emits an invisible element (like a state update) before a page break
- Fixed behavior of stepping a counter at a deeper level than its current state has
- Fixed citation formatting not working in table headers and a few other places
- Displaying the footnote counter will now respect the footnote numbering style

## Model

- Document set rules do not need to be at the very start of the document anymore. The only restriction is that they must not occur inside of layout containers.
- The `spacing` property of lists[3.6.2/3.5], enumerations[3.6.11/3.8], and term lists[3.6.20/3.5] is now also respected for tight lists
- Tight lists now only attach (with tighter spacing) to preceding paragraphs, not arbitrary blocks
- The `quote`[3.6.16] element is now locatable (can be used in queries)
- The bibliography heading now uses `depth` instead of `level` so that its level can still be configured via a show-set rule
- Added support for more numbering[3.6.12] formats: Devanagari, Eastern Arabic, Bengali, and circled numbers
- Added `hanging-indent`[3.6.9/5.8] parameter to heading function to tweak the appearance of multi-line headings and improved default appearance of multi-line headings
- Improved handling of bidirectional text in outline entry
- Fixed document set rules being ignored in an otherwise empty document
- Fixed document set rules not being usable in context expressions
- Fixed bad interaction between `set document` and `set page`
- Fixed `show figure: set align(..)`. Since the default figure alignment is now a show-set rule, it is not revoked by `show figure: it => it.body` anymore. **(Minor breaking change)**
- Fixed numbering of footnote references
- Fixed spacing after bibliography heading

## Bibliography

- The Hayagriva YAML `publisher` field can now accept a dictionary with a `location` key. The top-level `location` key is now primarily intended for event and item locations.
- Multiple page ranges with prefixes and suffixes are now allowed
- Added `director` and catch-all editor types to BibLaTeX parsing

- Added support for disambiguation to alphanumeric citation style
- The year 0 will now render as 1BC
- Fixes for sorting of bibliography entries
- Fixed pluralization of page range labels
- Fixed sorting of citations by their number
- Fixed how citation number ranges collapse
- Fixed when the short form of a title is used
- Fixed parsing of unbalanced dollars in BibLaTeX `url` field
- Updated built-in citation styles

## Visualization

- Added `fill-rule` parameter to `path` and `polygon`[3.11.9/2.2] functions
- Fixed color mixing and gradients for Luma colors[3.11.3/4.1]
- Fixed conversion from Luma to CMYK colors
- Fixed offset gradient strokes in PNG export
- Fixed unintended cropping of some SVGs
- SVGs with foreign objects now produce a warning as they will likely not render correctly in Typst

## Syntax

- Added support for nested imports like `import "file.typ": module.item`
- Added support for parenthesized imports like `import "file.typ": (a, b, c)`. With those, the import list can break over multiple lines.
- Fixed edge case in parsing of reference syntax
- Fixed edge case in parsing of heading, list, enum, and term markers immediately followed by comments
- Fixed rare crash in parsing of parenthesized expressions

## Scripting

- Added new fixed-point `decimal`[3.5.11] number type for highly precise arithmetic on numbers in base 10, as needed for finance
- Added `std` module for accessing standard library definitions even when a variable with the same name shadows/overwrites it
- Added `array.to-dict`[3.5.2/3.32], `array.reduce`[3.5.2/3.33], `array.windows`[3.5.2/3.29] methods
- Added `exact` argument to `array.zip`[3.5.2/3.17]
- Added `arguments.at`[3.5.1/4.2] method
- Added `int.from-bytes`[3.5.17/5.8], `int.to-bytes`[3.5.17/5.9], `float.from-bytes`[3.5.15/4.4], and `float.to-bytes`[3.5.15/4.5]
- Added proper support for negative values of the `digits` parameter of `calc.round`[3.5.9/2.31] (the behaviour existed before but was subtly broken)
- Conversions from `int`[3.5.17] to `float`[3.5.15] will now error instead of saturating if the float is too large **(Minor breaking change)**
- Added `float.nan`[3.5.15/3.2] and `float.inf`[3.5.15/3.1], removed `calc.nan` **(Minor breaking change)**
- Certain symbols are now generally callable like functions and not only specifically in math. Examples are accents or `floor`[3.8.20/2.5] and `ceil`[3.8.20/2.6].
- Improved `repr`[3.5.25] of relative values, sequences, infinities, NaN, `type(none)` and `type(auto)`
- Fixed crash on whole packages (rather than just files) cyclically importing each other
- Fixed return type of `calc.round`[3.5.9/2.31] on integers when a non-zero value is provided for `digits`

## Styling

- Text show rules now match across multiple text elements
- The string `"` in a text show rule now matches smart quotes
- Fixed a long-standing styling bug where the header and footer would incorrectly inherit styles from a lone element on the page (e.g. a heading)

- Fixed `set page` not working directly after a counter/state update
- Page fields configured via an explicit `page(..)[..]` call can now be properly retrieved in context expressions

## Export

- Highly reduced PDF file sizes due to better font subsetting
- Emoji are now exported properly in PDF
- Added initial support for PDF/A. For now, only the standard PDF/A-2b is supported, but more is planned for the future. Enabled via `--pdf-standard a-2b` in the CLI and via the UI in File > Export as > PDF in the web app.
- Setting `page.fill`[3.10.17/4.9] to `none` will now lead to transparent pages instead of white ones in PNG and SVG. The new default of `auto` means transparent for PDF and white for PNG and SVG.
- Improved text copy-paste from PDF in complex scenarios
- Exported SVGs now contain the `data-typst-label` attribute on groups resulting from labelled boxes[3.10.5] and blocks[3.10.4]
- Fixed a bug where some fonts would not print correctly on professional printers
- Fixed a bug where transparency could leak from one PDF object to another
- Fixed a bug with CMYK gradients in PDF
- Fixed various bugs with export of Oklab gradients in PDF
- Fixed crashes related to rendering of non-outline glyphs
- Two small fixes for PDF standard conformance

## Performance

- Typst's layout engine is now multithreaded. Typical speedups are 2-3x for larger documents. The multithreading operates on page break boundaries, so explicit page breaks are necessary for it to kick in.
- Paragraph justification was optimized with a new two-pass algorithm. Speedups are larger for shorter paragraphs and range from 1-6x.

## Command Line Interface

- Added `--pages` option to select specific page ranges to export
- Added `--package-path` and `--package-cache-path` as well as `TYPST_PACKAGE_PATH` and `TYPST_PACKAGE_CACHE_PATH` environment variables for configuring where packages are loaded from and cached in, respectively
- Added `--ignore-system-fonts` flag to disable system fonts fully for better reproducibility
- Added `--make-deps` argument for outputting the dependencies of the current compilation as a Makefile
- Added `--pretty` option to `typst query`, with the default now being to minify (only applies to JSON format)
- Added `--backup-path` to `typst update` to configure where the previous version is backed up
- Added useful links to help output
- The CLI will now greet users who invoke just `typst` for the first time
- The document can now be written to stdout by passing `-` as the output filename (for PDF or single-page image export)
- Typst will now emit a proper error message instead of failing silently when the certificate specified by `--cert` or `TYPST_CERT` could not be loaded
- The CLI now respects the `SOURCE_DATE_EPOCH` environment variable for better reproducibility
- When exporting multiple images, you can now use `t` (total pages), `p` (current page), and `0p` (zero-padded current page, same as current `n`) in the output path
- The input and output paths now allow non-UTF-8 values
- Times are now formatted more consistently across the CLI
- Fixed a bug related to the `--open` flag
- Fixed path completions for `typst` not working in zsh

## Tooling and Diagnostics

- The "compiler" field for specifying the minimum Typst version required by a package now supports imprecise bounds like 0.11 instead of 0.11.0
- Added warning when a label is ignored by Typst because no preceding labellable element exists
- Added hint when trying to apply labels in code mode
- Added hint when trying to call a standard library function that has been shadowed/overwritten by a local definition
- Added hint when trying to set both the language and the region in the `lang` parameter
- Added hints when trying to compile non-Typst files (e.g. after having typed `typst c file.pdf` by accident)
- Added hint when a string is used where a label is expected
- Added hint when a stray end of a block comment (`*/`) is encountered
- Added hints when destructuring arrays with the wrong number of elements
- Improved error message when trying to use a keyword as an identifier in a let binding
- Improved error messages when accessing nonexistent fields
- Improved error message when a package exists, but not the specified version
- Improved hints for unknown variables
- Improved hint when trying to convert a length with non-zero em component to an absolute unit
- Fixed a crash that could be triggered by certain hover tooltips
- Fixed an off-by-one error in to-source jumps when first-line-indent is enabled
- Fixed suggestions for `.` after the end of an inline code expressions
- Fixed autocompletions being duplicated in a specific case

## Symbols

- New: `parallelogram`, `original`, `image`, `crossmark`, `rest`, `natural`, `flat`, `sharp`, `tiny`, `miny`, `copyleft`, `trademark`, `emoji.beet`, `emoji.fingerprint`, `emoji.harp`, `emoji.shovel`, `emoji.splatter`, `emoji.tree.leafless`,
- New variants: `club.stroked`, `diamond.stroked`, `heart.stroked`, `spade.stroked`, `gt.neq`, `lt.neq`, `checkmark.heavy`, `paren.double`, `brace.double`, `shell.double`, `arrow.turn`, `plus.double`, `plus.triple`, `infinity.bar`, `infinity.incomplete`, `infinity.tie`, `multimap.double`, `ballot.check`, `ballot.check.heavy`, `emptyset.bar`, `emptyset.circle`, `emptyset.arrow.l`, `emptyset.arrow.r`, `parallel.struck`, `parallel.eq`, `parallel.equiv`, `parallel.slanted`, `parallel.tilde`, `angle.l.curly`, `angle.l.dot`, `angle.r.curly`, `angle.r.dot`, `angle.oblique`, `angle.s`, `em.two`, `em.three`
- Renamed: `turtle` to `shell`, `notes` to `note`, `ballot.x` to `ballot.cross`, `succ.eq` to `succ.curly.eq`, `prec.eq` to `prec.curly.eq`, `servicemark` to `trademark.service`, `emoji.face.tired` to `emoji.face.distress` **(Breaking change)**
- Changed codepoint: `prec.eq`, `prec.neq`, `succ.eq`, `succ.neq`, `triangle` from ▷ to △, `emoji.face.tired` **(Breaking change)**
- Removed: `lt.curly` in favor of `prec`, `gt.curly` in favor of `succ` **(Breaking change)**

