

# 并不复杂的 Typst 讲座

Typst is Simple

OrangeX4

南京大学

2024 年 03 月 17 日

# 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。
- 简单来说:
  - Typst = L<sup>A</sup>T<sub>E</sub>X 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



# Typst 介绍与展示

## 什么是 Typst?

Typst 是一门用于文档排版的标记语言。通过 Typst，你可以用简洁语法编写出美观的文档。

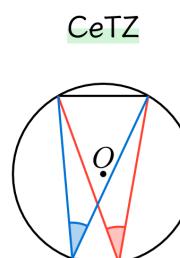
如果你使用过 Markdown 或者 L<sup>A</sup>T<sub>E</sub>X，你应该很熟悉“从带标记的文本生成文档”的流程。

## Typst 的优势

- 公式排版、参考文献管理等基本功能
- 语法简洁，容易上手
- 现代的、增量编译的编程语言可以
  - 快速地生成文档，并实时预览
  - 有更好的代码提示和报错信息
  - 更方便地编程与编写模板
- 环境搭建简单
- 有包管理器，不需要像 TeX Live 那样在本地安装大量用不到的包
- 得益于友好的编程语言以及 Wasm 插件支持，Typst 已经有了许多功能强大的包

部分包展示

```
#set heading(numbering: "一、") // 样式设置
= 兰亭集序
永和九年，岁在癸丑，暮春之初，会于会稽山阴之兰亭，修禊事也。 // 标记语法
#{ // 编程语言
  set text(size: 0.8em)
  let t(name, s, offset, n) = table(
    columns: (3em, ) + (auto, ) * n,
    inset: 0.3em, align: center,
    strong(name),
    ..s.clusters(),
    [*年份*],
    ..range(n).map(i =>
      str(calc.rem(i + offset, n)))
  )
  t("天干", "甲乙丙丁戊己庚辛壬癸", 4, 10)
  t("地支", "子丑寅卯辰巳午未申酉戌亥", 4, 12)
}
$ cases(x equiv 3 (mod 10),
         x equiv 5 (mod 12) )
  => x equiv 53 (mod 60) $
$ 353 - floor(353/60) = 53 $
```



Pinit  
后之览者，亦将有  
感于斯文。  
这

## 效果展示

### 一、兰亭集序

永和九年，岁在癸丑，暮春之初，会于会稽山阴之兰亭，修禊事也。

天干	甲	乙	丙	丁	戊	己	庚	辛	壬	癸
年份	4	5	6	7	8	9	0	1	2	3

地支	子	丑	寅	卯	辰	巳	午	未	申	酉	戌	亥
年份	4	5	6	7	8	9	10	11	0	1	2	3

$$\begin{cases} x \equiv 3 \pmod{10} \\ x \equiv 5 \pmod{12} \end{cases} \Rightarrow x \equiv 53 \pmod{60}$$

$$353 - \left\lfloor \frac{353}{60} \right\rfloor = 53$$

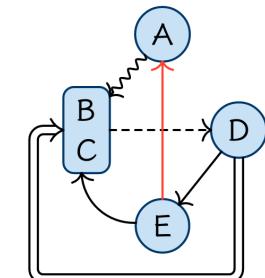
### Showybox

通知

近日将有一股强冷空气来袭，请注意保暖。

秋季天干物燥，要时刻注意消防安全。

### Fletcher



This work by Wallbreaker5th is marked with CC0 1.0.

To view a copy of this license, visit <http://creativecommons.org/publicdomain/zero/1.0>

# Typst 速览

```
#set page(width: 10cm, height: auto)
#set heading(numbering: "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 = round(1 / sqrt(5) phi.alt^n), quad phi.alt = (1 +
sqrt(5)) / 2 $
```

```
#let count = 8
#let nums = range(1, count + 1)
#let fib(n) =
  if n <= 2 { 1 }
  else { fib(n - 1) + fib(n - 2) }
)
```

The first `#count` numbers of the sequence are:

```
#align(center, table(
  columns: count,
  ..nums.map(n => $F_#n$),
  ..nums.map(n => str(fib(n))),
))
```

