

SLIDEV Docs - 简体中文



Sliddev

Table of contents

• Cn - README	4
• Cn - TRANSLATIONS	7
• Addons - Use	10
• Addons - Write an /addon	11
• Builtin - Components	13
• Builtin - Layouts	15
• Custom - Config katex	19
• Custom - Config mermaid	20
• Custom - Config monaco	21
• Custom - Config shortcuts	23
• Custom - Config vite	25
• Custom - Config vue	26
• Custom - Config windicss	27
• Custom - Directory structure	28
• Custom - Fonts	32
• Custom - Global layers	34
• Custom - Highlighters	36
• Custom - Index	38
• Custom - Vue context	40
• Guide - Animations	42
• Guide - Drawing	46
• Guide - Editors	48
• Guide - Exporting	51
• Guide - Faq	52
• Guide - Hosting	55
• Guide - Index	58
• Guide - Install	61
• Guide - Navigation	66
• Guide - Presenter mode	68
• Guide - Recording	69
• Guide - Syntax	70
• Guide - Why	79
• Resources - Covers	82
• Resources - Learning	83

• Cn - Showcases	84
• Themes - Gallery	85
• Themes - Use	86
• Themes - Write a /theme	87



Slidev

为开发者打造的演示文稿工具 🤖🤖🤖

v0.37.1 | 5.9k/month | [中文文档](#) | [主题](#)

Stars 23k

[简体中文](#) | [English](#)

翻译版本

	仓库	地址	维护者
English	docs	sli.dev	@antfu
简体中文	docs-cn	cn.sli.dev	@QC-L @Ivocin
Français	docs-fr	fr.sli.dev	@ArthurDanjou
Español	docs-es	es.sli.dev	@owlnai
Русский	docs-ru	ru.sli.dev	@xesjkeee
Việt Nam	docs-vn	vn.sli.dev	@bongudth
Deutsch	docs-de	de.sli.dev	@fabiankachlock
Português (BR)	docs-br	br.sli.dev	@luisfelipesdn12
Ελληνικά	docs-el	el.sli.dev	@GeopJr
日本語	docs-ja	ja.sli.dev	@IkumaTadokoro

Slidev 中文文档

-  **Markdown 支持** —— 使用你最喜欢的编辑器和工作流编写 Markdown 文件
-  **对开发者友好** —— 内置代码高亮、实时编码等功能
-  **可定制主题** —— 以 npm 包的形式共享、使用主题
-  **灵活样式** —— 使用 Windi CSS 按需使用的实用类和易用的内嵌样式表
-  **交互** —— 无缝嵌入 Vue 组件
-  **演示者模式** —— 可以使用另一个窗口，甚至是你的手机来控制幻灯片
-  **LaTeX 支持** —— 内置了对 LaTeX 数学公示的支持
-  **图表支持** —— 使用文本描述语言创建图表
-  **图标** —— 能够直接从任意图标库中获取图标
-  **编辑器** —— 集成的编辑器，或者使用 VS Code 扩展
-  **录制** —— 内置录制功能和摄像头视图
-  **跨平台** —— 能够导出 PDF、PNG 文件，甚至是一个可以托管的单页应用
-  **快速** —— 基于 Vite 的即时重载
-  **可配置** —— 支持使用 Vite 插件、Vue 组件以及任何的 npm 包

与官网文档同步

目前 Slidev 中文文档翻译已全部完成。

同步原理：[印记中文](#) 的机器人每天会拉取 [Slidev 的英文文档](#) 的内容，然后自动合并到 `main` 或 PR 到 `main` 分支。

仓库维护者需要每天检查该类型的 PR：

- 如无冲突，会自动 merge 到 `main` 分支中；
- 若有冲突，需要先解决 PR 中的冲突，再 merge 到 `main` 分支中。

参与贡献

贡献者首先 fork 本仓库，基于 `main` 创建新的翻译分支，进行翻译。翻译完成后，发起到 `main` 的 PR，等待 review，review 通过，仓库维护者将 PR merge。

