



Typst template for Shandong University

Continuously Improving...

孙更欣

计算机科学与技术学院

2025-06-20



Knowledge Infinite • Spirit Vast
学无止境 · 气有浩然

Outline

- 🔥 介绍
- 🔥 安装
- 🔥 快速入门
- 🔥 部分基础功能展示
- 🔥 小组件
- 🔥 页面

1. 介绍

什么是 Typst?

► 介绍:

- **Typst** 是为写作而诞生的基于标记的排版系统。**Typst** 的目标是成为功能强大的排版工具，并且让用户可以愉快地使用它。

什么是 Typst?

► 介绍:

- **Typst** 是为写作而诞生的基于标记的排版系统。**Typst** 的目标是成为功能强大的排版工具，并且让用户可以愉快地使用它。

► 简单来说:

- **Typst** = L^AT_EX 的排版能力 + **Markdown** 的简洁语法 + 强大且现代的脚本语言

什么是 Typst?

▶ 介绍:

- **Typst** 是为写作而诞生的基于标记的排版系统。**Typst** 的目标是成为功能强大的排版工具，并且让用户可以愉快地使用它。

▶ 简单来说:

- **Typst** = L^AT_EX 的排版能力 + **Markdown** 的简洁语法 + 强大且现代的脚本语言

▶ 运行环境: Web Wasm / CLI / LSP Language Server

▶ 编辑器: Web App / VS Code / Neovim / Emacs

```
1 #set page(width: 10cm, height: auto)
2 #set heading(numbering: "1.")
3
4 = Fibonacci sequence
5 The Fibonacci sequence is defined through the recurrence relation
6 $F_n = F_{(n-1)} + F_{(n-2)}$.
7 It can also be expressed in closed form:
8
9 $ F_n = \text{round}(1 / \text{sqrt}(5) \cdot \text{phi.alt}^n), \text{quad } \text{phi.alt} = (1 + \text{sqrt}(5)) / 2 $
10
11 #let count = 8
12 #let nums = range(1, count + 1)
13 #let fib(n) = (
14   if n <= 2 { 1 }
15   else { fib(n - 1) + fib(n - 2) }
16 )
17 The first #count numbers of the sequence are:
18
19 #align(center, table(
20   columns: count,
21   ..nums.map(n => $F_{#n}$),
22   ..nums.map(n => str(fib(n))),
23 ))
```

来源: Typst 官方 Repo [🔗](#)

1. Fibonacci sequence

The Fibonacci sequence is defined through the recurrence relation $F_n = F_{n-1} + F_{n-2}$. It can also be expressed in *closed form*:

$$F_n = \left\lfloor \frac{1}{\sqrt{5}} \phi^n \right\rfloor, \quad \phi = \frac{1 + \sqrt{5}}{2}$$

The first 8 numbers of the sequence are:

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	1	2	3	5	8	13	21

- ▶ **语法简洁：**上手难度跟 **Markdown** 相当，文本源码可阅读性高。

Typst 优势

- ▶ **语法简洁：**上手难度跟 **Markdown** 相当，文本源码可阅读性高。
- ▶ **编译速度快：**
 - Typst 使用 Rust 语言编写，即 `typ(esetting+rust)`。
 - 增量编译时间一般维持在**数毫秒**到**数十毫秒**。

Typst 优势

- ▶ **语法简洁：**上手难度跟 **Markdown** 相当，文本源码可阅读性高。
- ▶ **编译速度快：**
 - Typst 使用 Rust 语言编写，即 `typ(esetting+ru)st`。
 - 增量编译时间一般维持在**数毫秒**到**数十毫秒**。
- ▶ **环境搭建简单：**不像 L^AT_EX 安装起来困难重重，**Typst** 原生支持中日韩等非拉丁语言，官方 Web App 和本地 VS Code 均能**开箱即用**。