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

# Typst 优势

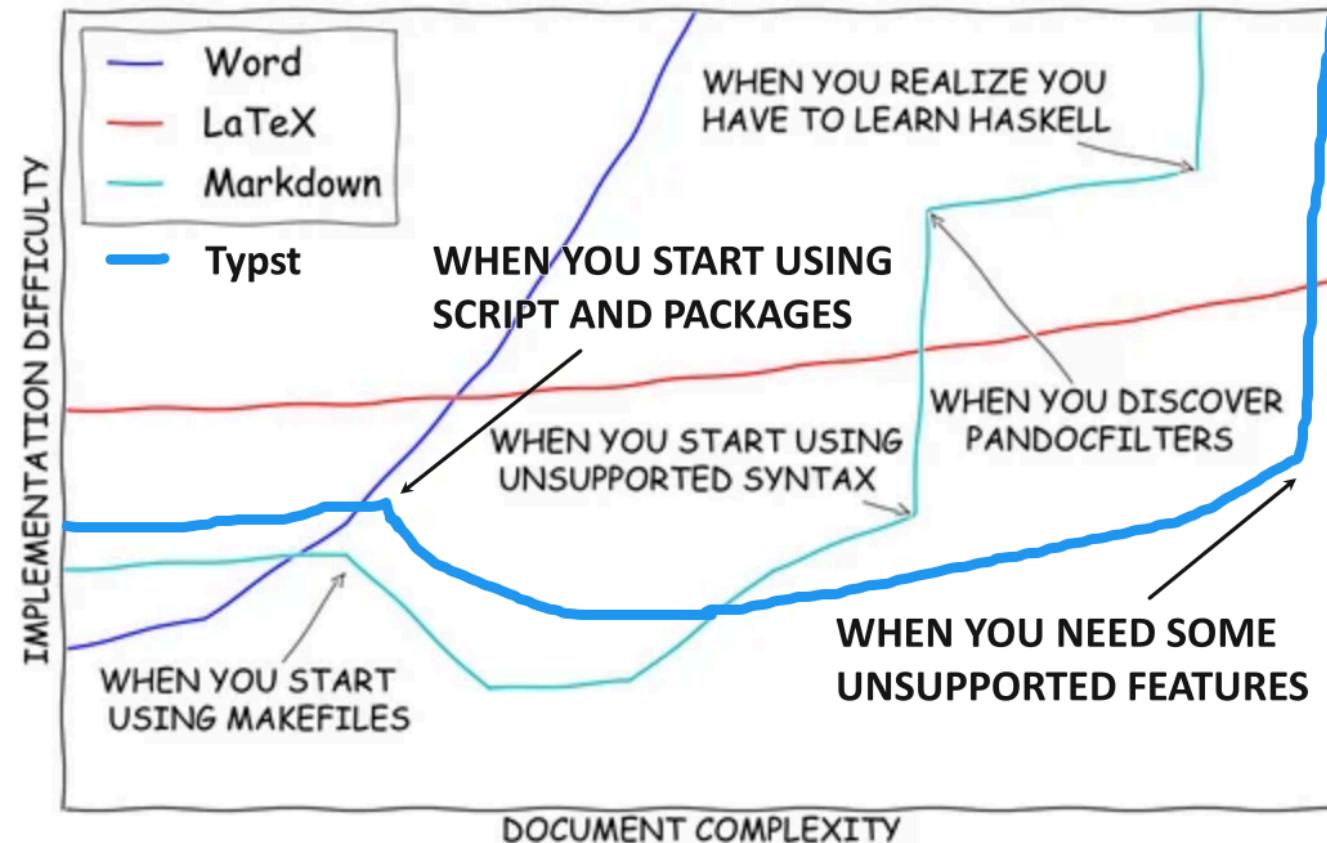
- **语法简洁：**上手难度跟 **Markdown** 相当，文本源码可阅读性高。
- **编译速度快：**
  - Typst 使用 Rust 语言编写，即 `typ(esetting+ru)st`。
  - 增量编译时间一般维持在数毫秒到数十毫秒。
- **环境搭建简单：**不像 LATEX 安装起来困难重重，**Typst** 原生支持中日韩等非拉丁语言，官方 Web App 和本地 VS Code 均能开箱即用。
- **现代脚本语言：**
  - 变量、函数、闭包与错误检查 + 函数式编程的纯函数理念
  - 可嵌套的 `[标记模式]`、`{脚本模式}` 与 `$ 数学模式 $` 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
<b>LaTeX</b>	<b>难</b> 选项多 + 体积大 + 流程复杂	<b>难</b> 语法繁琐 + 嵌套多 + 难调试	<b>慢</b> 宏语言编译速度极慢	<b>强</b> 拥有最多的历史积累	<b>强</b> 拥有众多的模板和开发者	<b>中等</b> 图灵完备但只是宏语言	<b>中等</b> 众多格式、引擎和发行版
<b>Markdown</b>	<b>易</b> 大多编辑器默认支持	<b>易</b> 入门语法十分简单	<b>快</b> 语法简单编译速度较快	<b>弱</b> 基于 HTML 排版能力弱	<b>中等</b> 语法简单易于更换模板	<b>弱</b> 图灵不完备需要外部脚本	<b>多</b> 方言众多且难以统一
<b>Word</b>	<b>易</b> 默认已安装	<b>易</b> 所见即所得	<b>中等</b> 能实时编辑大文件会卡顿	<b>强</b> 大公司开发通用排版软件	<b>弱</b> 二进制格式难以自动化	<b>弱</b> 编程能力极弱	<b>少</b> 统一的标准和文件格式
<b>Typst</b>	<b>易</b> 安装简单开箱即用	<b>中等</b> 入门语法简单进阶使用略难	<b>快</b> 增量编译渲染速度最快	<b>较强</b> 已满足日常排版需求	<b>强</b> 制作和使用模板都较简单	<b>强</b> 图灵完备现代编程语言	<b>少</b> 统一的语法统一的编译器

