文本

以各种方式自定义文本的外观和布局。

该函数经常被使用，无论是 set 规则还是直接使用。虽然 set 规则通常是更简单的选择，但是直接调用 text 函数在将文本作为参数传递给另一个函数时非常有用。

例

```
#set text(18pt)
With a set rule.

#emph(text(blue)[
  With a function call.
])
```

With a set rule.

With a function call.

参数

```
text(
  font: str array,
  fallback: bool,
  style: str,
  weight: int str,
  stretch: ratio,
  size: length,
  fill: color gradient pattern,
  stroke: none length color gradient stroke pattern 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,
  kerning: bool,
  alternates: bool,
  stylistic-set: none int,
  ligatures: bool,
  discretionary-ligatures: bool,
  historical-ligatures: bool,
  number-type: auto str,
  number-width: auto str,
  slashed-zero: bool,
  fractions: bool,
  features: array dictionary,
  content,
  str,
) -> content
```

字体

字体系列名称或字体系列名称的优先级列表。

处理文本时，Typst 会按顺序尝试所有指定的字体系列，直到找到具有必要字形的字体为止。在下面的示例中，该字体是首选，但由于它不包含阿拉伯文字形，因此改用阿拉伯文本。Inria SerifNoto Sans Arabic

可用字体的集合因平台而异：

在 Web 应用程序中，您可以通过单击“Ag”按钮查看可用字体列表。您可以通过将文件上传或上传到项目中来提供其他字体。它们将被自动发现。.t-tf.otf

在本地，Typst 使用您安装的系统字体。此外，还可以使用参数或环境变量来添加应扫描字体的目录。–font-pathTYPST_FONT_PATHS

默认："linux libertine"

样式

所需的字体样式。

当请求斜体样式并且只有斜体样式可用时，将使用它。同样，反过来，斜体样式可以代替斜体样式。当斜体样式和斜体样式都不可用时，Typst 会选择普通样式。由于大多数字体仅提供斜体或斜体样式，因此很少观察到斜体和斜体样式之间的区别。

如果你想强调你的文本，你应该使用 emph 函数来代替。如果您改变主意如何表示重点，这样以后就很容易调整样式。

```
text(
  font: str array,
  fallback: bool,
  style: str,
  weight: int str,
  stretch: ratio,
  size: length,
  fill: color gradient pattern,
  stroke: none length color gradient stroke pattern 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,
  kerning: bool,
  alternates: bool,
  stylistic-set: none int,
  ligatures: bool,
  discretionary-ligatures: bool,
  historical-ligatures: bool,
  number-type: auto str,
  number-width: auto str,
  slashed-zero: bool,
  fractions: bool,
  features: array dictionary,
  content,
  str,
) -> content
```

默认："normal"

重量

字体字形的所需粗细。接受预定义权重名称和/或其中一个之间的整数。当所需的粗细不可用时，Typst 会从粗细最接近的字体系列中选择字体。

100900

如果你想强烈强调你的文本，你应该使用 strong 函数来代替。如果您改变主意如何表示强烈的强调，这样以后就很容易调整风格。

变体	详
"thin"	重量薄（100）。
"extralight"	超轻量级（200）。
"light"	重量轻（300）。
"regular"	常规重量（400）。
"medium"	中等重量（500）。
"semibold"	半粗体重量（600）。
"bold"	粗体重量（700）。
"extrabold"	超粗重量（800）。
"black"	黑色重量（900）。

默认："regular"

伸展

字形的所需宽度。接受 和 之间的比率。当所需的宽度不可用时，Typst 会从拉伸最接近的族中选择字体。只有当字体的压缩或扩展版本可用时，这才会拉伸文本。50%200%

如果要调整字符之间的间距量而不是拉伸字形本身，请改用 tracking 属性。

默认：100%

尺寸

所需的字体大小。该值构成 em 基础单位：1em 等于字体大小。

你也可以直接给出字体大小，单位为 em。当然，这是相对于前面指定的字体大小。

默认：11pt

填充

字体填充颜色

默认：luma(0%)

间距
字符之间的间距。

默认：0pt

间距
单词之间的间距。

可以给出绝对长度，也可以相对于字体中空格字符的宽度。

如果想调整字符之间的间距而不是单词之间的间距，请改用 tracking 属性。

默认：100%

CJK 字符和拉丁字符之间间距
是否在 CJK 字符和拉丁字符之间自动插入间距。

默认：auto

基准线距离
文本基准线移动的距离。

默认：0pt

悬挂边距
是否允许某些字形悬挂到边距中。这可以使对齐更加美观。

默认：true

上边缘
文本布局和定位中使用的文本周围概念框的顶端。这会影响包含文本的容器的大小。

变体	详
"ascender"	字体的升序，通常超过所有字形的高度。
"cap-height"	大写字母的近似高度。
"x-height"	非升序小写字母的近似高度。
"baseline"	字母所在的基线。
"bounds"	字形边界框的上边缘。