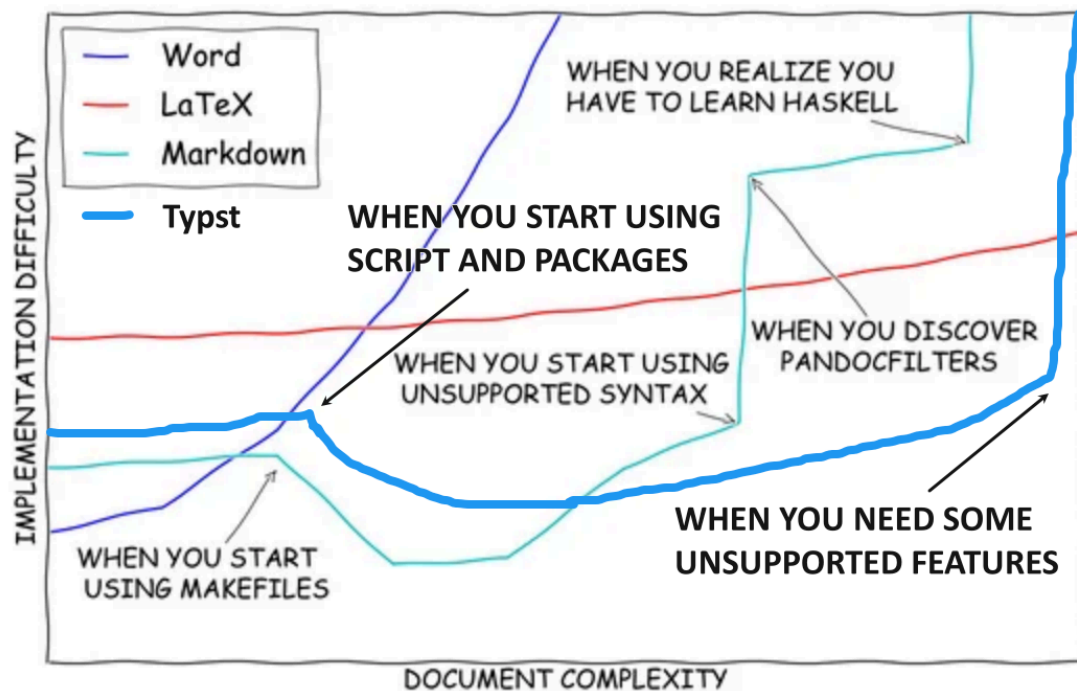
Typst 优势

- ▶ **语法简洁：**上手难度跟 **Markdown** 相当，文本源码可阅读性高。
- ▶ **编译速度快：**
 - Typst 使用 Rust 语言编写，即 `typ(etting+ru)st`。
 - 增量编译时间一般维持在**数毫秒**到**数十毫秒**。
- ▶ **环境搭建简单：**不像 L^AT_EX 安装起来困难重重，**Typst** 原生支持中日韩等非拉丁语言，官方 Web App 和本地 VS Code 均能**开箱即用**。
- ▶ **现代脚本语言：**
 - 变量、函数、闭包与错误检查 + 函数式编程的纯函数理念
 - 可嵌套的 [标记模式]、{ 脚本模式 } 与 \$ 数学模式 \$ 不就是 ~~JSX~~ 嘛
 - 统一的包管理，支持导入 WASM 插件和按需自动安装第三方包

Typst 对比其他排版系统

排版系统	安装难度	语法难度	编译速度	排版能力	模板能力	编程能力	方言数量
LaTeX	难 选项多 + 体积大 + 流程复杂	难 语法繁琐 + 嵌套多 + 难调试	慢 宏语言编译速度极慢	强 拥有最多的历史积累	强 拥有众多的模板和开发者	中等 图灵完备但只是宏语言	中等 众多格式、引擎和发行版
Markdown	易 大多编辑器默认支持	易 入门语法十分简单	快 语法简单编译速度较快	弱 基于 HTML 排版能力弱	中等 语法简单易于更换模板	弱 图灵不完备需要外部脚本	多 方言众多且难以统一
Word	易 默认已安装	易 所见即所得	中等 能实时编辑大文件会卡顿	强 大公司开发通用排版软件	弱 二进制格式难以自动化	弱 编程能力极弱	少 统一的标准和文件格式
Typst	易 安装简单开箱即用	中等 入门语法简单进阶使用略难	快 增量编译渲染速度最快	较强 已满足日常排版需求	强 制作和使用模板都较简单	强 图灵完备现代编程语言	少 统一的语法统一的编译器


Typst 对比其他排版系统




From reddit r/LaTeX [🔗](#) and modified by OrangeX4

2. 安裝

- ▶ 官方提供了 Web App, 可以直接在浏览器中使用 

- ▶ 官方提供了 Web App, 可以直接在浏览器中使用 
- ▶ 优点:
 - 即开即用, 无需安装。
 - 类似于 L^AT_EX 的 Overleaf, 可以直接编辑、编译、分享文档。
 - 拥有「**多人协作**」支持, 可以实时共同编辑。