# Typst 对比其他排版系统

介绍



From reddit r/LaTeX ↯ and modified by OrangeX4

# 安装

---

- 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<sup>A</sup>T<sub>E</sub>X 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。

- **VS Code 方案 (推荐)**
  - 在插件市场安装「Tinymist Typst」和「Typst Preview」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 ↗

# 快速入门

---

# Hello World

```
#set page(width: 20em, height: auto)
#show heading.where(level: 1): set align(center)

#show "Typst": set text(fill: blue, weight: "bold")
#show "LaTeX": set text(fill: red, weight: "bold")
#show "Markdown": set text(fill: purple, weight: "bold")
```

## 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

#underline[本讲座]包括以下内容：

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

```
```py
print('Hello 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!')
```

### = 一级标题

```
#heading(level: 1, [一级标题])
```

### == 二级标题

```
#heading(level: 2, [二级标题])
```

简单的段落，可以`*加粗*`和`_强调_`。

简单的段落，可以`#strong`[加粗]和`#emph`[强调]。

- 无序列表
- + 有序列表
- / 术语：术语列表

```
#list.item[无序列表]  
#enum.item[有序列表]  
#terms.item[术语][术语列表]
```

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

```
#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。
- 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])`。
- **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」：
    - `#show heading.where(level: 1): body => {  
 set align(center)  
 body  
}`
    - 化简为 `#show heading.where(level: 1): set align(center)`

- $\$x\$$  是行内公式,  $\$ x^2 + y^2 = 1 \$ <\text{circle}>$  是行间公式。
- 与 LATEX 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: ↗

```
#import "@preview/mitex:0.2.2": *
```

```
Write inline equations like #mi("x") or #mi[y].
```

```
#mitex(`  
  \frac{x + 1}{x} \geq 1 \Rightarrow \frac{1}{x} \geq 0  
)
```