## Deprecations

- `counter.display`[3.12.1/11.2] without an established context
- `counter.final`[3.12.1/11.4] with a location
- `state.final`[3.12.7/6.3] with a location
- `state.display`
- `query`[3.12.6] with a location as the second argument
- `locate`[3.12.3] with a callback function
- `measure`[3.10.14] with styles
- `style`

## Development

- Added `typst-kit` crate which provides useful APIs for `World` implementors

- Added go-to-definition API in `typst-ide`
- Added package manifest parsing APIs to `typst-syntax`
- As the compiler is now capable of multithreading, `World` implementations must satisfy `Send` and `Sync`
- Changed signature of `World::main` to allow for the scenario where the main file could not be loaded
- Removed `Tracer` in favor of `Warned<T>` and `typst::trace` function
- The `xz2` dependency used by the self-updater is now statically linked
- The Dockerfile now has an `ENTRYPOINT` directive

# Typst 0.11.1 (May 17, 2024)

## Security

- Fixed a vulnerability where image files at known paths could be embedded into the PDF even if they were outside of the project directory

## Bibliography

- Fixed et-al handling in subsequent citations
- Fixed suppression of title for citations and bibliography references with no author
- Fixed handling of initials in citation styles without a delimiter
- Fixed bug with citations in footnotes

## Text and Layout

- Fixed interaction of `first-line-indent`[3.6.14/4.6] and `outline`[3.6.13]
- Fixed compression of CJK punctuation marks at line start and end
- Fixed handling of rectangles[3.11.10] with negative dimensions
- Fixed layout of `path` in explicitly sized container
- Fixed broken `raw`[3.7.6] text in right-to-left paragraphs
- Fixed tab rendering in `raw` text with language `typ` or `typc`
- Fixed highlighting of multi-line `raw` text enclosed by single backticks
- Fixed indentation of overflowing lines in `raw` blocks
- Fixed extra space when `raw` text ends with a backtick

## Math

- Fixed broken equations[3.8.16] in right-to-left paragraphs
- Fixed missing blackboard bold[3.8.27/1.5] letters
- Fixed error on empty arguments in 2D math argument list
- Fixed stretching via `mid`[3.8.20/2.2] for various characters
- Fixed that alignment points in equations were affected by `set align(..)`

## Export

- Fixed smart quotes[3.7.8] in PDF outline
- Fixed patterns[3.11.13] with spacing in PDF
- Fixed wrong PDF page labels when page numbering[3.10.17/4.10] was disabled after being previously enabled

## Scripting

- Fixed overflow for large numbers in external data files (by converting to floats instead)
- Fixed `str.trim(regex, at: end)`[3.5.28/4.19] when the whole string is matched

## Miscellaneous

- Fixed deformed strokes for specific shapes and thicknesses
- Fixed newline handling in code mode: There can now be comments within chained method calls and between an `if` branch and the `else` keyword
- Fixed inefficiency with incremental reparsing
- Fixed autocompletions for relative file imports
- Fixed crash in autocompletion handler
- Fixed a bug where the path and entrypoint printed by `typst init` were not properly escaped
- Fixed various documentation errors

# Typst 0.11.0 (March 15, 2024)

## Tables

- Tables are now *much* more flexible, read the new Table Guide[4.3] to get started
- Added `table.cell`[3.6.19/4.1] element for per-cell configuration
- Cells can now span multiple columns[3.6.19/4.1.4] or rows[3.6.19/4.1.5]
- The stroke[3.6.19/4.1.9] of individual cells can now be customized
- The `align`[3.6.19/3.7] and `inset`[3.6.19/3.6] arguments of the table function now also take `(x, y) => ..` functions
- Added `table.hline`[3.6.19/4.2] and `table.vline`[3.6.19/4.3] for convenient line customization
- Added `table.header`[3.6.19/4.4] element for table headers that repeat on every page
- Added `table.footer`[3.6.19/4.5] element for table footers that repeat on every page
- All the new table functionality is also available for grids[3.10.10]
- Fixed gutter-related bugs

*Thanks to @PgBiel[A] for his work on tables!*

## Templates

- You can now use template packages to get started with new projects. Click *Start from template* on the web app's dashboard and choose your preferred template or run the `typst init <template>` command in the CLI. You can browse the available templates here[B].
- Switching templates after the fact has become easier. You can just import a styling function from a different template package.
- Package authors can now submit their own templates to the package repository[C]. Share a template for a paper, your institution, or an original work to help the community get a head start on their projects.
- Templates and packages are now organized by category and discipline. Filter packages by either taxonomy in the *Start from template* wizard. If you are a package author, take a look at the new documentation for categories[D] and disciplines[E].

## Context

- Added *context expressions:* Read the chapter on context[3.4] to get started
- With context, you can access settable properties, e.g. `context text.lang` to access the language set via `set text(lang: "..")`
- The following existing functions have been made contextual: `query`[3.12.6], `locate`[3.12.3], `measure`[3.10.14], `counter.display`[3.12.1/11.2], `counter.at`[3.12.1/11.3], `counter.final`[3.12.1/11.4], `state.at`[3.12.7/6.2], and `state.final`[3.12.7/6.3]
- Added contextual methods `counter.get`[3.12.1/11.1] and `state.get`[3.12.7/6.1] to retrieve the value of a counter or state in the current context
- Added contextual function `here`[3.12.2] to retrieve the location[3.12.4] of the current context
- The `locate`[3.12.3] function now returns the location of a selector's unique match. Its old behavior has been replaced by context expressions and only remains temporarily available for compatibility.
- The `counter.at`[3.12.1/11.3] and `state.at`[3.12.7/6.2] methods are now more flexible: They directly accept any kind of locatable[3.12.4/1] selector with a unique match (e.g. a label) instead of just locations
- When context is available, `counter.display`[3.12.1/11.2] now directly returns the result of applying the numbering instead of yielding opaque content. It should not be used anymore without context. (Deprecation planned)
- The `state.display` function should not be used anymore, use `state.get`[3.12.7/6.1] instead (Deprecation planned)
- The `location` argument of `query`[3.12.6], `counter.final`[3.12.1/11.4], and `state.final`[3.12.7/6.3] should not be used anymore (Deprecation planned)

[A] https://github.com/PgBiel
[B] https://typst.app/universe/search/?kind=templates
[C] https://github.com/typst/packages
[D] https://github.com/typst/packages/blob/main/docs/CATEGORIES.md
[E] https://github.com/typst/packages/blob/main/docs/DISCIPLINES.md

- The `styles` argument of the `measure` function should not be used anymore (Deprecation planned)
- The `style` function should not be used anymore, use context instead (Deprecation planned)
- The correct context is now also provided in various other places where it is available, e.g. in show rules, layout callbacks, and numbering functions in the outline

## Styling

- Fixed priority of multiple show-set rules[3.2.2]: They now apply in the same order as normal set rules would
- Show-set rules on the same element (e.g. `show heading.where(level: 1): set heading(numbering: "1.")`) now work properly
- Setting properties on an element within a transformational show rule (e.g. `show heading: it => { set heading(..); it }`) is **not** supported anymore (previously it also only worked sometimes); use show-set rules instead **(Breaking change)**
- Text show rules that match their own output now work properly (e.g. `show "cmd": \`cmd\``)
- The elements passed to show rules and returned by queries now contain all fields of their respective element functions rather than just specific ones
- All settable properties can now be used in where[3.5.16/8.2] selectors
- And[3.5.26/3.2] and or[3.5.26/3.1] selectors can now be used with show rules
- Errors within show rules and context expressions are now ignored in all but the last introspection iteration, in line with the behavior of the old `locate`[3.12.3]
- Fixed a bug where document set rules were allowed after content

## Layout

- Added `reflow` argument to `rotate`[3.10.23] and `scale`[3.10.24] which lets them affect the layout
- Fixed a bug where floating placement[3.10.19/4.3] or floating figures[3.6.7/5.3] could end up out of order
- Fixed overlap of text and figure for full-page floating figures
- Fixed various cases where the `hide`[3.10.11] function didn't hide its contents properly
- Fixed usage of `h`[3.10.26] and `v`[3.10.27] in stacks[3.10.28]
- Invisible content like a counter update will no longer force a visible block for just itself
- Fixed a bug with horizontal spacing followed by invisible content (like a counter update) directly at the start of a paragraph

## Text

- Added `stroke`[3.7.12/2.8] property for text
- Added basic i18n for Serbian and Catalan
- Added support for contemporary Japanese numbering[3.6.12] method
- Added patches for various wrong metadata in specific fonts
- The text direction[3.7.12/2.19] can now be overridden within a paragraph
- Fixed Danish smart quotes[3.7.8]
- Fixed font fallback next to a line break
- Fixed width adjustment of JIS-style Japanese punctuation
- Fixed Finnish translation of "Listing"
- Fixed Z-ordering of multiple text decorations (underlines, etc.)
- Fixed a bug due to which text features[3.7.12/2.32] could not be overridden in consecutive set rules

## Model

- Added `depth`[3.6.9/5.2] and `offset`[3.6.9/5.3] arguments to heading to increase or decrease the heading level for a bunch of content; the heading syntax now sets `depth` rather than `level` **(Breaking change)**
- List markers[3.6.2/3.2] now cycle by default
- The `quote`[3.6.16] function now more robustly selects the correct quotes based on language and nesting
- Fixed indent bugs related to the default show rule of terms[3.6.20]

## Math

- Inline equations now automatically linebreak at appropriate places
- Added `number-align`[3.8.16/3.3] argument to equations
- Added support for adjusting the `size`[3.8.9/2.3] of accents relative to their base
- Improved positioning of accents
- Primes[3.8.22] are now always attached as scripts[3.8.10/3.2] by default
- Exposed `math.primes`[3.8.22] element which backs the `$f'$` syntax in math
- Math mode is not affected by `strong`[3.6.18] and `emph`[3.6.6] anymore
- Fixed `attach`[3.8.10/3.1] under fractions[3.8.19]
- Fixed that `math.class`[3.8.14] did not affect smart limit placement
- Fixed weak spacing in `lr`[3.8.20/2.1] groups
- Fixed layout of large operators for Cambria Math font
- Fixed math styling of Hebrew symbol codepoints