若需要本地预览网站效果，可执行如下命令：

```
# 全局安装 pnpm
$ npm i -g pnpm

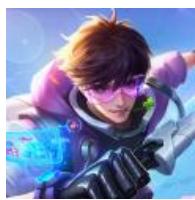
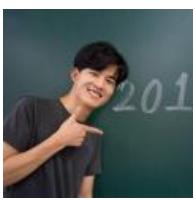
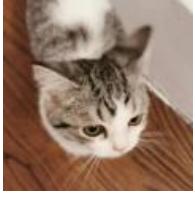
# 安装依赖，使用 pnpm
$ pnpm i
# 启动文档
$ pnpm run dev
```

接着访问 <http://localhost:3000>，即可。

或者安装 [基于 VSCode 的 Vite 插件](#) 进行编辑。

贡献者

感谢各位贡献者的付出（以下排名不分先后）：

Anthony Fu	QiChang Li	清秋	Chuck	Songhn	ArcherGu
					
Jacob	Qiang	raintygao	六个骨头	Kim Yang	KnowsCount
					
ZhengX					
					

[Go to TOC](#)

Help on Translating

First of all, thank you for being interested in contributing to translations!

You can find the repositories for each existing translation in [README.md](#). To help improve them, simply sending a Pull Request to their repo.

If the language you want to contribute isn't on the list, join [our Discord server](#), and find the `#translations` channel to see if someone is already working on the language you want, consider joining them and translate together. If not, you can start a new translation project with the following steps.

In case it's already been translated but you're wondering how to maintain it, skip to the end. ## Some tips before you get started

- It is recommended that you use your IDE of choice (e.g VSCode) paired with a development server running, so you can see your translation changes in real-time.
- You can mark these checkmarks as the translation progresses or use your own workflow. The translations don't need to be made in any particular order.
- Translations don't need to be literal, but they should convey the same message. In case you're not sure how to translate something, you can either leave it as it is or use online tools like WordReference or Linguee to aid you.
- Most translations will simply consist in editing Markdown files. Certain areas are buried under Vue components, which will be listed below. You can also use your IDE to find the string to translate.

Getting started

- Fork the main docs repo: [slidevjs/docs](#)
- Translate README.md, you can take one of the already translated repositories as an example.
- Share your repo's link to the `#translations` channel telling people you are working on it and find collaborators.

Translating Markdown files

- `showcases.md` - A gallery showcase of Slidev presentations.
- `index.md` - Mainpage content, note that some of it is buried under Vue components listed further below.

.vitepress/

- `config.js` - Sitemap
- `/theme/components/WorkingInProgress.vue` - WIP notice shown in mainpage
- `/theme/components/demo/Demo.vue` - Animated demo shown in mainpage
- `/theme/components/Environment.vue` - Describes the environment of a setting.

builtin/

- `components.md` - Use [Vue components](#) inside Slidev
- `layouts.md` - Use Vue layouts inside Slidev

custom/

- `config-katex.md` - Configuring Katex
- `config-mermaid.md` - Configuring Mermaid
- `config-monaco.md` - Configuring Monaco
- `config-shortcuts.md` - Configuring Shortcuts
- `config-vite.md` - Configuring Vite
- `config-vue.md` - Configuring Vue
- `config-windicss.md` - Configuring Windicss
- `directory-structure.md` - Configuring the directory structure
- `fonts.md` - Configuring fonts
- `global-layers.md` - Configuring the global layers
- `highlighters.md` - Configuring code highlighters
- `index.md` - Customizations index page
- `vue-context.md` - The Vue global context

guide/

- `animations.md` - Animations and transitions
- `editors.md` - Editor integrations
- `exporting.md` - Exporting your slides
- `faq.md` - Frequent Answered Questions
- `index.md` - Getting started with Slidev
- `navigation.md` - Navigation across slides
- `presenter-mode.md` - Toggling presenter mode
- `recording.md` - Recording your presentation
- `syntax.md` - Markdown syntax
- `why.md` - *Why Slidev?*

resources/

- `covers.md` - Curated covers for Slidev

themes/

- `gallery.md` - Theme gallery
- `use.md` - How to use Slidev themes
- `write-a-theme.md` - Write your own theme

addons/

- [use.md](#) - How to use Slidev addons
- [write-an-addon.md](#) - Write your own addon