# 制作模板

---

现在，我们想要为一个会议制作一个模板，以下是需求规范：

1. 字体应为 11pt 的衬线字体；
2. 标题应为 17pt 的粗体，居中对齐；
3. 论文包含单栏摘要和两栏正文；
4. 摘要应居中；
5. 正文应两端对齐；
6. 一级章节标题应为 13pt，居中并以小写字母呈现；
7. 二级标题是短标题，斜体，与正文文本具有相同的大小；
8. 最后，页面尺寸应为 US letter，编号在页脚的中心，每页的左上角应包含论文的标题。

```
#let conf(  
    title: none,  
    authors: (),  
    abstract: [],  
    doc,  
) = {  
    // 字体应为 11pt 的衬线字体  
    set text(font: "Linux Libertine", 11pt)  
    // 正文应两端对齐  
    set par(justify: true)  
  
    // 页面尺寸应为 US letter, 编号在页脚的中心,  
    // 每页的左上角应包含论文的标题  
    set page(  
        "us-letter",  
        margin: auto,  
        header: align(  
            right + horizon,  
            title  
)  
        ,  
        numbering: "1",  
)  
}
```

```
// 一级章节标题应为 13pt, 居中并以小写字母呈现  
show heading.where(  
    level: 1  
) : it => block(  
    align(center,  
        text(  
            13pt,  
            weight: "regular",  
            smallcaps(it.body),  
        )  
    ),  
)  
  
// 二级标题是短标题, 斜体, 与正文文本具有相同的大小  
show heading.where(  
    level: 2  
) : it => box(  
    text(  
        11pt,  
        weight: "regular",  
        style: "italic",  
        it.body + [.]  
)
```

# 制作论文模板

制作模板

```
)  
  
// 标题应为 17pt 的粗体，居中对齐  
set align(center)  
text(17pt, title)  
  
// 添加作者列表，最多分为 3 列  
let count = calc.min(authors.len(), 3)  
grid(  
    columns: (1fr,) * count,  
    row-gutter: 24pt,  
    ..authors.map(author => [  
        #author.name \  
        #author.affiliation \  
        #link("mailto:" + author.email)  
    ]),  
)  
  
// 单栏摘要，摘要应居中  
par(justify: false)[  
    *Abstract* \  
]
```

```
#abstract  
]  
  
// 两栏正文  
set align(left)  
columns(2, doc)  
}
```

来源：Typst 官方文档 ↗

# 制作论文模板

```
#show: doc => conf(
  title: [
    Towards Improved Modelling
  ],
  authors: (
    (
      name: "Theresa Tungsten",
      affiliation: "Artos Institute",
      email: "tung@artos.edu",
    ),
    (
      name: "Eugene Deklan",
      affiliation: "Honduras State",
      email: "e.dekhan@hstate.hn",
    ),
  ),
  abstract: lorem(80),
  doc,
)
= Introduction
#lorem(90)

== Motivation
#lorem(140)

== Problem Statement
#lorem(50)

= Related Work
#lorem(200)
```

# 制作模板

Towards Improved Modelling

## Towards Improved Modelling

Theresa Tungsten  
Artos Institute  
tung@artos.edu

Eugene Deklan  
Honduras State  
e.dekhan@hstate.hn

**Abstract**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim auctor 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 distinguere 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 accedit, saluto: 'chaere, inquam, 'Tite! fato, turba omnis chorusque: 'chaere.

**INTRODUCTION**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim auctor 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 distinguere 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 accedit, saluto: 'chaere, inquam, 'Tite! fato, turba omnis chorusque: 'chaere.

**Problem Statement**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim auctor 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 distinguere 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 accedit, saluto: 'chaere, inquam, 'Tite! fato, turba omnis chorusque: 'chaere.

**RELATED WORK**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim auctor 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 distinguere 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 accedit, saluto: 'chaere, inquam, 'Tite! fato, turba omnis chorusque: 'chaere.

1

- Word / HTML 简历模板?
  - 不够美观
- L<sup>A</sup>T<sub>E</sub>X 简历模板?
  - 环境配置复杂
  - 自主定制困难
- Typst 简历模板?
  - 绝对优势领域
  - Chinese-Resume-in-Typst ↩

方橙  
(+86) 155-5555-5555 · 南京大学 · 人工智能 · orange4@qq.com · github.com/orangex4

我是OrangeX4，你也可以叫我一只方橙或方橙。现在是南京大学人工智能学院2020级本科生，正深陷于学习数学、编程和英语的无边苦海中。你问我为什么我的名字那么奇怪？大概是我喜欢吃橘子和橙子；又谐音方程，还有和我的名字谐音的缘故吧。喜欢一切新奇的东西，兴趣十分广泛。