## Symbols

- Added `gradient` as an alias for `nabla`
- Added `partial` as an alias for `diff`, `diff` will be deprecated in the future
- Added `colon.double`, `gt.approx`, `gt.napprox`, `lt.approx`, and `lt.napprox`
- Added `arrow.r.tilde` and `arrow.l.tilde`
- Added `tilde.dot`
- Added `forces` and `forces.not`
- Added `space.nobreak.narrow`
- Added `lrm` (Left-to-Right Mark) and `rlm` (Right-to-Left Mark)
- Fixed `star.stroked` symbol (which previously had the wrong codepoint)

## Scripting

- Arrays can now be compared lexicographically
- Added contextual method `to-absolute`[3.10.13/3.5] to lengths
- Added `calc.root`[3.5.9/2.5]
- Added `int.signum`[3.5.17/5.1] and `float.signum`[3.5.15/4.3] methods
- Added `float.is-nan`[3.5.15/4.1] and `float.is-infinite`[3.5.15/4.2] methods
- Added `int.bit-not`[3.5.17/5.2], `int.bit-and`[3.5.17/5.3], `int.bit-or`[3.5.17/5.4], `int.bit-xor`[3.5.17/5.5], `int.bit-lshift`[3.5.17/5.6], and `int.bit-rshift`[3.5.17/5.7] methods
- Added `array.chunks`[3.5.2/3.28] method
- A module can now be converted to a dictionary with the dictionary constructor[3.5.12/2] to access its contents dynamically
- Added `row-type`[3.13.3/2.3] argument to `csv` function to configure how rows will be represented
- XML parsing[3.13.7] now allows DTDs (document type definitions)
- Improved formatting of negative numbers with `str`[3.5.28] and `repr`[3.5.25]
- For loops can now iterate over bytes[3.5.7]
- Fixed a bug with pattern matching in for loops
- Fixed a bug with labels not being part of `.fields()`[3.5.8/3.4] dictionaries
- Fixed a bug where unnamed argument sinks wouldn't capture excess arguments
- Fixed typo in `repr` output of strokes

## Syntax

- Added support for nested destructuring patterns[3.3.3]
- Special spaces (like thin or non-breaking spaces) are now parsed literally instead of being collapsed into normal spaces **(Breaking change)**
- Korean text can now use emphasis syntax without adding spaces **(Breaking change)**
- The token `context`[3.4] is now a keyword and cannot be used as an identifier anymore **(Breaking change)**
- Nested line comments aren't allowed anymore in block comments **(Breaking change)**

- Fixed a bug where `x.)` would be treated as a field access
- Text elements can now span across curly braces in markup
- Fixed silently wrong parsing when function name is parenthesized
- Fixed various bugs with parsing of destructuring patterns, arrays, and dictionaries

## Tooling & Diagnostics

- Click-to-jump now works properly within `raw`[3.7.6] text
- Added suggestion for accessing a field if a method doesn't exist
- Improved hint for calling a function stored in a dictionary
- Improved errors for mutable accessor functions on arrays and dictionaries
- Fixed error message when calling constructor of type that doesn't have one
- Fixed confusing error message with nested dictionaries for strokes on different sides
- Fixed autocompletion for multiple packages with the same name from different namespaces

## Visualization

- The `image`[3.11.7] function doesn't upscale images beyond their natural size anymore
- The `image`[3.11.7] function now respects rotation stored in EXIF metadata
- Added support for SVG filters
- Added alpha component to `luma`[3.11.3/4.1] colors
- Added `color.transparentize`[3.11.3/4.19] and `color.opacify`[3.11.3/4.20] methods
- Improved `color.negate`[3.11.3/4.16] function
- Added `stroke`[3.7.1/2.2] and `radius`[3.7.1/2.6] arguments to `highlight` function
- Changed default `highlight`[3.7.1] color to be transparent
- CMYK to RGB conversion is now color-managed
- Fixed crash with gradients in Oklch color space
- Fixed color-mixing for hue-based spaces
- Fixed bugs with color conversion
- SVG sizes are not rounded anymore, preventing slightly wrong aspect ratios
- Fixed a few other SVG-related bugs
- `color.components`[3.11.3/4.9] doesn't round anything anymore

## Export

- PDFs now contain named destinations for headings derived from their labels
- The internal PDF structure was changed to make it easier for external tools to extract or modify individual pages, avoiding a bug with Typst PDFs in Apple Preview
- PDFs produced by Typst should now be byte-by-byte reproducible when `set document(date: none)` is set
- Added missing flag to PDF annotation
- Fixed multiple bugs with gradients in PDF export
- Fixed a bug with patterns in PDF export
- Fixed a bug with embedding of grayscale images in PDF export
- Fixed a bug with To-Unicode mapping of CFF fonts in PDF export
- Fixed a bug with the generation of the PDF outline
- Fixed a sorting bug in PDF export leading to non-reproducible output
- Fixed a bug with transparent text in PNG export
- Exported SVG files now include units in their top-level `width` and `height`

## Command line interface

- Added support for passing `inputs`[3.5.27] via a CLI flag
- When passing the filename `-`, Typst will now read input from stdin
- Now uses the system-native TLS implementation for network fetching which should be generally more robust

- Watch mode will now properly detect when a previously missing file is created
- Added `--color` flag to configure whether to print colored output
- Fixed user agent with which packages are downloaded
- Updated bundled fonts to the newest versions

## Development

- Added `--vendor-openssl` to CLI to configure whether to link OpenSSL statically instead of dynamically (not applicable to Windows and Apple platforms)
- Removed old tracing (and its verbosity) flag from the CLI
- Added new `--timings` flag which supersedes the old flamegraph profiling in the CLI
- Added minimal CLI to `typst-docs` crate for extracting the language and standard library documentation as JSON
- The `typst_pdf::export` function's `ident` argument switched from `Option` to `Smart`. It should only be set to `Smart::Custom` if you can provide a stable identifier (like the web app can). The CLI sets `Smart::Auto`.

# Typst 0.10.0 (December 04, 2023)

## Bibliography management

- Added support for citation collapsing (e.g. `[1]–[3]` instead of `[1], [2], [3]`) if requested by a CSL style
- Fixed bug where an additional space would appear after a group of citations
- Fixed link show rules for links in the bibliography
- Fixed show-set rules on citations
- Fixed bibliography-related crashes that happened on some systems
- Corrected name of the GB/T 7714 family of styles from 7114 to 7714
- Fixed missing title in some bibliography styles
- Fixed printing of volumes in some styles
- Fixed delimiter order for contributors in some styles (e.g. APA)
- Fixed behavior of alphanumeric style
- Fixed multiple bugs with GB/T 7714 style
- Fixed escaping in Hayagriva values
- Fixed crashes with empty dates in Hayagriva files
- Fixed bug with spacing around math blocks
- Fixed title case formatting after verbatim text and apostrophes
- Page ranges in `.bib` files can now be arbitrary strings
- Multi-line values in `.bib` files are now parsed correctly
- Entry keys in `.bib` files now allow more characters
- Fixed error message for empty dates in `.bib` files
- Added support for years of lengths other than 4 without leading zeros in `.bib` files
- More LaTeX commands (e.g. for quotes) are now respected in `.bib` files

## Visualization

- Added support for patterns[3.11.13] as fills and strokes
- The `alpha` parameter of the `components`[3.11.3/4.9] function on colors is now a named parameter **(Breaking change)**
- Added support for the Oklch[3.11.3/4.3] color space
- Improved conversions between colors in different color spaces
- Removed restrictions on Oklab[3.11.3/4.2] chroma component
- Fixed clipping[3.10.4/2.12] on blocks and boxes without a stroke
- Fixed bug with gradients[3.11.6] on math
- Fixed bug with gradient rotation on text
- Fixed bug with gradient colors in PDF
- Fixed relative base of Oklab chroma ratios
- Fixed Oklab color negation

## Text and Layout

- CJK text can now be emphasized with the * and _ syntax even when there are no spaces
- Added basic i18n for Greek and Estonian
- Improved default figure caption separator[3.6.7/6.1.2] for Chinese, French, and Russian
- Changed default figure supplement[3.6.7/5.7] for Russian to short form
- Fixed CJK-Latin-spacing[3.7.12/2.11] before line breaks and in `locate`[3.12.3] calls
- Fixed line breaking at the end of links

## Math

- Added `mid`[3.8.20/2.2] function for scaling a delimiter up to the height of the surrounding `lr`[3.8.20/2.1] group
- The op[3.8.26] function can now take any content, not just strings
- Improved documentation for math alignment[3.8.5]
- Fixed swallowing of trailing comma when a symbol is used in a function-like way (e.g. `pi(a,b,)`)

## Scripting

- Any non-identifier dictionary key is now interpreted as an expression: For instance, `((key): value)` will create a dictionary with a dynamic key
- The `stroke`[3.11.12] type now has a constructor that converts a value to a stroke or creates one from its parts
- Added constructor for `arguments`[3.5.1] type
- Added `calc.div-euclid`[3.5.9/2.38] and `calc.rem-euclid`[3.5.9/2.39] functions
- Fixed equality of `arguments`[3.5.1]
- Fixed `repr`[3.5.25] of `cmyk`[3.11.3/4.6] colors
- Fixed crashes with provided elements like figure captions, outline entries, and footnote entries

## Tooling and Diagnostics

- Show rules that match on their own output now produce an appropriate error message instead of a crash (this is a first step, in the future they will just work)
- Too highly or infinitely nested layouts now produce error messages instead of crashes
- Added hints for invalid identifiers
- Added hint when trying to use a manually constructed footnote or outline entry
- Added missing details to autocompletions for types
- Improved error message when passing a named argument where a positional one is expected
- Jump from click now works on raw blocks

## Export

- PDF compilation output is now again fully byte-by-byte reproducible if the document's `date`[3.6.5/3.7] is set manually
- Fixed color export in SVG
- Fixed PDF metadata encoding of multiple authors[3.6.5/3.4]

## Command line interface

- Fixed a major bug where `typst watch` would confuse files and fail to pick up updates
- Fetching of the release metadata in `typst update` now respects proxies
- Fixed bug with `--open` flag on Windows when the path contains a space
- The `TYPST_FONT_PATHS` environment variable can now contain multiple paths (separated by `;` on Windows and `:` elsewhere)
- Updated embedded New Computer Modern fonts to version 4.7
- The watching process doesn't stop anymore when the main file contains invalid UTF-8