Publishing your translations

- When you finish the translation (at least 90%), [@antfu](#) in the Discord and we will invite you to the org and make the translation official.
- Once the transferring is done, we will set up the subdomain, auto-deployment, and a daily sync-up bot to keep the translation up-to-date with the latest English docs.
- The site is live, and we will send a shout-out tweet on [our Twitter account](#).

Maintaining the translations up-to-date

- [docschina-bot](#) will periodically submit merge requests from the [slidev/docs](#) repository. Switch to the branch created in the pull request, make any changes necessary and merge it. [example](#).
- Sometimes it will occur that a merge request is made and you haven't merged the previous one. The latest PR always checks your main branch against the English one; so you can just close the previous PR(s), move your work to the latest one and merge it.

[Working-in-progress translation list](#)

Thanks again!

[Go to TOC](#)

使用扩展插件

扩展插件是你可以你在演示文稿中使用的附加组件、布局、样式、配置等集。

它们与 [主题](#) 相似，但总而言之：

- 它们不影响幻灯片的全局样式
- 你可以在同一演示文稿中使用多个插件

为了使用扩展插件，你必须通过以下方式手动安装它们：

```
$ npm install [slidev-addon-package1] [slidev-addon-package2]
```

然后，你需要在 `frontmatter` 中声明你的扩展插件：

```
---  
addons:  
  - slidev-addon-package1  
  - slidev-addon-package2  
---
```

或者在你的文件 `package.json` 中：

```
// package.json  
{  
  "slidev": {  
    "addons": [  
      "slidev-addon-package1",  
      "slidev-addon-package2",  
    ]  
  }  
}
```

编写扩展插件

自 v0.32.1 起可用

扩展插件能力

一个扩展插件可以自定义以下功能：

- 全局样式（谨慎使用，它更通常作为 [主题](#) 的能力）
- 自定义布局或者重写现有布局
- 自定义组件或者重写现有组件
- 扩展 Windi CSS 配置
- 配置 Monaco、Prism 等工具

约定

扩展插件发布到 npm，需遵循以下约定：

- 包名应该以 `slidev-addon-` 开头，例如：`slidev-addon-awesome`
- 在主题 `package.json` 的 `keywords` 中添加 `slidev-addon` 和 `slidev` 关键词

配置说明

初始化

如果想要创建扩展插件，请首先创建一个带有 `package.json` 文件的目录（你可以使用 `npm init`）。

然后，安装 Slidev 依赖项：

```
$ npm install -D @slidev/cli
```

测试

如果想要测试自己编写的扩展插件，你可以新建 `example.md` 并在其内填写一些内容。

你还可以在 `package.json` 增加一些脚本以方便测试：

```
// package.json
{
  "scripts": {
    "dev": "slidev example.md",
    "build": "slidev build example.md",
    "export": "slidev export example.md",
    "screenshot": "slidev export example.md --format png"
  }
}
```

你只需在命令行中执行 `npm publish` 就可以发布自己的扩展插件，并不需要额外的构建过程（这意味着你可以直接发布 `.vue` 和 `.ts` 文件，Slidev 可以直接识别它们）。

扩展插件可以定制的范围与本地自定义相一致，可以参阅 [自定义文档](#)。

扩展插件原数据

Slidev 版本

如果主题依赖于 Slidev 的某项新特性，你可以为主题设置最小的 Slidev 版本，以使你的主题可以正常工作：

```
// package.json
{
  "engines": {
    "slidev": ">=0.32.1"
  }
}
```