**教育背景**

2023.05 | 南京大学 - 人工智能学院 - 人工智能专业  
2020.09 | GPA: 4.48 / 5 · Rank: 15%

**专业技能**

操作系统 | Linux, Windows  
掌握 | React, JavaScript, Python  
熟悉 | Vue, TypeScript, Node.js  
了解 | Webpack, Java

**获奖情况**

人民奖学金 | 一等奖 · 二等奖 | 2021年11月 - 2022年11月  
人工智能+ | 二等奖 | 2021年11月 - 2022年11月

**项目经历**

**Latex Sympy Calculator** | 个人项目 | 2021年02月 - 2021年04月  
NodeJS, Python, VS Code  
一个用于在 VS Code 中使用 LaTeX 数学公式进行「科学计算」的插件  
➤ 使用 ANTLR 将 LaTeX 语句编译为 SymPy 语句  
➤ 通过 Flask 搭建本地 HTTP 服务器与 VS Code 插件进行通信  
➤ 可以进行多种类型的科学计算，如积分求导、矩阵计算、无穷级数计算等

**黑白棋 Reversi** | 课程项目 | 2021年02月 - 2021年04月  
React, Python, AI  
基于 React 与 Antd 的黑白棋前端，与基于 Python 的黑白棋 AI 后端  
➤ 使用基于评估函数的 BFS 实现了黑白棋 AI，并实现了 Alpha-Beta 剪枝  
➤ 基于 React 搭建了一个黑白棋平台前端，支持玩家对战、人机对战和 AI 对战  
➤ 在后端使用 Flask 及 Socket.io 库，实现了玩家之间的联机对战

**校园经历**

微软学生俱乐部技术部部长 | 2021年09月 - 2022年09月



# 制作简历模板

制作模板

```
#show heading: set text(font:  
"Linux Biolinum")  
// 为链接加入下划线  
#show link: underline  
// 推荐的简历文本大小为“10pt”到“12pt”  
#set text(size: 16pt)  
// 设置页边距  
#set page(margin: (x: 0.9cm, y:  
1.3cm))  
#set par(justify: true)  
// 横线  
#let chiline() = {v(-3pt);  
line(length: 100%); v(-5pt)}  
  
= Alex Chi
```

```
skyzh\@cmu.edu | #link("https://  
github.com/skyzh") [github.com/  
skyzh] | #link("https://skyzh.dev")  
[skyzh.dev]
```

```
== Education  
#chiline()
```

```
#link("https://typst.app/")  
[*#lorem(2)*] #h(1fr) 2333/23 --  
2333/23 \  
#lorem(5) #h(1fr) #lorem(2) \  
- #lorem(10)
```

```
*#lorem(2)* #h(1fr) 2333/23 --  
2333/23 \  
#lorem(5) #h(1fr) #lorem(2) \  
- #lorem(10)
```

## == Work Experience

```
#chiline()
```

```
*#lorem(2)* #h(1fr) 2333/23 --  
2333/23 \  
#lorem(5) #h(1fr) #lorem(2) \  
- #lorem(20)  
- #lorem(30)  
- #lorem(40)
```

```
*#lorem(2)* #h(1fr) 2333/23 --  
2333/23 \  
#lorem(5) #h(1fr) #lorem(2) \  
- #lorem(20)  
- #lorem(30)  
- #lorem(40)
```

来源: chicv ↗

Alex Chi

skyzh@cmu.edu | github.com/skyzh | skyzh.dev

## Education

### Lorem ipsum.

Lorem ipsum dolor sit amet.

2333/23 – 2333/23

Lorem ipsum.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

### Lorem ipsum.

Lorem ipsum dolor sit amet.

2333/23 – 2333/23

Lorem ipsum.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## Work Experience

### Lorem ipsum.

Lorem ipsum dolor sit amet.

2333/23 – 2333/23

Lorem ipsum.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliquam quaerat.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliquam quaerat voluptatem. Ut enim aequo doleamus animo, cum corpore dolemus, fieri.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliquam quaerat voluptatem. Ut enim aequo doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere.