默认："cap-height"

下边缘

文本布局和定位中使用的文本周围概念框的底部。这会影响包含文本的容器的大小。

变体	详
"baseline"	字母所在的基线。
"descender"	字体的降序，通常超过所有字形的深度。
"bounds"	字形边界框的下边缘。

默认："baseline"

语言

一个 ISO 639-1/2/3 语言代码。

设置正确的语言会影响 Typst 的各个部分：

- 文本处理功能可以做出更明智的选择。
- 连字符会匹配相应的语言模式。
- 智能引号会变成响应语言的引号。
- 以及其他所有能感知语言的功能。

默认："en"

区域

一个 ISO 3166-1 alpha-2 地区代码。

这使文本处理功能能够做出更明智的选择。

默认：none

脚本

OpenType 编写脚本。

lang 和 script 的组合决定了字体功能（如字形替换）的实现方式。通常，该值是修改后的（全小写）ISO 15924 脚本标识符，数学书写脚本用于适用于数学符号的特征。

当设置为 auto（默认设置和建议设置）时，将为共享公共 Unicode 脚本属性的每个字符块选择适当的脚本。

默认：auto

方向

文本和内联对象的主导方向。可能的取值有：

- auto：从 lang 属性自动推断方向。
- ltr：从左到右布局文本。
- rtl：从右到左布局文本。

当使用从右到左的语言（如阿拉伯语或希伯来语）书写时，应设置 text language 或 direction。虽然各个文本行会自动按正确的方向布局，但设置主要方向可为双向重新排序算法提供必要的信息，以正确放置标点符号和内联对象。此外，设置方向会影响对齐值 start 和 end，在 ltr 文本中等同于 left 和 right，在 rtl 文本中则相反。

如果设置为 rtl 遇到错误或某种方式产生不佳的输出，请通过 contact table 或我们的 Discord server 与我们联系！

默认：auto

连字符

是否使用连字符改善断行。当设定为 auto 时，只有在启用对齐时才会连字符化文本。

设置文本语言可确保使用正确的连字符化模式。

默认：auto

字距

是否应用字距调整。

启用后，特定的字母配对会相互靠近或远离，以获得更美观的结果。下面的示例演示了如何通过减小 “T” 和 “o” 之间的间距来获得更自然的外观。将其设置为 false 会通过关闭 OpenType kern 字体特性来禁用字距调整。

默认：true

替代

是否应用样式替代。

有时一些字体会包含相同代码点的替代字形。将其设置为 true 会通过启用 OpenType salt 字体特性来切换到这些字形。

默认：false

样式集

应用哪个样式集。字体设计师可以将替代字形形式分类为样式集。由于该值高度依赖于字体，因此你需要查阅字体以了解可用的集合。当设置为介于 1 和 20 之间的整数时，会启用相应的 OpenType 字体特性 ss01、…、ss20。

默认：none

连字

是否启用标准连字。

某些字母组合（例如“fi"）通常显示为单个合并字形，称为 连字。将其设置为 false 会通过关闭 OpenType liga 和 clig 字体特性来禁用这些连字。

默认：true

自由连字

是否启用应当谨慎使用的连字。将其设置为 true 会通过启用 OpenType dlig 字体特性来启用这些连字。

默认：false

历史连字

是否启用历史连字。将其设置为 true 会通过启用 OpenType hlig 字体特性来启用这些连字。

默认：false

数字字体

选择哪种数字字体。当设置为 auto 时，使用字体的默认数字。

- “lining”

适合大写文本的数字（OpenType lnum 字体功能）。

- “old-style”

非常适合大写和小写文本流的数字（OpenType onum 字体功能）。

默认：auto

数字宽度

数字字符的宽度。当设置为 auto 时，使用字体的默认数字。

- “proportional”

具有特定于字形宽度的数字（OpenType pnum 字体功能）。

- “tabular”

等宽的数字（OpenType tnum 字体功能）。

默认：auto

带斜线的零字形

是否使用带斜线的零字形。将其设置为 true 会通过启用 OpenType zero 字体特性来使用这些字形。

默认：false

分数

是否将数字转换为分数。将其设置为 true 会通过启用 OpenType frac 字体特性来使用这些字形。

不建议全局启用此属性，因为它会破坏斜杠后所有数字的外观（例如，在 URL 中）。相反，当你想要分数时，请在本地启用它。

默认：false

特色

要使用的原始 OpenType 功能。

- 如果给定字符串数组，则将字符串标识的功能设置为 1。
- 如果给定映射到数字的字典，则将键标识的功能设置为值。

默认：(:)

内容

要应用样式的内容，根据其他参数进行设定。

文本

这文本。