- ▶ 官方提供了 Web App, 可以直接在浏览器中使用 
- ▶ **优点:**
 - 即开即用, 无需安装。
 - 类似于 L^AT_EX 的 Overleaf, 可以直接编辑、编译、分享文档。
 - 拥有「**多人协作**」支持, 可以实时共同编辑。
- ▶ **缺点:**
 - 中文字体较少, 经常需要手动上传字体文件, 但有上传大小限制。
 - 缺少版本控制, 目前无法与 GitHub 等代码托管平台对接。

本地使用（推荐）

► VS Code 方案（推荐）

- 在插件市场安装「Tinymist Typst」插件。
- 新建一个 `.typ` 文件，然后按下 **Ctrl** + **K** **V** 即可实时预览。
- **不再需要其他配置**，例如我们并不需要命令行安装 Typst CLI。

本地使用（推荐）

► VS Code 方案（推荐）

- 在插件市场安装「Tinymist Typst」插件。
- 新建一个 .typ 文件，然后按下 **Ctrl** + **K** **V** 即可实时预览。
- **不再需要其他配置**，例如我们并不需要命令行安装 Typst CLI。

► Neovim / Emacs 方案

- 可以配置相应的 LSP 插件和 Preview 插件。

本地使用（推荐）

► VS Code 方案（推荐）

- 在插件市场安装「Tinymist Typst」插件。
- 新建一个 .typ 文件，然后按下 **Ctrl** + **K** **V** 即可实时预览。
- **不再需要其他配置**，例如我们并不需要命令行安装 Typst CLI。

► Neovim / Emacs 方案

- 可以配置相应的 LSP 插件和 Preview 插件。

► CLI 方案： `typst compile --root <DIR> <INPUT_FILE>`

- Windows: `winget install --id Typst.Typst`
- macOS: `brew install typst`
- Linux: 查看 Typst on Repology [↗](#)

3. 快速入门

Hello World

```
1 #set page(width: 20em, height: auto)
2 #show heading.where(level: 1): set align(center)
3
4 #show "Typst": set text(fill: blue, weight: "bold")
5 #show "LaTeX": set text(fill: red, weight: "bold")
6 #show "Markdown": set text(fill: purple, weight: "bold")
7
8 = Typst 讲座
9
10 Typst 是为 *学术写作* 而生的基于 标记 的排版系统。
11
12 Typst = LaTeX 的排版能力 + Markdown 的简洁语法 + 现代脚本语言
13
14 #underline[本讲座]包括以下内容:
15
16 + 快速入门 Typst
17 + Typst 编写各类模板
18   - 笔记、论文、简历和 Slides
19 + Typst 高级特性
20   - 脚本、样式和包管理
21 + Typst 周边生态开发体验
22   - Pinit、MiTeX、Touying 和 VS Code 插件
23
24 ```py
25 print('Hello Typst!')
26 ```
```

Typst

Typst 讲座

Typst 是为 **学术写作** 而生的基于标记的排版系统。

Typst = **LaTeX** 的排版能力 + **Markdown** 的简洁语法 + 现代的脚本语言


本讲座包括以下内容：

1. 快速入门 **Typst**
2. **Typst** 编写各类模板
 - 笔记、论文、简历和 Slides
3. **Typst** 高级特性
 - 脚本、样式和包管理
4. **Typst** 周边生态开发体验
 - Pinit、MiTeX、Touying 和 VS Code 插件

```
print('Hello Typst!')
```

标记只是语法糖

1 = 一级标题

 Typst

2

3 = 二级标题

4

5 简单的段落，可以*加粗*和_强调_。

6

7 - 无序列表

8 + 有序列表

9 / 术语：术语列表


10

11 ```py

12 print('Hello Typst!')

13 ```

1 #heading(level: 1, [一级标题])

 Typst

2

3 #heading(level: 2, [二级标题])

4

5 简单的段落，可以#strong[加粗]和#emph[强调]。

6

7 #list.item[无序列表]

8 #enum.item[有序列表]

9 #terms.item[术语][术语列表]

10

11 #raw(lang: "py", block: true, "print('Hello Typst!')")