### Lorem ipsum.

Lorem ipsum dolor sit amet.

2333/23 – 2333/23

Lorem ipsum.

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliquam quaerat.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliquam quaerat voluptatem. Ut enim aequo doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere.

## • nju-thesis-typst ↗

- ▶ 总共开发时间：一周
- ▶ 语法简洁、编译迅速
- ▶ 通过「闭包」封装保存全局配置
- ▶ 本科生模板 + 研究生模板

```
// 文稿设置
#show: doc
// 封面页
#cover()
// 声明页
#decl-page()
// 前言
#show: preface
// 中文摘要
#abstract(
    keywords:
```

```
("我", "就是", "测
试用", "关键词")
)[
    中文摘要
]
// 目录
#outline-page()
// 插图目录
#list-of-
figures()
// 表格目录
```

```
#list-of-tables()
// 正文
#show: mainmatter
基本功能
== 脚注
我们可以添加一个脚
注。#footnote[脚注
内容]
```



# 南京大學

本科毕业论文

院 系 某学院  
专 业 某专业  
题 目 基于 Typst 的  
南京大学学位论文  
年 级 20XX 学 号 1234567890  
学生姓名 张三  
指导教师 李四 职 称 教授  
提交日期 2024 年 03 月 16 日

# 制作 Slides

---

# Touying

- Touying 是为 Typst 开发的 Slides 包，类似于 LATEX 的 Beamer。
  - 取自中文「投影」，而 Beamer 是德语「投影仪」的意思。 ↗
- 基本框架：
  - 全局单例对象 `s` 保存标题、作者和日期等信息。
  - 使用 `=节`、`== 小节` 和 `==== 标题` 划分 Slides 结构。
  - 使用 `#slide[...]` 块来实现更优雅且精细的控制。
- 使用主题： `#let s = themes.university.register()`
- 动画：
  - `#pause` 和 `#meanwhile` 标记。
  - `#only("2-")[]`、`#uncover("2-")[]` 和 `#alternatives[][][]`。

# Touying

```
#import "@preview/touying:0.3.2": *

#let s =
themes.aqua.register(aspect-
ratio: "16-9", lang:
"en")
#let s = (s.methods.info)
(
  self: s,
  title: [Title],
  subtitle: [Subtitle],
  author: [Authors],
  date: datetime.today(),
  institution:
[Institution],
)
#let (init, slides,
touying-outline, alert) =
utils.methods(s)
#show: init

#show strong: alert

#let (slide, title-slide,
outline-slide, focus-
slide) = utils.slides(s)
#show: slides
```