## Miscellaneous Improvements

- Parallelized image encoding in PDF export
- Improved the internal representation of content for improved performance
- Optimized introspection (query, counter, etc.) performance
- The document title[3.6.5/3.3] can now be arbitrary content instead of just a string
- The `number-align`[3.6.11/3.9] parameter on numbered lists now also accepts vertical alignments
- Fixed selectors on `quote`[3.6.16] elements
- Fixed parsing of `#return` expression in markup
- Fixed bug where inline equations were displayed in equation outlines
- Fixed potential CRLF issue in `raw`[3.7.6] blocks
- Fixed a bug where Chinese numbering couldn't exceed the number 255

## Development

- Merged `typst` and `typst-library` and extracted `typst-pdf`, `typst-svg`, and `typst-render` into separate crates
- The Nix flake now includes the git revision when running `typst --version`

# Typst 0.9.0 (October 31, 2023)

## Bibliography management

- New bibliography engine based on CSL[A] (Citation Style Language). Ships with about 100 commonly used citation styles and can load custom `.csl` files.
- Added new `form`[3.6.3/3.3] argument to the `cite` function to produce different forms of citations (e.g. for producing a citation suitable for inclusion in prose)
- The `cite`[3.6.3] function now takes only a single label/key instead of allowing multiple. Adjacent citations are merged and formatted according to the citation style's rules automatically. This works both with the reference syntax and explicit calls to the `cite` function. **(Breaking change)**
- The `cite` function now takes a label[3.5.18] instead of a string **(Breaking change)**
- Added `full`[3.6.1/4.3] argument to bibliography function to print the full bibliography even if not all works were cited
- Bibliography entries can now contain Typst equations (wrapped in `$..$` just like in markup), this works both for `.yaml` and `.bib` bibliographies
- The hayagriva YAML format was improved. See its changelog[B] for more details. **(Breaking change)**
- A few bugs with `.bib` file parsing were fixed
- Removed `brackets` argument of `cite` function in favor of `form`

## Visualization

- Gradients and colors (thanks to @Dherse[C])
  - Added support for gradients[3.11.6] on shapes and text
  - Supports linear, radial, and conic gradients
  - Added support for defining colors in more color spaces, including Oklab[3.11.3/4.2], Linear RGB(A)[3.11.3/4.4], HSL[3.11.3/4.7], and HSV[3.11.3/4.8]
  - Added `saturate`[3.11.3/4.14], `desaturate`[3.11.3/4.15], and `rotate`[3.11.3/4.17] functions on colors
  - Added `color.map`[3.11.3/3.1] module with predefined color maps that can be used with gradients
  - Rename `kind` function on colors to `space`[3.11.3/4.10]
  - Removed `to-rgba`, `to-cmyk`, and `to-luma` functions in favor of a new `components`[3.11.3/4.9] function
- Improved rendering of rectangles[3.11.10] with corner radius and varying stroke widths
- Added support for properly clipping boxes[3.10.5/2.9] and blocks[3.10.4/2.12] with a border radius
- Added `background` parameter to `overline`[3.7.5], `underline`[3.7.13], and `strike`[3.7.9] functions
- Fixed inaccurate color embedding in PDFs
- Fixed ICC profile handling for images embedded in PDFs

## Text and Layout

- Added support for automatically adding proper spacing[3.7.12/2.11] between CJK and Latin text (enabled by default)
- Added support for automatic adjustment of more CJK punctuation
- Added `quote`[3.6.16] element for inserting inline and block quotes with optional attributions
- Added `raw.line`[3.7.6/5.1] element for customizing the display of individual lines of raw text, e.g. to add line numbers while keeping proper syntax highlighting
- Added support for per-side inset[3.6.19/3.6] customization to table function
- Added Hungarian and Romanian translations
- Added support for Czech hyphenation
- Added support for setting custom smart quotes[3.7.8]
- The default figure separator[3.6.7/6.1.2] now reacts to the currently set language and region
- Improved line breaking of links / URLs (especially helpful for bibliographies with many URLs)
- Improved handling of consecutive hyphens in justification algorithm
- Fixed interaction of justification and hanging indent

[A] https://citationstyles.org/
[B] https://github.com/typst/hayagriva/blob/main/CHANGELOG.md
[C] https://github.com/Dherse

- Fixed a bug with line breaking of short lines without spaces when justification is enabled
- Fixed font fallback for hyphen generated by hyphenation
- Fixed handling of word joiner and other no-break characters during hyphenation
- Fixed crash when hyphenating after an empty line
- Fixed line breaking of composite emoji like 🏳️‍🌈
- Fixed missing text in some SVGs
- Fixed font fallback in SVGs
- Fixed behavior of `to`[3.10.18/2.2] argument on `pagebreak` function
- Fixed `set align(..)` for equations
- Fixed spacing around `placed`[3.10.19] elements
- Fixed coalescing of `above`[3.10.4/2.10] and `below`[3.10.4/2.11] spacing if given in em units and the font sizes differ
- Fixed handling of `extent` parameter of `underline`[3.7.13], `overline`[3.7.5], and `strike`[3.7.9] functions
- Fixed crash for floating placed elements[3.10.19/4.3] with no specified vertical alignment
- Partially fixed a bug with citations in footnotes

## Math

- Added `gap` argument for `vec`[3.8.28/2.3], `mat`[3.8.21/2.4], and `cases`[3.8.13/2.3] function
- Added `size` argument for `abs`[3.8.20/2.3], `norm`[3.8.20/2.4], `floor`[3.8.20/2.5], `ceil`[3.8.20/2.6], and `round`[3.8.20/2.7] functions
- Added `reverse`[3.8.13/2.2] parameter to cases function
- Added support for multinomial coefficients to `binom`[3.8.11] function
- Removed `rotation` argument on `cancel`[3.8.12] function in favor of a new and more flexible `angle` argument **(Breaking change)**
- Added `wide` constant, which inserts twice the spacing of `quad`
- Added `csch` and `sech` operators[3.8.26]
- ↼, ⇀, ↔, and ↦ can now be used as accents[3.8.9]
- Added `integral.dash`, `integral.dash.double`, and `integral.slash` symbols[3.9.2]
- Added support for specifying negative indices for augmentation[3.8.21/2.3] lines to position the line from the back
- Fixed default color of matrix augmentation[3.8.21/2.3] lines
- Fixed attachment of primes to inline expressions
- Math content now respects the text baseline[3.7.12/2.12] setting

## Performance

- Fixed a bug related to show rules in templates which would effectively disable incremental compilation in affected documents
- Micro-optimized code in several hot paths, which brings substantial performance gains, in particular in incremental compilations
- Improved incremental parsing, which affects the whole incremental compilation pipeline
- Added support for incremental parsing in the CLI
- Added support for incremental SVG encoding during PDF export, which greatly improves export performance for documents with many SVG

## Tooling and Diagnostics

- Improved autocompletion for variables that are in-scope
- Added autocompletion for package imports
- Added autocompletion for labels[3.5.18]
- Added tooltip that shows which variables a function captures (when hovering over the equals sign or arrow of the function)
- Diagnostics are now deduplicated
- Improved diagnostics when trying to apply unary + or − to types that only support binary + and −
- Error messages now state which label or citation key isn't present in the document or its bibliography

- Fixed a bug where function argument parsing errors were shadowed by function execution errors (e.g. when trying to call `array.sorted`[3.5.2/3.30] and passing the key function as a positional argument instead of a named one).

## Export

- Added support for configuring the document's creation `date`[3.6.5/3.7]. If the `date` is set to `auto` (the default), the PDF's creation date will be set to the current date and time.
- Added support for configuring document `keywords`[3.6.5/3.6]
- Generated PDFs now contain PDF document IDs
- The PDF creator tool metadata now includes the Typst version

## Web app

- Added version picker to pin a project to an older compiler version (with support for Typst 0.6.0+)
- Fixed desyncs between editor and compiler and improved overall stability
- The app now continues to highlight the document when typing while the document is being compiled

## Command line interface

- Added support for discovering fonts through fontconfig
- Now clears the screen instead of resetting the terminal
- Now automatically picks correct file extension for selected output format
- Now only regenerates images for changed pages when using `typst watch` with PNG or SVG export

## Miscellaneous Improvements

- Added `version`[3.5.32] type and `sys.version`[3.5.27/1.1] constant specifying the current compiler version. Can be used to gracefully support multiple versions.
- The U+2212 MINUS SIGN is now used when displaying a numeric value, in the `repr`[3.5.25] of any numeric value and to replace a normal hyphen in text mode when before a digit. This improves, in particular, how negative integer values are displayed in math mode.
- Added support for specifying a default value instead of failing for `remove` function in `array`[3.5.2/3.8] and `dictionary`[3.5.12/3.4]
- Simplified Page Setup Guide examples
- Switched the documentation from using the word "hashtag" to the word "hash" where appropriate
- Added support for `array.zip`[3.5.2/3.17] without any further arguments
- Fixed crash when a plugin tried to read out of bounds memory
- Fixed crashes when handling infinite `lengths`[3.10.13]
- Fixed introspection (mostly bibliography) bugs due to weak page break close to the end of the document

## Development

- Extracted `typst::ide` into separate `typst_ide` crate
- Removed a few remaining `'static` bounds on `&dyn World`
- Removed unnecessary dependency, which reduces the binary size
- Fixed compilation of `typst` by itself (without `typst-library`)
- Fixed warnings with Nix flake when using `lib.getExe`

