功能

从参数值到返回值的映射。

您可以通过在函数名称后面直接写入用括号括起来的以逗号分隔的函数*参数*列表来调用函数。此外，您可以在正常参数列表*之后*将任意数量的尾随内容块参数传递给函数。如果普通参数列表变空，则可以省略。Typst 支持位置参数和命名参数。前者通过位置和类型来标识，而后者则写为`name: value`。

在数学模式下，函数调用具有特殊行为。有关更多详细信息，请参阅数学文档。

例子

```
// 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

函数是 Typst 的基本构建块。 Typst 提供了各种排版任务的功能。此外，您编写的标记由函数支持，并且所有样式都通过函数发生。此参考列出了所有可用的功能以及如何使用它们。另请参阅有关设置和显示规则的文档，以了解在 Typst 中使用函数的其他方法。

元素功能

有些功能与标题或表格等*元素*相关联。当被调用时，它们会创建各自类型的元素。与普通函数相比，它们可以进一步用于设置规则、显示规则和选择器。

功能范围

函数可以在自己的范围内保存相关的定义，类似于module。这方面的例子有`assert.eq`或`list.item`。不过，此功能目前仅适用于内置函数。

定义函数

您可以使用let 绑定定义您自己的函数，该绑定在绑定名称后面有一个参数列表。参数列表可以包含强制位置参数、具有默认值的命名参数和参数接收器。

函数绑定的右侧是函数体，它可以是块或任何其他表达式。它定义函数的返回值并且可以依赖于参数。如果函数体是代码块，则返回值是将块中每个表达式的值连接起来的结果。

在函数体内，return关键字可用于提前退出并可选择指定返回值。如果未给出显式返回值，则主体的计算结果为连接 之前的所有表达式的结果return。

```
#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
]
```

未命名函数

=>您还可以通过指定参数列表和函数体来创建未命名函数，而无需创建绑定。如果您的函数只有一个参数，则参数列表两边的括号是可选的。未命名函数主要用于显示规则，但也适用于采用页面函数footer属性等函数的可设置属性。

```
#show "once?": it => [#it #it]
once?
```

once? once?

关于函数纯度的注意事项

在 Typst 中，所有函数都是*纯函数*。这意味着对于相同的参数，它们总是返回相同的结果。当第二次调用它们时，它们无法"记住"事物以产生另一个值。

唯一的例外是内置方法，例如array.push(value).这些可以修改它们所调用的值。

定义

with

返回一个预先应用了给定参数的新函数。

```
自己。和（..任何）->功能
```

arguments 任何 *必需的* *位置性* *可变参数*

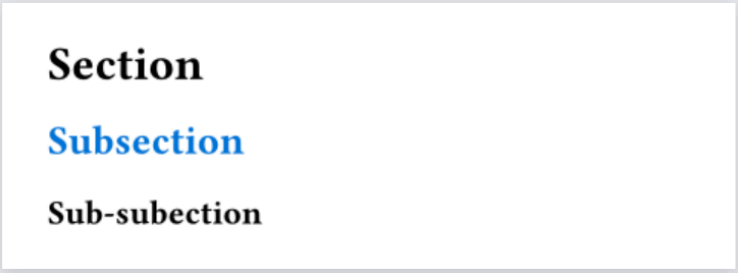
应用于函数的参数。

where

返回一个选择器，用于过滤属于该函数的元素，其字段具有给定参数的值。

```
自己。在哪里（..任何）->选择器
```

```
#show heading.where(level: 2): set text(blue)
= Section
== Subsection
=== Sub-subection
```



fields 任何 *必需的* *位置性* *可变参数*

要过滤的字段。