= The Section

== Slide Title

```
#slide[
  #lorem(40)
]
```

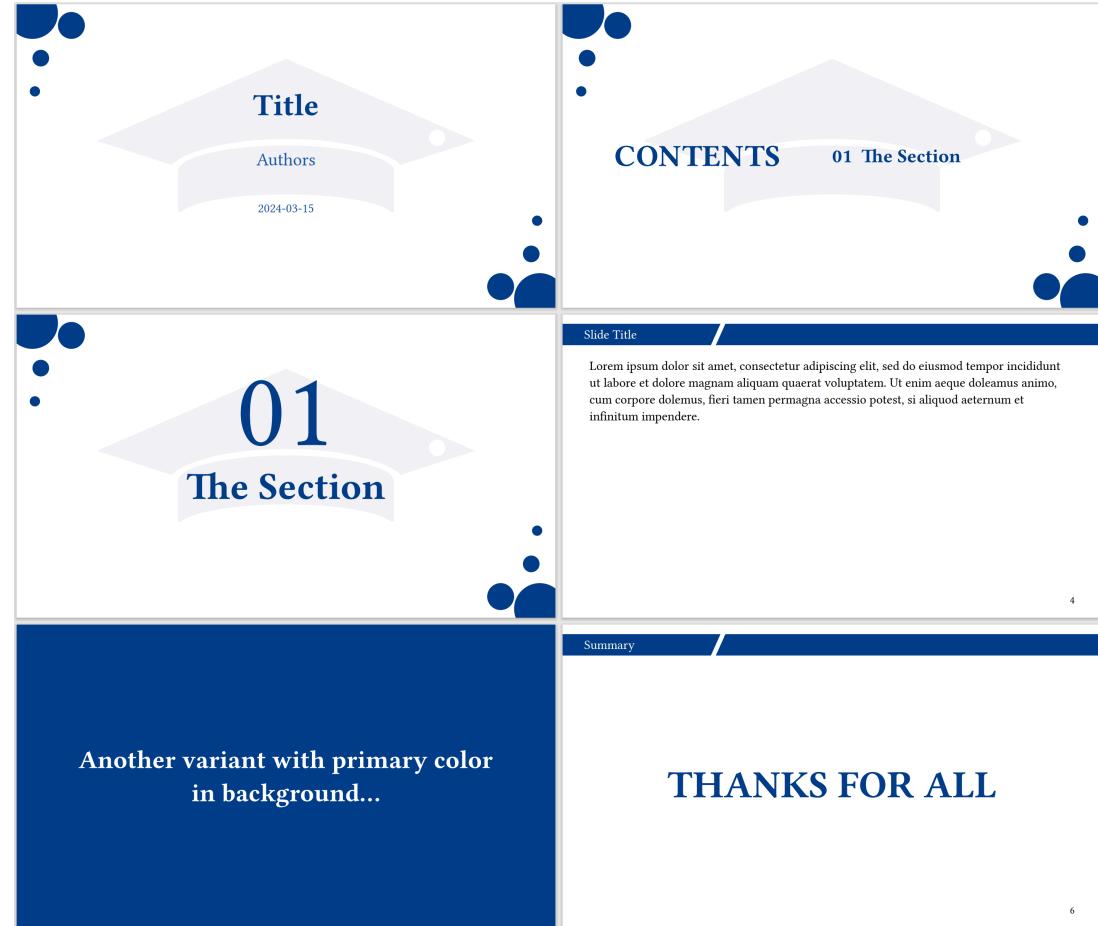
#focus-slide[

Another variant with primary color in background...

== Summary

```
#align(center + horizon)[
  #set text(size: 3em,
weight: "bold",
s.colors.primary)
  THANKS FOR ALL
]
```

来源: Touying 文档 ↗



- Pinit 包提供基于「图钉」(pin) 进行相对定位的能力。
- 可以方便地实现「箭头指示」与「解释说明」的效果。
- 一个简单示例：

```
#import "@preview/pinit:0.1.3": *  
#set text(size: 24pt)
```

A simple `#pin(1)highlighted`  
`text#pin(2).`

```
#pinit-highlight(1, 2)
```

```
#pinit-point-from(2)[It is simple.]
```

A simple highlighted text.

It is simple.

$$\frac{q_T^* p_T}{p_E} p_E^* \geq (c + q_T^* p_T^*)(1 + r^*)^{2N}$$

price of Terran goods, on Trantor

quantity of Terran goods

使用 Typst 和 Pinit 复刻算法课的 Slides, 样式来源于  $\mathcal{C}$

示例代码 

## Asymptotic Notation: $O$

Use asymptotic notations to describe asymptotic efficiency of algorithms.  
(Ignore constant coefficients and lower-order terms.)

Given a function  $g(n)$ , we denote by  $O(g(n))$  the following **set of functions**:  
 $\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{ such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

$f(n) = O(g(n))$ :  ~~$f(n)$  is asymptotically smaller than  $g(n)$ .~~

$f(n) \in O(g(n))$ :  $f(n)$  is asymptotically at most  $g(n)$ .

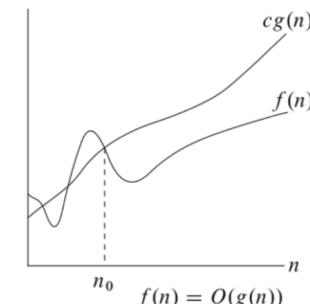
Insertion Sort as an example:

- Best Case:  $T(n) \approx cn + c'n - c''$   $T(n) = O(n)$
- Worst case:  $T(n) \approx cn + (c'/2)n^2 - c''$   $T(n) = O(n^2)$

Q: Is  $n^3 = O(n^2)$ ? How to prove your answer?