# Typst 0.8.0 (September 13, 2023)

## Scripting

- Plugins (thanks to @astrale-sharp[A] and @arnaudgolfouse[B])
  - Typst can now load plugins[3.5.23] that are compiled to WebAssembly
  - Anything that can be compiled to WebAssembly can thus be loaded as a plugin
  - These plugins are fully encapsulated (no access to file system or network)
  - Plugins can be shipped as part of packages[3.3.9]
  - Plugins work just the same in the web app
- Types are now first-class values **(Breaking change)**
  - A type[3.5.31] is now itself a value
  - Some types can be called like functions (those that have a constructor), e.g. `int`[3.5.17] and `str`[3.5.28]
  - Type checks are now of the form `type(10) == int` instead of the old `type(10) == "integer"`. Compatibility with the old way will remain for a while to give package authors time to upgrade, but it will be removed at some point.
  - Methods are now syntax sugar for calling a function scoped to a type, meaning that `"hello".len()` is equivalent to `str.len("hello")`
- Added support for `import`[3.3.8] renaming with `as`
- Added a `duration`[3.5.13] type
- Added support for CBOR[3.13.2] encoding and decoding
- Added encoding and decoding functions from and to bytes for data formats: `json.decode`, `json.encode`[3.13.4/4.1], and similar functions for other formats
- Added `array.intersperse`[3.5.2/3.27] function
- Added `str.rev`[3.5.28/4.21] function
- Added `calc.tau`[3.5.9/1.3] constant
- Made bytes[3.5.7] joinable and addable
- Made `array.zip`[3.5.2/3.17] function variadic
- Fixed bug with `eval`[3.5.14] when the `mode` was set to `"math"`
- Fixed bug with `ends-with`[3.5.28/4.13] function on strings
- Fixed bug with destructuring in combination with break, continue, and return
- Fixed argument types of hyperbolic functions[3.5.9/2.14], they don't allow angles anymore **(Breaking change)**
- Renamed some color methods: `rgba` becomes `to-rgba`, `cmyk` becomes `to-cmyk`, and `luma` becomes `to-luma` **(Breaking change)**

## Export

- Added SVG export (thanks to @Enter-tainer[C])
- Fixed bugs with PDF font embedding
- Added support for page labels that reflect the page numbering[3.10.17/4.10] style in the PDF

## Text and Layout

- Added `highlight`[3.7.1] function for highlighting text with a background color
- Added `polygon.regular`[3.11.9/3.1] function for drawing a regular polygon
- Added support for tabs in `raw`[3.7.6] elements alongside `tab-width`[3.7.6/4.7] parameter
- The layout engine now tries to prevent "runts" (final lines consisting of just a single word)
- Added Finnish translations
- Added hyphenation support for Polish
- Improved handling of consecutive smart quotes of different kinds
- Fixed vertical alignments for `number-align`[3.10.17/4.12] argument on page function **(Breaking change)**
- Fixed weak pagebreaks after counter updates
- Fixed missing text in SVG when the text font is set to "New Computer Modern"

[A] https://github.com/astrale-sharp
[B] https://github.com/arnaudgolfouse
[C] https://github.com/Enter-tainer

- Fixed translations for Chinese
- Fixed crash for empty text in show rule
- Fixed leading spaces when there's a linebreak after a number and a comma
- Fixed placement of floating elements in columns and other containers
- Fixed sizing of block containing just a single box

## Math

- Added support for augmented matrices[3.8.21/2.3]
- Removed support for automatic matching of fences like `|` and `||` as there were too many false positives. You can use functions like `abs`[3.8.20/2.3] or `norm`[3.8.20/2.4] or an explicit `lr`[3.8.20/2.1] call instead. **(Breaking change)**
- Fixed spacing after number with decimal point in math
- Fixed bug with primes in subscript
- Fixed weak spacing
- Fixed crash when text within math contains a newline

## Tooling and Diagnostics

- Added hints when trying to call a function stored in a dictionary without extra parentheses
- Fixed hint when referencing an equation without numbering
- Added more details to some diagnostics (e.g. when SVG decoding fails)

## Command line interface

- Added `typst update` command for self-updating the CLI (thanks to @jimvdl[A])
- Added download progress indicator for packages and updates
- Added `--format` argument to explicitly specify the output format
- The CLI now respects proxy configuration through environment variables and has a new `--cert` option for setting a custom CA certificate
- Fixed crash when field wasn't present and `--one` is passed to `typst query`

## Miscellaneous Improvements

- Added Page Setup Guide[4.2]
- Added `figure.caption`[3.6.7/6.1] function that can be used for simpler figure customization (**Breaking change** because `it.caption` now renders the full caption with supplement in figure show rules and manual outlines)
- Moved `caption-pos` argument to `figure.caption` function and renamed it to `position` **(Breaking change)**
- Added `separator`[3.6.7/6.1.2] argument to `figure.caption` function
- Added support for combination of and/or and before/after selectors[3.5.26]
- Packages can now specify a minimum compiler version[B] they require to work
- Fixed parser bug where method calls could be moved onto their own line for `#let` expressions in markup **(Breaking change)**
- Fixed bugs in sentence and title case conversion for bibliographies
- Fixed supplements for alphanumeric and author-title bibliography styles
- Fixed off-by-one error in APA bibliography style

## Development

- Made `Span` and `FileId` more type-safe so that all error conditions must be handled by `World` implementors

[A] https://github.com/jimvdl
[B] https://github.com/typst/packages#package-format

# Typst 0.7.0 (August 07, 2023)

## Text and Layout

- Added support for floating figures through the `placement`[3.6.7/5.3] argument on the figure function
- Added support for arbitrary floating content through the `float`[3.10.19/4.3] argument on the place function
- Added support for loading `.sublime-syntax` files as highlighting syntaxes[3.7.6/4.5] for raw blocks
- Added support for loading `.tmTheme` files as highlighting themes[3.7.6/4.6] for raw blocks
- Added *bounds* option to `top-edge`[3.7.12/2.14] and `bottom-edge`[3.7.12/2.15] arguments of text function for tight bounding boxes
- Removed nonsensical top- and bottom-edge options, e.g. *ascender* for the bottom edge **(Breaking change)**
- Added `script`[3.7.12/2.18] argument to text function
- Added `alternative`[3.7.8/3.3] argument to smart quote function
- Added basic i18n for Japanese
- Added hyphenation support for `nb` and `nn` language codes in addition to `no`
- Fixed positioning of placed elements[3.10.19] in containers
- Fixed overflowing containers due to optimized line breaks

## Export

- Greatly improved export of SVG images to PDF. Many thanks to @LaurenzV[A] for their work on this.
- Added support for the alpha channel of RGBA colors in PDF export
- Fixed a bug with PPI (pixels per inch) for PNG export

## Math

- Improved layout of primes (e.g. in `$a'_1$`)
- Improved display of multi-primes (e.g. in `$a''$`)
- Improved layout of roots[3.8.23/2.1]
- Changed relations to show attachments as limits[3.8.10/3.3] by default (e.g. in `$a ->^x b$`)
- Large operators and delimiters are now always vertically centered
- Boxes[3.10.5] in equations now sit on the baseline instead of being vertically centered by default. Notably, this does not affect blocks[3.10.4] because they are not inline elements.
- Added support for weak spacing[3.10.26/4.2]
- Added support for OpenType character variants
- Added support for customizing the math class[3.8.14] of content
- Fixed spacing around `.`, `\/`, and `...`
- Fixed spacing between closing delimiters and large operators
- Fixed a bug with math font weight selection
- Symbols and Operators **(Breaking changes)**
  - Added `id`, `im`, and `tr` text operators[3.8.26]
  - Renamed `ident` to `equiv` with alias `eq.triple` and removed `ident.strict` in favor of `eq.quad`
  - Renamed `ast.sq` to `ast.square` and `integral.sq` to `integral.square`
  - Renamed `.eqq` modifier to `.equiv` (and `.neqq` to `.nequiv`) for `tilde`, `gt`, `lt`, `prec`, and `succ`
  - Added `emptyset` as alias for `nothing`
  - Added `lt.curly` and `gt.curly` as aliases for `prec` and `succ`
  - Added `aleph`, `beth`, and `gimmel` as alias for `alef`, `bet`, and `gimel`

## Scripting

- Fields
  - Added `abs` and `em` field to lengths[3.10.13]
  - Added `ratio` and `length` field to relative lengths[3.10.21]
  - Added `x` and `y` field to 2d alignments[3.10.1/5.1]

[A] https://github.com/LaurenzV

  - Added `paint`, `thickness`, `cap`, `join`, `dash`, and `miter-limit` field to strokes[3.11.12]
- Accessor and utility methods
  - Added `dedup`[3.5.2/3.31] method to arrays
  - Added `pt`, `mm`, `cm`, and `inches` method to lengths[3.10.13]
  - Added `deg` and `rad` method to angles[3.10.3]
  - Added `kind`, `hex`, `rgba`, `cmyk`, and `luma` method to colors[3.11.3]
  - Added `axis`, `start`, `end`, and `inv` method to directions[3.10.28/3.1]
  - Added `axis` and `inv` method to alignments[3.10.1/5.1]
  - Added `inv` method to 2d alignments[3.10.1/5.1]
  - Added `start` argument to `enumerate`[3.5.2/3.16] method on arrays
- Added `color.mix`[3.11.3/4.18] function
- Added `mode` and `scope` arguments to `eval`[3.5.14] function
- Added `bytes`[3.5.7] type for holding large byte buffers
  - Added `encoding`[3.13.5/2.2] argument to read function to read a file as bytes instead of a string
  - Added `image.decode` function for decoding an image directly from a string or bytes
  - Added `bytes`[3.5.7] function for converting a string or an array of integers to bytes
  - Added `array`[3.5.2] function for converting bytes to an array of integers
  - Added support for converting bytes to a string with the `str`[3.5.28] function

## Tooling and Diagnostics

- Added support for compiler warnings
- Added warning when compilation does not converge within five attempts due to intense use of introspection features
- Added warnings for empty emphasis (`__` and `**`)
- Improved error message for invalid field assignments
- Improved error message after single #
- Improved error message when a keyword is used where an identifier is expected
- Fixed parameter autocompletion for functions that are in modules
- Import autocompletion now only shows the latest package version until a colon is typed
- Fixed autocompletion for dictionary key containing a space
- Fixed autocompletion for `for` loops

## Command line interface

- Added `typst query` subcommand to execute a query[3.12.6/3] on the command line
- The `--root` and `--font-path` arguments cannot appear in front of the command anymore **(Breaking change)**
- Local and cached packages are now stored in directories of the form `{namespace}/{name}/{version}` instead of `{namespace}/{name}-{version}` **(Breaking change)**
- Now prioritizes explicitly given fonts (via `--font-path`) over system and embedded fonts when both exist
- Fixed `typst watch` not working with some text editors
- Fixed displayed compilation time (now includes export)