► Simplicity through Consistency

- 类似 **Markdown** 的特殊标记语法，实现「内容与格式分离」。
- = 一级标题 只是 `#heading[一级标题]` 的**语法糖**。

► Simplicity through Consistency

- 类似 **Markdown** 的特殊标记语法，实现「内容与格式分离」。
- `= 一级标题` 只是 `#heading[一级标题]` 的**语法糖**。

► 标记模式和脚本模式

- 标记模式下，使用井号 `#` 进入脚本模式，如 `#strong[加粗]`。
 - 脚本模式下不需要额外井号，例如 `#heading(strong([加粗]))`
 - 大段脚本代码可以使用花括号 `{}`，例如 `#{1 + 1}`。

► Simplicity through Consistency

- 类似 **Markdown** 的特殊标记语法，实现「**内容与格式分离**」。
- `=` 一级标题 只是 `#heading[一级标题]` 的**语法糖**。

► 标记模式和脚本模式

- 标记模式下，使用井号 `#` 进入脚本模式，如 `#strong[加粗]`。
 - 脚本模式下不需要额外井号，例如 `#heading(strong([加粗]))`
 - 大段脚本代码可以使用花括号 `{}`，例如 `#{1 + 1}`。
- 脚本模式下，使用方括号 `[]` 进入标记模式，称为**内容块**。
 - **Typst** 是强类型语言，有常见的数据类型，如 `int` 和 `str`。
 - 内容块 `[]` 类型 `content` 是 **Typst** 的核心类型，可嵌套使用。
 - `#fn(..)[XXX][YYY]` 是 `#fn(.., [XXX], [YYY])` 的**语法糖**。

- ▶ Set 规则可以设置样式, 即「为函数设置参数默认值」的能力。
 - 例如 `#set heading(numbering: "1.")` 用于设置标题的编号。
 - 使得 `#heading[标题]` 变为 `#heading(numbering: "1.", [标题])`。

Set/Show Rules

- ▶ Set 规则可以设置样式，即「为函数设置参数默认值」的能力。
 - 例如 `#set heading(numbering: "1.")` 用于设置标题的编号。
 - 使得 `#heading[标题]` 变为 `#heading(numbering: "1.", [标题])`。
- ▶ Show 规则用于全局替换，即在语法树上进行「宏编程」的能力。
 - 例如 `#show "LaTeX": "Typst"` 将单词 LaTeX 替换为 Typst。
 - 例如让一级标题居中，可以用「箭头函数」：

```
1 #show heading.where(level: 1): body => {  
2   set align(center)  
3   body  
4 }
```

 Typst

- ▶ 化简为 `#show heading.where(level: 1): set align(center)`

数学公式

- ▶ x 是行内公式,
$$x^2 + y^2 = 1$$
 是行间公式。


数学公式

- ▶ x 是行内公式, $x^2 + y^2 = 1$ 是行间公式。
- ▶ 与 L^AT_EX 的差异:
 - $(x + 1) / x \geq 1 \Rightarrow 1/x \geq 0$
 - `\frac{x + 1}{x} \geq 1 \Rightarrow \frac{1}{x} \geq 0`

数学公式

- ▶ x 是行内公式, $x^2 + y^2 = 1$ 是行间公式。
- ▶ 与 L^AT_EX 的差异:
 - $(x + 1) / x \geq 1 \Rightarrow 1/x \geq 0$
 - `\frac{x + 1}{x} \geq 1 \Rightarrow \frac{1}{x} \geq 0`
- ▶ 报告, 我想用 LaTeX 语法: [🔗](#)

```
1 #import "@preview/mitex:0.2.2": *
2
3 Write inline equations like #mi("x") or #mi[y].
4 #mitex(`
5   \frac{x + 1}{x} \geq 1 \Rightarrow \frac{1}{x} \geq 0
6 `)
```

 Typst

Touying 对比其他 Slides 方案

方案	语法难度	编译速度	排版能力	模板能力	编程能力	动画效果	代码公式
PowerPoint	易 所见即所得	快 实时编辑	强 大公司开发 通用软件	强 模板数量最多 容易制作模板	弱 编程能力极弱 难以显示进度	强 动画效果多 但用起来复杂	难 难以插入代码和公式 贴 图片
Beamer	难 语法繁琐 + 嵌套多 + 难 调试	慢 宏语言编译 速度极慢	弱 使用模板后 排版难以修改	中等 拥有较多模板 开发模板较难	中等 图灵完备 但只是宏语言	中等 简单动画方便 无过渡动画	易 基本默认支持
Markdown	易 入门语法十分简单	快 语法简单 编译速度较快	弱 语法限制 排版能力弱	弱 难以制作模板 只有内置模板	弱 图灵不完备 需要外部脚本	中等 动画效果全看提供了什 么	易 基本默认支持
Touying	易 语法简单 使用方便	快 增量编译渲染 速度最快	中等 满足日常学术 Slides 需求	强 制作和使用 模板都较简单	强 图灵完备 现代编程语言	中等 简单动画方便 无过渡动画	易 默认支持 MiTeX 包