No.

Show that the equation  $\left(\frac{3}{2}\right)^n \geq c$  has infinitely many solutions for  $n$ .



# Touying 对比其他 Slides 方案

制作 Slides

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

- 能不能插入 LaTeX 公式?
  - 可以, 只需要使用 MiTeX 包。 ↗
- 能不能够加入 GIF 动图或者视频?
  - GIF 动图可以, 但是要使用 **Typst Preview** 插件的 Slide 模式。
    - 这是因为 **Typst Preview** 插件是基于 SVG 的。
- 插入图片方便吗?
  - 方便, 比如本讲座的 Slides 就有一堆图片。
    - 你可以使用 **grid** 布局。
    - 也可以使用 **Pinit** 包的 「图钉」 功能。

# 包管理

---

# Typst 包管理

- Typst 已经有了一个简单但强大的包管理方案。
  - 包可以通过 `#import "@preview/pkg:1.0.0"` 的方式导入。
    - 按需自动下载和自动导入第三方包。
      - 因此我们不需要像 TexLive 一样全量安装吃满硬盘。
      - 使用 `@preview` 命名空间。
      - 需要写上版本号，以保证文档源代码可复现性。
    - 包目前存放于统一的 GitHub Repo 中。 ↗
    - 包可以是 Package 和 Template。
    - 包也可以存放在本地，并且可以全局导入。
  - Typst 有一个 Typst Universe，可以浏览已有包。 ↗

- WASM 是一种基于 Web 的跨平台汇编语言表示。
- Typst 有 WASM Plugin 功能，也就是说：
  - Typst 的包并不一定要是纯 Typst 代码。
  - Typst 的包基本上可以用任意语言编写，例如 Rust 和 JS。
- 一些 WASM 包的例子：
  - jogs：封装 QuickJS，在 Typst 中运行 JavaScript 代码。
  - pyrunner：在 Typst 中运行 Python 代码。
  - tiaoma：封装 Zint，生成条码和二维码。
  - diagraph：在 Typst 中使用 Graphviz。

# Typst 周边生态开发体验

---

- **Touying**: Touying 是为 Typst 开发的 Slides 包。 ↗
- **MiTeX**: 一个 Rust 写的转译器，用于快速地渲染 LaTeX 公式。 ↗
- **Pinit**: 提供基于「图钉」(pin) 进行相对定位的能力。 ↗
- **nju-thesis-typst**: 基于 Typst 的南京大学学位论文。 ↗
- **Chinese-Resume-in-Typst**: 美观的 Typst 中文简历。 ↗
- **Tablem**: 在 Typst 中支持 Markdown 形式的表格。 ↗
- **Typst Sympy Calculator**: 在 VS Code 中做科学符号运算。 ↗
- **Typst Sync**: 云端同步本地包的 VS Code 插件。 ↗

- **Typst** 生态现状：~~勃勃生机，万物竞发~~
- 语法简单，强类型语言，易于开发和调试。
  - 写起 DSL 也很方便，比如 **MiTeX**、**Touying** 和 **Tablem**。
- 还有很多功能可以开发，~~例如把 LATEX 的宏包全都复刻一遍。~~
- 一些例子：
  - 国人开发的 **Tinymist** 插件和 **Typst Preview** 插件。
  - **Pandoc** 支持和 **Quarto** 支持。
  - 在网页上运行 **Typst**：typst.ts 和 typst-book。 ↗
  - 在 **VS Code** 的编辑器里显示数学符号的 **Typst Math** 插件。

最后

---

# 参考与鸣谢

- [1] Typst 官方文档 ↗
- [2] 现代 L<sup>A</sup>T<sub>E</sub>X 入门讲座 ↗
- [3] Typst 中文教程 ↗
- [4] Typst 非官方中文交流群 793548390
- [5] 南京大学 Typst 交流群 943622984

本幻灯片：<https://github.com/OrangeX4/typst-talk>

最后更新：2024-03-17

**License：** CC BY-SA 4.0

作者：OrangeX4 ©

#thank