## Miscellaneous Improvements

- Added `bookmarked`[3.6.9/5.7] argument to heading to control whether a heading becomes part of the PDF outline
- Added `caption-pos`[3.6.7/6.1.1] argument to control the position of a figure's caption
- Added `metadata`[3.12.5] function for exposing an arbitrary value to the introspection system
- Fixed that a `state`[3.12.7] was identified by the pair `(key, init)` instead of just its `key`
- Improved indent logic of enumerations[3.6.11]. Instead of requiring at least as much indent as the end of the marker, they now require only one more space indent than the start of the marker. As a result, even long markers like `12.` work with just 2 spaces of indent.
- Fixed bug with indent logic of `raw`[3.7.6] blocks
- Fixed a parsing bug with dictionaries

## Development

- Extracted parser and syntax tree into `typst-syntax` crate
- The `World::today` implementation of Typst dependents may need fixing if they have the same bug[A] that the CLI world had

[A] https://github.com/typst/typst/issues/1842

# Typst 0.6.0 (June 30, 2023)

## Package Management

- Typst now has built-in package management[3.3.9]
- You can import published[A] community packages or create and use system-local[B] ones
- Published packages are also supported in the web app

## Math

- Added support for optical size variants of glyphs in math mode
- Added argument to enable `limits`[3.8.10/3.3] conditionally depending on whether the equation is set in `display`[3.8.25/1.1] or `inline`[3.8.25/1.2] style
- Added `gt.eq.slant` and `lt.eq.slant` symbols
- Increased precedence of factorials in math mode (`$1/n!$` works correctly now)
- Improved underlines[3.8.15/1.1] and overlines[3.8.15/1.2] in math mode
- Fixed usage of `limits`[3.8.10/3.3] function in show rules
- Fixed bugs with line breaks in equations

## Text and Layout

- Added support for alternating page margins[3.10.17/4.5] with the `inside` and `outside` keys
- Added support for specifying the page `binding`[3.10.17/4.7]
- Added `to`[3.10.18/2.2] argument to pagebreak function to skip to the next even or odd page
- Added basic i18n for a few more languages (TR, SQ, TL)
- Fixed bug with missing table row at page break
- Fixed bug with underlines[3.7.13]
- Fixed bug superfluous table lines
- Fixed smart quotes after line breaks
- Fixed a crash related to text layout

## Command line interface

- **Breaking change:** Added requirement for `--root`/`TYPST_ROOT` directory to contain the input file because it designates the *project* root. Existing setups that use `TYPST_ROOT` to emulate package management should switch to local packages[C]
- **Breaking change:** Now denies file access outside of the project root
- Added support for local packages and on-demand package download
- Now watches all relevant files, within the root and all packages
- Now displays compilation time

## Miscellaneous Improvements

- Added `outline.entry`[3.6.13/5.1] to customize outline entries with show rules
- Added some hints for error messages
- Added some missing syntaxes for `raw`[3.7.6] highlighting
- Improved rendering of rotated images in PNG export and web app
- Made footnotes[3.6.8] reusable and referenceable
- Fixed bug with citations and bibliographies in `locate`[3.12.3]
- Fixed inconsistent tense in documentation

[A] https://typst.app/universe
[B] https://github.com/typst/packages#local-packages
[C] https://github.com/typst/packages#local-packages

## Development

- Added contribution guide[A]
- Reworked `World` interface to accommodate for package management and make it a bit simpler to implement *(Breaking change for implementors)*

[A] https://github.com/typst/typst/blob/main/CONTRIBUTING.md

# Typst 0.5.0 (June 09, 2023)

## Text and Layout

- Added `raw`[3.7.6] syntax highlighting for many more languages
- Added support for Korean numbering[3.6.12]
- Added basic i18n for a few more languages (NL, SV, DA)
- Improved line breaking for East Asian languages
- Expanded functionality of outline `indent`[3.6.13/4.4] property
- Fixed footnotes in columns
- Fixed page breaking bugs with footnotes[3.6.8]
- Fixed bug with handling of footnotes in lists, tables, and figures
- Fixed a bug with CJK punctuation adjustment
- Fixed a crash with rounded rectangles
- Fixed alignment of `line`[3.11.8] elements

## Math

- **Breaking change:** The syntax rules for mathematical attachments[3.8.10/3.1] were improved: `$f^abs(3)$` now parses as `$f^(abs(3))$` instead of `$(f^abs)(3)$`. To disambiguate, add a space: `$f^zeta (3)$`.
- Added forced size[3.8.25] commands for math (e.g., `display`[3.8.25/1.1])
- Added `supplement`[3.8.16/3.4] parameter to `equation`[3.8.16], used by references[3.6.17]
- New symbols[3.9.2]: `bullet`, `xor`, `slash.big`, `sigma.alt`, `tack.r.not`, `tack.r.short`, `tack.r.double.not`
- Fixed a bug with symbols in matrices
- Fixed a crash in the `attach`[3.8.10/3.1] function

## Scripting

- Added new `datetime`[3.5.10] type and `datetime.today`[3.5.10/5.1] to retrieve the current date
- Added `str.from-unicode`[3.5.28/4.9] and `str.to-unicode`[3.5.28/4.8] functions
- Added `fields`[3.5.8/3.4] method on content
- Added `base` parameter to `str`[3.5.28] function
- Added `calc.exp`[3.5.9/2.3] and `calc.ln`[3.5.9/2.20]
- Improved accuracy of `calc.pow`[3.5.9/2.2] and `calc.log`[3.5.9/2.19] for specific bases
- Fixed removal[3.5.12/3.4] order for dictionary
- Fixed `.at(default: ..)` for strings[3.5.28/4.4] and content[3.5.8/3.3]
- Fixed field access on styled elements
- Removed deprecated `calc.mod` function

## Command line interface

- Added PNG export via `typst compile source.typ output-{n}.png`. The output path must contain `{n}` if the document has multiple pages.
- Added `--diagnostic-format=short` for Unix-style short diagnostics
- Doesn't emit color codes anymore if stderr isn't a TTY
- Now sets the correct exit when invoked with a nonexistent file
- Now ignores UTF-8 BOM in Typst files

## Miscellaneous Improvements

- Improved errors for mismatched delimiters
- Improved error message for failed length comparisons
- Fixed a bug with images not showing up in Apple Preview
- Fixed multiple bugs with the PDF outline
- Fixed citations and other searchable elements in `hide`[3.10.11]
- Fixed bugs with reference supplements[3.6.17/4.2]

- Fixed Nix flake

# Typst 0.4.0 (May 20, 2023)

## Footnotes

- Implemented support for footnotes
- The `footnote`[3.6.8] function inserts a footnote
- The `footnote.entry`[3.6.8/4.1] function can be used to customize the footnote listing
- The `"chicago-notes"` citation style[3.6.3/3.4] is now available

## Documentation

- Added a Guide for LaTeX Users[4.1]
- Now shows default values for optional arguments
- Added richer outlines in "On this Page"
- Added initial support for search keywords: "Table of Contents" will now find the outline[3.6.13] function. Suggestions for more keywords are welcome!
- Fixed issue with search result ranking
- Fixed many more small issues

## Math

- **Breaking change**: Alignment points (&) in equations now alternate between left and right alignment
- Added support for writing roots with Unicode: For example, `$root(x+y)$` can now also be written as `$√(x+y)$`
- Fixed uneven vertical `attachment`[3.8.10/3.1] alignment
- Fixed spacing on decorated elements (e.g., spacing around a canceled[3.8.12] operator)
- Fixed styling for stretchable symbols
- Added `tack.r.double`, `tack.l.double`, `dotless.i` and `dotless.j` symbols[3.9.2]
- Fixed show rules on symbols (e.g. `show sym.tack: set text(blue)`)
- Fixed missing rename from `ast.op` to `ast` that should have been in the previous release

## Scripting

- Added function scopes: A function can now hold related definitions in its own scope, similar to a module. The new `assert.eq`[3.5.3/3.1] function, for instance, is part of the `assert`[3.5.3] function's scope. Note that function scopes are currently only available for built-in functions.
- Added `assert.eq`[3.5.3/3.1] and `assert.ne`[3.5.3/3.2] functions for simpler equality and inequality assertions with more helpful error messages
- Exposed list[3.6.2/4.1], enum[3.6.11/4.1], and term list[3.6.20/4.1] items in their respective functions' scope
- The `at` methods on strings[3.5.28/4.4], arrays[3.5.2/3.4], dictionaries[3.5.12/3.2], and content[3.5.8/3.3] now support specifying a default value
- Added support for passing a function to `replace`[3.5.28/4.18] that is called with each match.
- Fixed replacement[3.5.28/4.18] strings: They are now inserted completely verbatim instead of supporting the previous (unintended) magic dollar syntax for capture groups
- Fixed bug with trailing placeholders in destructuring patterns
- Fixed bug with underscore in parameter destructuring
- Fixed crash with nested patterns and when hovering over an invalid pattern
- Better error messages when casting to an integer[3.5.17] or float[3.5.15] fails

## Text and Layout

- Implemented sophisticated CJK punctuation adjustment
- Disabled overhang[3.7.12/2.13] for CJK punctuation
- Added basic translations for Traditional Chinese
- Fixed alignment[3.7.6/4.4] of text inside raw blocks (centering a raw block, e.g. through a figure, will now keep the text itself left-aligned)

- Added support for passing a array instead of a function to configure table cell alignment[3.6.19/3.7] and fill[3.6.19/3.8] per column
- Fixed automatic figure `kind`[3.6.7/5.6] detection
- Made alignment of enum numbers[3.6.11/3.9] configurable, defaulting to `end`
- Figures can now be made breakable with a show-set rule for blocks in figure
- Initial fix for smart quotes in RTL languages

## Export

- Fixed ligatures in PDF export: They are now copyable and searchable
- Exported PDFs now embed ICC profiles for images that have them
- Fixed export of strokes with zero thickness

## Web app

- Projects can now contain folders
- Added upload by drag-and-drop into the file panel
- Files from the file panel can now be dragged into the editor to insert them into a Typst file
- You can now copy-paste images and other files from your computer directly into the editor
- Added a button to resend confirmation email
- Added an option to invert preview colors in dark mode
- Added tips to the loading screen and the Help menu. Feel free to propose more!
- Added syntax highlighting for YAML files
- Allowed middle mouse button click on many buttons to navigate into a new tab
- Allowed more project names
- Fixed overridden Vim mode keybindings
- Fixed many bugs regarding file upload and more

## Miscellaneous Improvements

- Improved performance of counters, state, and queries
- Improved incremental parsing for more efficient recompilations
- Added support for `.yaml` extension in addition to `.yml` for bibliographies
- The CLI now emits escape codes only if the output is a TTY
- For users of the `typst` crate: The `Document` is now `Sync` again and the `World` doesn't have to be `'static` anymore

# Typst 0.3.0 (April 26, 2023)

## Breaking changes

- Renamed a few symbols: What was previous `dot.op` is now just `dot` and the basic dot is `dot.basic`. The same applies to `ast` and `tilde`.
- Renamed `mod` to `rem`[3.5.9/2.37] to more accurately reflect the behavior. It will remain available as `mod` until the next update as a grace period.
- A lone underscore is not a valid identifier anymore, it can now only be used in patterns
- Removed `before` and `after` arguments from `query`[3.12.6]. This is now handled through flexible selectors[3.5.26] combinator methods
- Added support for attachments[3.8.10/3.1] (sub-, superscripts) that precede the base symbol. The `top` and `bottom` arguments have been renamed to `t` and `b`.