4. 部分基础功能展示

注：这里也可以有内容

我是基于 touying 制作的非官方 Typst 模板，用于展示山东大学的学术成果。

展示超长标题

想分列显示?

展示 `void fn`

Second column.第二列



标题

内容

- ▶ 无序列表
- ▶ 第二项

1. 有序列表 1
2. 有序列表 2

	Exam 1	Exam 2	Exam 3
John		A	
Mary		A	A
Robert	B	A	B

行内公式: $a^2 + b^2 = c^2$ 块级公式:

$$E = mc^2$$

$$\langle a, b \rangle = \vec{a} \cdot \vec{b}$$

$$= a_1 b_1 + a_2 b_2 + \dots a_n b_n$$

$$= \sum_{i=1}^n a_i b_i.$$

Definition 4.8.1: A natural number is called a *prime number* if it is greater than 1 and cannot be written as the product of two smaller natural numbers.

Example: The numbers 2, 3, and 17 are prime. Corollary 4.8.1.1 shows that this list is not exhaustive!

Theorem 4.8.1 (Euclid): There are infinitely many primes.

Proof: Suppose to the contrary that p_1, p_2, \dots, p_n is a finite enumeration of all primes. Set $P = p_1 p_2 \dots p_n$. Since $P + 1$ is not in our list, it cannot be

prime. Thus, some prime factor p_j divides $P + 1$. Since p_j also divides P , it must divide the difference $(P + 1) - P = 1$, a contradiction. \square

Corollary 4.8.1.1: There is no largest prime number.

Corollary 4.8.1.2: There are infinitely many composite numbers.

Theorem 4.8.2: There are arbitrarily long stretches of composite numbers.

