

Typst workshop

王梓宁 Zining Wang  
first@wznmickey.com  
2024-10-06

介绍

Typst 是一个 Markup 语言，目前是用 Rust 写的实现，支持多种模式，包括**标记模式**、**数学模式**和**代码模式**。它的设计目标是提供一种更加直观、易用的文档编辑方式，同时提供 L<sup>A</sup>T<sub>E</sub>X 的强大功能。

使用 Typst 的方式

文件名后缀为 .typ，可以通过以下方式使用 Typst：

- 1. 在 Typst 的官网上使用在线编辑器 <https://typst.app/>
- 2. 使用 Typst 的 VSCode 插件，在 vscode 中编辑

一些资源

文档

- <https://typst-doc-cn.github.io/docs/packages/>
- <https://typst.app/docs/>

包和模板

- <https://typst.app/universe/>
- [https://github.com/wznmickey/JJ\\_Lab\\_Report\\_typst\\_template](https://github.com/wznmickey/JJ_Lab_Report_typst_template)


三种模式 参考 <https://typst.app/docs/reference/syntax/>

• 标记模式 Markup Mode (类似 Markdown)

- 默认模式
- 数学模式 Math Mode (类似 L<sup>A</sup>T<sub>E</sub>X)
  - 在 \$ 与 \$ 之间输入，同时支持行间公式和行内公式
- 代码模式 Code Mode (类似 一般编程代码)
  - 使用 # 开头。
  - 所有的标记模式里面的格式实际上都是代码模式里的部分函数的语法糖，可以通过函数方式调用以与其他代码协同

标记模式

- 标题

**Example**

```
1 = 我是标题1
2 == 我是标题2
```

typst

**我是标题 1**  
**我是标题 2**

- 列表

**Example**

```
1 1. 有序列表
2 1. 有序列表
3 2. 有序列表
4 1. 有序列表
5 2. 有序列表
6 - 无序列表
7 - 无序列表
8 - 无序列表
9 - 无序列表
10 - 无序列表
11 + 有序列表
12 + 有序列表
13 + 有序列表
14
```

typst

1. 有序列表

1. 有序列表

2. 有序列表

1. 有序列表

2. 有序列表

• 无序列表

- 无序列表

- 无序列表

• 无序列表


- 无序列表

1. 有序列表

2. 有序列表

1. 有序列表

- 强调

**Example**

```
1 *强调* _emphasis_
```

typst

**强调 *emphasis***

数学模式

行内公式


**Example**

```
1 这是一个行内公式 $a^2 + b^2 = c^2$ 这是一个行内公式
```

typst

**这是一个行内公式  $a^2 + b^2 = c^2$  这是一个行内公式**

行间公式


**Example**

```
1 这是一个行间公式 $ a^2 + b^2 = c^2 $ 这是一个行间公式
```

typst

**这是一个行间公式**  
$$a^2 + b^2 = c^2$$
  
**这是一个行间公式**

各类符号

**Example**

```
$alpha dot beta dots gamma sum_{-15}^{infinity} 3<=10
overline{1 + 2 + ... + 5} , a eq 3, b eq.not 3, a in
NN*+ sqrt(123) sin(x) = sin(2 pi), 1/2, 1/((2+2^e +
4/c)^(3^delta)) integral arrow.r.double.bar Delta sect
\u(2229) ns
$ alpha dot beta dots gamma sum_{-15}^{infinity} 3<=10
2 $ overline{1 + 2 + ... + 5} $
3 a eq 3, b eq.not 3, a in NN*+ sqrt(123) sin(x) = sin(2
pi), 1/2 $
$ 1/(2+2^e + 4/c)^(3^delta)) integral arrow.r.double.bar
Delta sect \u(2229) n $
```

typst

$$\alpha \cdot \beta \dots \gamma \sum_{i=1}^{\infty} 3 \leq 101 + 2 + \dots + 5, a = 3, b \neq 3, a \in \mathbb{N}^* \sqrt{123} \sin(x) = \sin(2\pi), \frac{1}{2}, \frac{1}{((2+2^e + \frac{4}{c})^{(3^\delta)})^\gamma} \int \Rightarrow \Delta \cap \cap \cap$$
$$\alpha \cdot \beta \dots \gamma \sum_{i=1}^{\infty} 3 \leq 101 + 2 + \dots + 5$$
$$a = 3, b \neq 3, a \in \mathbb{N}^* \sqrt{123} \sin(x) = \sin(2\pi), \frac{1}{2}$$
$$\frac{1}{(2 + 2^e + \frac{4}{c})^\gamma} \int \Rightarrow \Delta \cap \cap \cap$$