## New features

- Added support for more complex strokes[3.11.12] (configurable caps, joins, and dash patterns)
- Added `cancel`[3.8.12] function for equations
- Added support for destructuring[3.3.3] in argument lists and assignments
- Added `alt`[3.11.7/3.5] text argument to image function
- Added `toml`[3.13.6] function for loading data from a TOML file
- Added `zip`[3.5.2/3.17], `sum`[3.5.2/3.19], and `product`[3.5.2/3.20] methods for arrays
- Added `fact`, `perm`, `binom`, `gcd`, `lcm`, `atan2`, `quo`, `trunc`, and `fract` calculation[3.5.9] functions

## Improvements

- Text in SVGs now displays properly
- Typst now generates a PDF heading outline
- References[3.6.17] now provides the referenced element as a field in show rules
- Refined linebreak algorithm for better Chinese justification
- Locations are now a valid kind of selector
- Added a few symbols for algebra
- Added Spanish smart quote support
- Added `selector`[3.5.26] function to turn a selector-like value into a selector on which combinator methods can be called
- Improved some error messages
- The outline and bibliography headings can now be styled with show-set rules
- Operations on numbers now produce an error instead of overflowing

## Bug fixes

- Fixed wrong linebreak before punctuation that follows inline equations, citations, and other elements
- Fixed a bug with argument sinks[3.5.1]
- Fixed strokes with thickness zero
- Fixed hiding and show rules in math
- Fixed alignment in matrices
- Fixed some alignment bugs in equations
- Fixed grid cell alignment
- Fixed alignment of list marker and enum markers in presence of global alignment settings
- Fixed path closing
- Fixed compiler crash with figure references
- A single trailing line breaks is now ignored in math, just like in text

## Command line interface

- Font path and compilation root can now be set with the environment variables `TYPST_FONT_PATHS` and `TYPST_ROOT`
- The output of `typst fonts` now includes the embedded fonts

## Development

- Added instrumentation for debugging and optimization
- Added `--update` flag and `UPDATE_EXPECT` environment variable to update reference images for tests
- You can now run a specific subtest with `--subtest`
- Tests now run on multiple threads

# Typst 0.2.0 (April 11, 2023)

## Breaking changes

- Removed support for iterating over index and value in for loops[3.3.5]. This is now handled via unpacking and enumerating. Same goes for the `map`[3.5.2/3.15] method.
- Dictionaries[3.5.12] now iterate in insertion order instead of alphabetical order.

## New features

- Added unpacking syntax[3.3.3] for let bindings, which allows things like `let (1, 2) = array`
- Added `enumerate`[3.5.2/3.16] method
- Added `path` function for drawing Bézier paths
- Added `layout`[3.10.12] function to access the size of the surrounding page or container
- Added `key` parameter to `sorted`[3.5.2/3.30] method

## Command line interface

- Fixed `--open` flag blocking the program
- New Computer Modern font is now embedded into the binary
- Shell completions and man pages can now be generated by setting the `GEN_ARTIFACTS` environment variable to a target directory and then building Typst

## Miscellaneous improvements

- Fixed page numbering in outline
- Added basic i18n for a few more languages (AR, NB, CS, NN, PL, SL, ES, UA, VI)
- Added a few numbering patterns (Ihora, Chinese)
- Added `sinc` operator[3.8.26]
- Fixed bug where math could not be hidden with `hide`[3.10.11]
- Fixed sizing issues with box, block, and shapes
- Fixed some translations
- Fixed inversion of "R" in `cal`[3.8.27/1.6] and `frak`[3.8.27/1.3] styles
- Fixed some styling issues in math
- Fixed supplements of references to headings
- Fixed syntax highlighting of identifiers in certain scenarios
- Ratios[3.10.20] can now be multiplied with more types and be converted to floats[3.5.15] with the `float`[3.5.15] function

# Typst 0.1.0 (April 04, 2023)

## Breaking changes

- When using the CLI, you now have to use subcommands:
  - `typst compile file.typ` or `typst c file.typ` to create a PDF
  - `typst watch file.typ` or `typst w file.typ` to compile and watch
  - `typst fonts` to list all fonts
- Manual counters now start at zero. Read the "How to step" section here[3.12.1] for more details
- The bibliography styles[3.6.1/4.4] `"author-date"` and `"author-title"` were renamed to `"chicago-author-date"` and `"chicago-author-title"`

## Figure improvements

- Figures now automatically detect their content and adapt their behavior. Figures containing tables, for instance, are automatically prefixed with "Table X" and have a separate counter
- The figure's supplement (e.g. "Figure" or "Table") can now be customized
- In addition, figures can now be completely customized because the show rule gives access to the automatically resolved kind, supplement, and counter

## Bibliography improvements

- The `bibliography`[3.6.1] now also accepts multiple bibliography paths (as an array)
- Parsing of BibLaTeX files is now more permissive (accepts non-numeric edition, pages, volumes, dates, and Jabref-style comments; fixed abbreviation parsing)
- Labels and references can now include `:` and `.` except at the end
- Fixed APA bibliography ordering

## Drawing additions

- Added `polygon`[3.11.9] function for drawing polygons
- Added support for clipping in boxes[3.10.5/2.9] and blocks[3.10.4/2.12]

## Command line interface

- Now returns with non-zero status code if there is an error
- Now watches the root directory instead of the current one
- Now puts the PDF file next to input file by default
- Now accepts more kinds of input files (e.g. `/dev/stdin`)
- Added `--open` flag to directly open the PDF

## Miscellaneous improvements

- Added `yaml`[3.13.8] function to load data from YAML files
- Added basic i18n for a few more languages (IT, RU, ZH, FR, PT)
- Added numbering support for Hebrew
- Added support for integers[3.5.17] with base 2, 8, and 16
- Added symbols for double bracket and laplace operator
- The `link`[3.6.10] function now accepts labels[3.5.18]
- The link syntax now allows more characters
- Improved justification of Japanese and Chinese text
- Calculation functions behave more consistently w.r.t to non-real results
- Replaced deprecated angle brackets
- Reduced maximum function call depth from 256 to 64
- Fixed `first-line-indent`[3.6.14/4.6] being not applied when a paragraph starts with styled text
- Fixed extraneous spacing in unary operators in equations

- Fixed block spacing, e.g. in `block(above: 1cm, below: 1cm, ..)`
- Fixed styling of text operators in math
- Fixed invalid parsing of language tag in raw block with a single backtick
- Fixed bugs with displaying counters and state
- Fixed crash related to page counter
- Fixed crash when `symbol`[3.5.29] function was called without arguments
- Fixed crash in bibliography generation
- Fixed access to label of certain content elements
- Fixed line number in error message for CSV parsing
- Fixed invalid autocompletion after certain markup elements

# Changes in early, unversioned Typst

## March 28, 2023

- **Breaking changes:**
  - Enumerations now require a space after their marker, that is, `1.ok` must now be written as `1. ok`
  - Changed default style for term lists[3.6.20]: Does not include a colon anymore and has a bit more indent
- Command line interface
  - Added `--font-path` argument for CLI
  - Embedded default fonts in CLI binary
  - Fixed build of CLI if `git` is not installed
- Miscellaneous improvements
  - Added support for disabling matrix[3.8.21] and vector[3.8.28] delimiters. Generally with `#set math.mat(delim: none)` or one-off with `$mat(delim: #none, 1, 2; 3, 4)$`.
  - Added separator[3.6.20/3.2] argument to term lists
  - Added round[3.8.20/2.7] function for equations
  - Numberings now allow zeros. To reset a counter, you can write `#counter(..).update(0)`
  - Added documentation for `page()` and `position()` methods on `location`[3.12.4] type
  - Added symbols for double, triple, and quadruple dot accent
  - Added smart quotes for Norwegian Bokmål
  - Added Nix flake
  - Fixed bibliography ordering in IEEE style
  - Fixed parsing of decimals in math: `$1.2/3.4$`
  - Fixed parsing of unbalanced delimiters in fractions: `$1/(2 (x)$`
  - Fixed unexpected parsing of numbers as enumerations, e.g. in `1.2`
  - Fixed combination of page fill and header
  - Fixed compiler crash if repeat[3.10.22] is used in page with automatic width
  - Fixed matrices[3.8.21] with explicit delimiter
  - Fixed indent[3.6.20/3.3] property of term lists
  - Numerous documentation fixes
  - Links in bibliographies are now affected by link styling
  - Fixed hovering over comments in web app

## March 21, 2023

- Reference and bibliography management
  - Bibliographies[3.6.1] and citations[3.6.3] (currently supported styles are APA, Chicago Author Date, IEEE, and MLA)
  - You can now reference[3.6.17] sections, figures, formulas, and works from the bibliography with `@label`
  - You can make an element referenceable with a label:
    - `= Introduction <intro>`
    - `$ A = pi r^2 $ <area>`
- Introspection system for interactions between different parts of the document
  - counter[3.12.1] function
    - Access and modify counters for pages, headings, figures, and equations
    - Define and use your own custom counters
    - Time travel: Find out what the counter value was or will be at some other point in the document (e.g. when you're building a list of figures, you can determine the value of the figure counter at any given figure).
    - Counters count in layout order and not in code order
  - state[3.12.7] function
    - Manage arbitrary state across your document
    - Time travel: Find out the value of your state at any position in the document
    - State is modified in layout order and not in code order