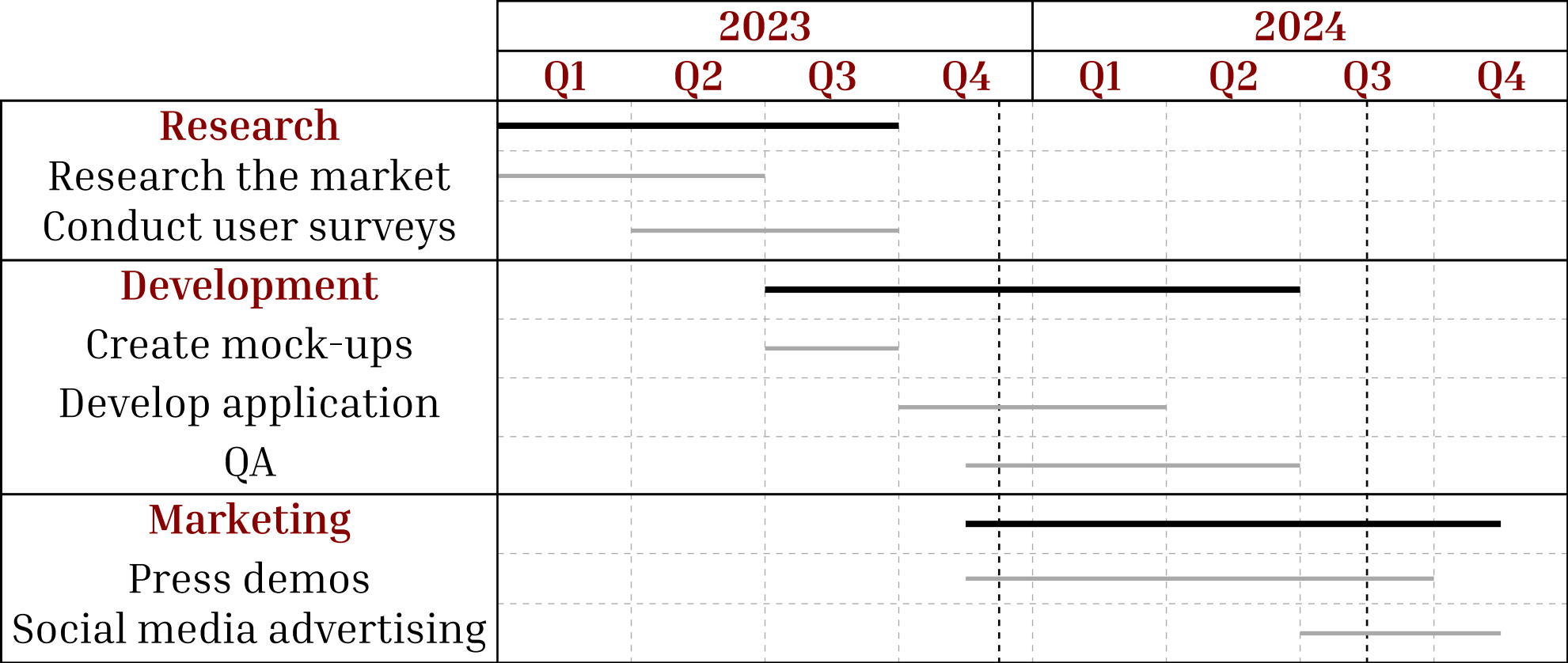
Proof: For any $n > 2$, consider

$$n! + 2, \quad n! + 3, \quad \dots, \quad n! + n$$

\square

5. 小组件

时间轴，很简单



Conference demo

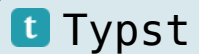
App store launch

Dec 2023


Aug 2024

代码块，很优雅

```
1 pub fn main() {  
2     println!("Hello, world!");  
3 }
```



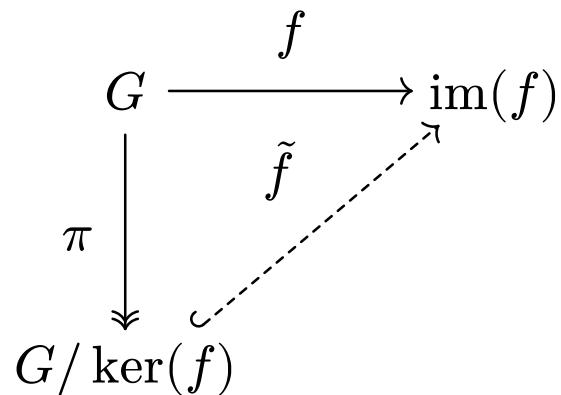
```
1 pub fn main() {  
2     println!("Hello, world!");  
3 }
```



用节点和箭头绘制图表

```
1 #diagram(cell-size: 15mm, $
2   G edge(f, ->) edge("d", pi, ->>) & im(f) \
3   G slash ker(f) edge("ur", tilde(f), "hook-->")
4 $)
```

Typst



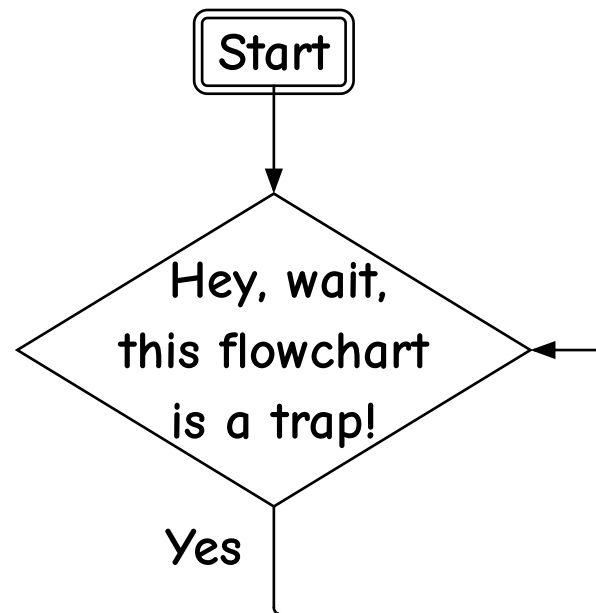
1

<https://typst.app/docs>

用节点和箭头绘制图表

```
1  #import fletcher.shapes: diamond
2  #set text(font: "Comic Neue", weight: 600)
3
4  #diagram(
5    node-stroke: 1pt,
6    edge-stroke: 1pt,
7    node((0,0), [Start], corner-radius: 2pt, extrude: (0, 3)),
8    edge("-|>"),
9    node((0,1), align(center)[
10      Hey, wait,\ this flowchart\ is a trap!
11    ], shape: diamond),
12    edge("d,r,u,l", "-|>", [Yes], label-pos: 0.1)
13 )
```