分段函数、矩阵等多行情况

**Example**

```
1 $ f(x, y) := cases(
2 1 "if" (x dot y)/2 <= 0,
3 2 "if" x "is even",
4 3 "if" x in NN,
5 4 "else",
6 ) mat(
7 1, 2, ..., 10;
8 2, 2, ..., 10;
9 dots.v, dots.v, dots.down, dots.v;
10 10, 10, ..., 10;
11 ) $
12
```

typst


$$f(x,y) := \begin{cases} 1 \text{ if } \frac{x \cdot y}{2} \leq 0 \\ 2 \text{ if } x \text{ is even} \\ 3 \text{ if } x \in \mathbb{N} \\ 4 \text{ else} \end{cases} \begin{pmatrix} 1 & 2 & \dots & 10 \\ 2 & 2 & \dots & 10 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 10 & \dots & 10 \end{pmatrix}$$

代码模式 语法（与 rust 语法不同，请不要混淆）

- 以 # 开头（如果需要多行命令，使用大括号包起来）。部分内容需要一定的编程基础。

部分常用的 L<sup>A</sup>T<sub>E</sub>X 和 markdown 功能在 typst 使用方法


- 1. 代码：原生自带，使用 raw 函数或 导入，与 markdown 类似 <https://typst.app/docs/reference/text/raw/>

**Example**

```
1 `a = 5`
```

typst

**a = 5**


**Example**

```
1 ```python
2 a = 5
3 b = 10
4 ```
```

typst

```
1 a = 5
2 b = 10
```


python

**Example**

```
1 #raw("int x = 5;")
```

typst

**int x = 5;**

**Example**


```
1 #raw("int x = 5;",block: true, lang: "c++")
```

typst

```
1 int x = 5;
```

C++

- 2. bibliography：原生自带，使用 bibliography 函数导入，需要引用时使用@（也可以引用如图片之类的内部材料）。<https://typst.app/docs/reference/model/bibliography/>

**Example**

```
1 @madje2022programmable
```

typst

**[1]**

**Example**

```
1 #bibliography("a.bib")
```

typst

**Bibliography**  
[1] L. Madje, "A Programmable Markup Language for Typesetting," 2022.

- 3. 图片：原生自带，使用 image 函数导入 <https://typst.app/docs/reference/visualize/image/>

**Example**

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

typst



- 4. 表格：原生自带，使用 table 函数导入

**Example**

```
1 #table(columns:3){Name}[ID][Score][Zining Wang]
520370910042}[A+]
```

typst

Name	ID	Score
Zining Wang	520370910042	A+

- 5. beamer：使用第三方包（比如 touying）


- 6. 引用：原生自带，使用 quote 函数导入

**Example**

```
1 #quote()[Hello, world!]
```

typst

**"Hello, world!"**

**Example**

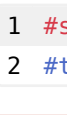
```
1 #quote(block:true)[Hello, world!]
```

typst

**Hello, world!**

代码编写

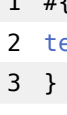
- 以 # 开头（如果需要多行命令，使用大括号包起来）。

**Example**

```
1 #text("123")
```

typst

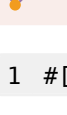
**123**

**Example**

```
1 #set text(size: 1.5em)
2 #text("123")
```

typst

**123**

**Example**

```
1 #(set text(size: 1.5em)
2 text("123")
3 )
```

typst

**123**

- 如果临时要使用标记模式的文本材料，使用 {} 包起来。

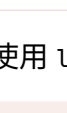
**Example**

```
1 #[= Hello]
```

typst

**Hello**

- 表达式（函数等的返回值）会以默认文本格式显示

**Example**


```
1 "hello".len()
```

typst

**5**

```
1 print(len("hello"))
python
printf("%d",strlen("hello"));//严格来说这一直有点小问题，因为 strlen 返回的是 size_t 类型，但是 printf 的格式化符号是 %d，应该是 %zu
C
1 disp(strlength("hello"));
matlab
```