  - `query`[3.12.6] function
    - Find all occurrences of an element or a label, either in the whole document or before/after some location
    - Link to elements, find out their position on the pages and access their fields
    - Example use cases: Custom list of figures or page header with current chapter title
  - `locate`[3.12.3] function
    - Determines the location of itself in the final layout
    - Can be accessed to get the `page` and `x`, `y` coordinates
    - Can be used with counters and state to find out their values at that location
    - Can be used with queries to find elements before or after its location
- New `measure`[3.10.14] function
  - Measure the layouted size of elements
  - To be used in combination with the new `style` function that lets you generate different content based on the style context something is inserted into (because that affects the measured size of content)
- Exposed content representation
  - Content is not opaque anymore
  - Content can be compared for equality
  - The tree of content elements can be traversed with code
  - Can be observed in hover tooltips or with `repr`[3.5.25]
  - New methods[3.5.8] on content: `func`, `has`, `at`, and `location`
  - All optional fields on elements are now settable
  - More uniform field names (`heading.title` becomes `heading.body`, `list.items` becomes `list.children`, and a few more changes)
- Further improvements
  - Added `figure`[3.6.7] function
  - Added `numbering`[3.8.16/3.2] parameter on equation function
  - Added `numbering`[3.10.17/4.10] and `number-align`[3.10.17/4.12] parameters on page function
  - The page function's `header`[3.10.17/4.13] and `footer`[3.10.17/4.15] parameters do not take functions anymore. If you want to customize them based on the page number, use the new `numbering`[3.10.17/4.10] parameter or `counter`[3.12.1] function instead.
  - Added `footer-descent`[3.10.17/4.16] and `header-ascent`[3.10.17/4.14] parameters
  - Better default alignment in header and footer
  - Fixed Arabic vowel placement
  - Fixed PDF font embedding issues
  - Renamed `math.formula` to `math.equation`[3.8.16]
  - Font family must be a named argument now: `#set text(font: "..")`
  - Added support for hanging indent[3.6.14/4.7]
  - Renamed paragraph `indent` to `first-line-indent`[3.6.14/4.6]
  - More accurate logarithm[3.5.9/2.19] when base is `2` or `10`
  - Improved some error messages
  - Fixed layout of `terms`[3.6.20] list
- Web app improvements
  - Added template gallery
  - Added buttons to insert headings, equations, raw blocks, and references
  - Jump to the source of something by clicking on it in the preview panel (works for text, equations, images, and more)
  - You can now upload your own fonts and use them in your project
  - Hover debugging and autocompletion now takes multiple files into account and works in show rules
  - Hover tooltips now automatically collapse multiple consecutive equal values
  - The preview now automatically scrolls to the right place when you type
  - Links are now clickable in the preview area
  - Toolbar, preview, and editor can now all be hidden
  - Added autocompletion for raw block language tags
  - Added autocompletion in SVG files

  - New back button instead of four-dots button
  - Lots of bug fixes

## February 25, 2023

- Font changes
  - New default font: Linux Libertine
  - New default font for raw blocks: DejaVu Sans Mono
  - New default font for math: Book weight of New Computer Modern Math
  - Lots of new math fonts available
  - Removed Latin Modern fonts in favor of New Computer Modern family
  - Removed unnecessary smallcaps fonts which are already accessible through the corresponding main font and the `smallcaps`[3.7.7] function
- Improved default spacing for headings
- Added `panic`[3.5.21] function
- Added `clusters`[3.5.28/4.6] and `codepoints`[3.5.28/4.7] methods for strings
- Support for multiple authors in `set document`[3.6.5/3.4]
- Fixed crash when string is accessed at a position that is not a char boundary
- Fixed semicolon parsing in `#var ;`
- Fixed incremental parsing when inserting backslash at end of `#"abc"`
- Fixed names of a few font families (including Noto Sans Symbols and New Computer Modern families)
- Fixed autocompletion for font families
- Improved incremental compilation for user-defined functions

## February 15, 2023

- Box[3.10.5] and block[3.10.4] have gained `fill`, `stroke`, `radius`, and `inset` properties
- Blocks may now be explicitly sized, fixed-height blocks can still break across pages
- Blocks can now be configured to be `breakable`[3.10.4/2.3] or not
- Numbering style[3.6.11/3.2] can now be configured for nested enums
- Markers[3.6.2/3.2] can now be configured for nested lists
- The `eval`[3.5.14] function now expects code instead of markup and returns an arbitrary value. Markup can still be evaluated by surrounding the string with brackets.
- PDFs generated by Typst now contain XMP metadata
- Link boxes are now disabled in PDF output
- Tables don't produce small empty cells before a pagebreak anymore
- Fixed raw block highlighting bug

## February 12, 2023

- Shapes, images, and transformations (move/rotate/scale/repeat) are now block-level. To integrate them into a paragraph, use a `box`[3.10.5] as with other elements.
- A colon is now required in an "everything" show rule: Write `show: it => ..` instead of `show it => ...` This prevents intermediate states that ruin your whole document.
- Non-math content like a shape or table in a math formula is now centered vertically
- Support for widow and orphan prevention within containers
- Support for RTL[3.7.12/2.19] in lists, grids, and tables
- Support for explicit `auto` sizing for boxes and shapes
- Support for fractional (i.e. `1fr`) widths for boxes
- Fixed bug where columns jump to next page
- Fixed bug where list items have no leading
- Fixed relative sizing in lists, squares and grid auto columns
- Fixed relative displacement in `place`[3.10.19] function
- Fixed that lines don't have a size
- Fixed bug where `set document(..)` complains about being after content
- Fixed parsing of `not in` operation

- Fixed hover tooltips in math
- Fixed bug where a heading show rule may not contain a pagebreak when an outline is present
- Added `baseline`[3.10.5/2.3] property on `box`[3.10.5]
- Added `tg`[3.8.26] and `ctg`[3.8.26] operators in math
- Added delimiter setting for `cases`[3.8.13] function
- Parentheses are now included when accepting a function autocompletion

## February 2, 2023

- Merged text and math symbols, renamed a few symbols (including `infty` to `infinity` with the alias `oo`)
- Fixed missing italic mappings
- Math italics correction is now applied properly
- Parentheses now scale in `$zeta(x/2)$`
- Fixed placement of large root index
- Fixed spacing in `$abs(-x)$`
- Fixed inconsistency between text and identifiers in math
- Accents are now ignored when positioning superscripts
- Fixed vertical alignment in matrices
- Fixed `text` set rule in `raw` show rule
- Heading and list markers now parse consistently
- Allow arbitrary math directly in content

## January 30, 2023

Go to the announcement blog post.[A]

- New expression syntax in markup/math
  - Blocks cannot be directly embedded in markup anymore
  - Like other expressions, they now require a leading hash
  - More expressions available with hash, including literals (`#"string"`) as well as field access and method call without space: `#emoji.face`
- New import syntax
  - `#import "module.typ"` creates binding named `module`
  - `#import "module.typ": a, b` or `#import "module.typ": *` to import items
  - `#import emoji: face, turtle` to import from already bound module
- New symbol handling
  - Removed symbol notation
  - Symbols are now in modules: `sym`, `emoji`, and `math`
  - Math module also reexports all of `sym`
  - Modified through field access, still order-independent
  - Unknown modifiers are not allowed anymore
  - Support for custom symbol definitions with `symbol` function
  - Symbols now listed in documentation
- New `math` module
  - Contains all math-related functions
  - Variables and function calls directly in math (without hash) access this module instead of the global scope, but can also access local variables
  - Can be explicitly used in code, e.g. `#set math.vec(delim: "[")`
- Delimiter matching in math
  - Any opening delimiters matches any closing one
  - When matched, they automatically scale
  - To prevent scaling, escape them
  - To forcibly match two delimiters, use `lr` function
  - Line breaks may occur between matched delimiters
  - Delimiters may also be unbalanced

[A] https://typst.app/blog/2023/january-update

  - You can also use the `lr` function to scale the brackets (or just one bracket) to a specific size manually
- Multi-line math with alignment
  - The `\` character inserts a line break
  - The `&` character defines an alignment point
  - Alignment points also work for underbraces, vectors, cases, and matrices
  - Multiple alignment points are supported
- More capable math function calls
  - Function calls directly in math can now take code expressions with hash
  - They can now also take named arguments
  - Within math function calls, semicolons turn preceding arguments to arrays to support matrices: `$mat(1, 2; 3, 4)$`
- Arbitrary content in math
  - Text, images, and other arbitrary content can now be embedded in math
  - Math now also supports font fallback to support e.g. CJK and emoji
- More math features
  - New text operators: `op` function, `lim`, `max`, etc.
  - New matrix function: `mat`
  - New n-ary roots with `root` function: `$root(3, x)$`
  - New under- and overbraces, -brackets, and -lines
  - New `abs` and `norm` functions
  - New shorthands: `[|`, `|]`, and `||`
  - New `attach` function, overridable attachments with `script` and `limit`
  - Manual spacing in math, with `h`, `thin`, `med`, `thick` and `quad`
  - Symbols and other content may now be used like a function, e.g. `$zeta(x)$`
  - Added Fira Math font, removed Noto Sans Math font
  - Support for alternative math fonts through `#show math.formula: set text("Fira Math")`
- More library improvements
  - New `calc` module, `abs`, `min`, `max`, `even`, `odd` and `mod` moved there
  - New `message` argument on `assert` function
  - The `pairs` method on dictionaries now returns an array of length-2 arrays instead of taking a closure
  - The method call `dict.at("key")` now always fails if `"key"` doesn't exist Previously, it was allowed in assignments. Alternatives are `dict.key = x` and `dict.insert("key", x)`.
- Smarter editor functionality
  - Autocompletion for local variables
  - Autocompletion for methods available on a value
  - Autocompletion for symbols and modules
  - Autocompletion for imports
  - Hover over an identifier to see its value(s)
- Further editor improvements
  - New Font menu with previews
  - Single projects may now be shared with share links
  - New dashboard experience if projects are shared with you
  - Keyboard Shortcuts are now listed in the menus and there are more of them
  - New Offline indicator
  - Tooltips for all buttons
  - Improved account protection
  - Moved Status indicator into the error list button
- Further fixes
  - Multiple bug fixes for incremental parser
  - Fixed closure parameter capturing
  - Fixed tons of math bugs
  - Bugfixes for performance, file management, editing reliability
  - Added redirection to the page originally navigated to after signin