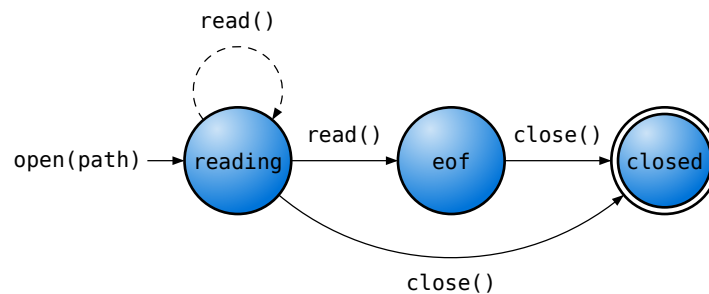
Typst



用节点和箭头绘制图表

```
1 #set text(10pt)
2 #diagram(
3   node-stroke: .1em,
4   node-fill: gradient.radial(blue.lighten(80%), blue, center: (30%,
5     20%), radius: 80%),
6   spacing: 4em,
7   edge((-1,0), "r", "-|>", `open(path)`, label-pos: 0, label-side:
8     center),
9   node((0,0), `reading`, radius: 2em),
10  edge(`read()`, "-|>"),
11  node((1,0), `eof`, radius: 2em),
12  edge(`close()`, "-|>"),
13  node((2,0), `closed`, radius: 2em, extrude: (-2.5, 0)),
14  edge((0,0), (0,0), `read()`, "- -|>", bend: 130deg),
15  edge((0,0), (2,0), `close()`, "-|>", bend: -40deg),
16 )
```

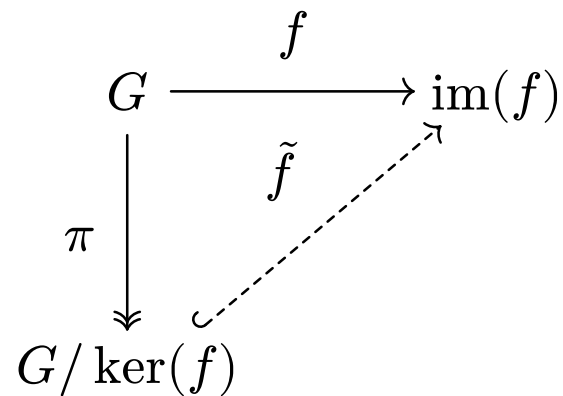
Typst



用节点和箭头绘制图表

```
1 #diagram(cell-size: 15mm, $
2   G edge(f, ->) edge("d", pi, ->>) & im(f) \
3   G slash ker(f) edge("ur", tilde(f), "hook-->")
4 $)
```

Typst



展示框，很有趣

```
1 #showybox(  
2   [Hello world!]  
3 )
```

Typst

```
1 showybox(  
2   frame: (  
3     dash: "dashed",  
4     border-color: red.darken(40%)  
5   ),  
6   body-style: (  
7     align: center  
8   ),  
9   sep: (  
10    dash: "dashed"  
11  ),  
12  shadow: (  
13    offset: (x: 2pt, y: 3pt),  
14    color: yellow.lighten(70%)  
15  ),  
16  [This is an important message!],  
17  [Be careful outside. There are dangerous bananas!]  
18 )
```


Typst

Hello world!

This is an important message!

Be careful outside. There are
dangerous bananas!

```
1 #info[ This is the info clue ... ]  
2 #tip(title: "Best tip ever")[Check out this cool package]
```

 Typst

Info

This is the info clue ...

Best tip ever

Check out this cool package

Info

This is information

Success

I' m making a note here: huge success

Warning

First warning...

? Question

Question?



Example

An example make things interesting

6. 页面

聚焦页

left

middle

right

top

bottom

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

引用如下（此处为空）。

致谢

1. [Typst 官方文档](#) 
2. [Typst 中文教程](#) 
3. [Typst 非官方中文交流群](#) 793548390
4. [OrangeX4 的 Typst 讲座](#)  (几乎占了介绍内容的全部)
5. [Touying 官方文档](#) 
6. [中科大 touying-pres-ustc 模板](#) 
7. [touying-brandred-uobristol](#) 

THANKS FOR ALL

敬请指正！