- 使用 let 赋值与（解）绑定

**Example**

```
1 #{
2 let x = 1
3 x
4 }
```

**1**

```
1 x = 1
2 print(x)
python
1 int x = 1;
2 printf("%d",x);
C
1 x = 1;
2 disp(x);
matlab
```

**Example**

```
1 #{
2 let myFun(x,y) = {x+y}
3 myFun(1,2)
4 }
```

**3**

```
1 def myFun(x,y):
2 return x + y
3 print(myFun(1, 2))
python
1 int myFun(int x, int y) {
2 return x + y;
3 }
4 printf("%d", myFun(1, 2));
C
1 function z = myFun(x, y)
2 z = x + y;
3 end
4 disp(myFun(1, 2));
matlab
```

- 函数调用的一些语法糖

**Example**

```
1 #{
2 let alert1(body, fill: red) = {
3 rect
4 fill: fill,
5 ["Warning:\ #body*],
6 }
7 }
8 let alert2(fill: red, body) = {
9 rect
10 fill: fill,
11 ["Warning:\ #body*],
12 }
13 }
14 alert1[
15 Danger is imminent!
16 ]
17 }
18 alert1(fill: blue)[
19 KEEP OFF TRACKS
20 ]
21 }
22 alert1("KEEP OFF TRACKS",fill: blue)
23 alert2("KEEP OFF TRACKS",fill: blue)
24 }
25 }
```

typst

Warning: Danger is imminent!

Warning: KEEP OFF TRACKS

Warning: KEEP OFF TRACKS

Warning: KEEP OFF TRACKS

- 有 if, while, for 是 in-range 设计。

**Example**

```
1 #{
2 let array = {1, 2, 3}
3 for i in array {
4 if i >= 2 {#i}
5 }
6 }
```

**23**

```
1 array = {1, 2, 3}
2 for i in array:
3 if i >= 2:
4 print(i)
python
1 int array[] = {1, 2, 3};
2 for (int i = 0; i < 3; i++) {
3 if (array[i] >= 2) {
4 printf("%d", array[i]);
5 }
6 }
C
1 array = [1, 2, 3];
2 for i = array
3 if i >= 2
4 disp(i);
matlab
5 end
6 end
```

- set 和 show 修改文档的样式属性 <https://typst.app/docs/reference/styling>
- set 设置属性


**Example**

```
1 #set text(size: 1em,style:"italic")
2 交^大
3 #set text(size: 1.5em)
4 密院
5 交^大
6 密院
7 #set text(style:"normal")
8 上海
9 闵行
10 #highlight{SJTU}
11 #highlight{Shanghai}
12 #set highlight(fill:blue)
13 #highlight{China}
14 #highlight{Asia}
```

typst

交大  
密院  
交大  
密院  
上海  
闵行  
SJTU Shanghai China Asia

- show 修改文档的默认设置

**Example**


```
1 #{
2 set heading(numbering: "I.I")
3 show heading: it => [
4 #set align(center)
5 \- #emph(it.body)
6 #counter(heading).display(it.numbering) \-
7 ]
8 [
9 == Typst
10 == A
11 == B
12
13 == Markdown
14
15 == D
16 ]
17 }
```

typst

~ Typst I ~  
~ A I.I ~  
~ B I.II ~  
~ Markdown II ~  
~ D II.II ~

代码编写 Advanced

- 类型
- 字符串 String""
- 数组(Dict) Array()
- 字典(Dict (key:value))
- 使用. 访问字段和方法

**Example**

```
1 #let it = [== Typst]
2 #it.depth
3 #let dict = (name: "JI")
4 #dict.at("name")
5 #dict.values()
```

**2 JI ("JI",)**

- 使用 let 解包数组

**Example**

```
1 #{
2 let (a,...,b) = {1,2,3,4,5}
3 b
4 }
```

**5**

Lab 模板

[https://github.com/wznmickey/JJ\\_Lab\\_Report\\_typst\\_template](https://github.com/wznmickey/JJ_Lab_Report_typst_template)