如果用户使用的是旧版本的 Slidev，将会抛出错误。

组件

内置组件

这部分文档尚未完成。在完成之前，你可以直接去看看 [源码](#)。

Toc {#toc}

插入目录。

如果你想让一张幻灯片不出现在 `<Toc>` 组件中，你可以在幻灯片的 matter 块中使用如下属性：

```
---
```

```
hideInToc: true
```

```
---
```

标题使用 `<Titles>` 组件 来展示

用法

```
<Toc />
```

参数：

- `columns (string | number, 默认值：1)` : 要显示的列数The number of columns of the display
- `listClass (string | string[], 默认值：'')` : 用于修饰目录的 class
- `maxDepth (string | number, 默认值：Infinity)` : 要显示标题的最大深度
- `minDepth (string | number, 默认值：1)` : 要显示标题的最小深度
- `mode ('all' | 'onlyCurrentTree' | 'onlySiblings', 默认值：'all')`:
 - `'all'` : 显示所有项目
 - `'onlyCurrentTree'` : 只显示当前树中的项目（活跃的项目，及其父节点以及子节点）
 - `'onlySiblings'` : 只显示当前树中的项目和它们的直接兄弟姐妹

Link {#link}

插入一个链接，你可以用它来导航到一个指定的幻灯片。

用法

```
<Link to="42">Go to slide 42</Link>
<Link to="42" title="Go to slide 42"/>
```

参数：

- `to (string | number)` : 幻灯片的路径，以导航到对应位置（幻灯片下标从 1 开始）
- `title (string)` : 要显示的标题

Titles {#titles}

在一个被解析为 HTML 的幻灯片中插入主标题

标题和标题级别会自动从每张幻灯片的第一个标题元素中检索出来。

目录的标题和标题层级根据每张幻灯片上的第一个标题元素自动生成。

可以使用前端语法覆盖幻灯片的这种自动生成目录行为：

```
---  
title: Amazing slide title  
level: 2  
---
```

用法

<Titles> 组件是一个虚拟组件，你可以使用如下方式导入：

```
import Titles from '@/slidev/titles.md'
```

然后你可以这样使用：

```
<Titles no="42" />
```

参数：

- `no (string | number)`: 显示标题的幻灯片编号（幻灯片下标从 1 开始）

自定义组件

在你的项目根目录里创建一个 `components/` 文件夹，然后直接把你的自定义 Vue 组件放进去；然后你就可以在你的 markdown 文件里使用该组件啦！

欲了解更多，请参阅 [自定义](#) 章节。

主题提供的组件

同时主题也可以提供组件。请阅读它们各自的文档，以知晓它们提供了什么。

欲了解更多，请参阅 [路径结构](#) 章节。

[Go to TOC](#)

布局

内置布局

由于主题可能会覆盖布局的行为，因此精确理解主题的使用、参数和例子最好的方式是查阅主题文档。

center

在屏幕中间展示内容。

cover

用来展示演讲稿的封面页，可以包含演讲的标题、演讲者、时间等。

default

最基础的布局，用于展示任意类型的内容。

end

演讲的最后一页。

fact

用来在屏幕上突出展示很多事实或数据。

full

使用屏幕全部空间来展示内容。

image-left

在屏幕左侧展示图片，屏幕右侧展示内容。

用法

```
---  
layout: image-left  
  
# the image source  
image: ./path/to/the/image  
  
# a custom class name to the content  
class: my-cool-content-on-the-right  
---
```

image-right

在屏幕右侧展示图片，屏幕左侧展示内容。

用法

```
---
```

```
layout: image-right
```

```
# the image source
```

```
image: ./path/to/the/image
```

```
# a custom class name to the content
```

```
class: my-cool-content-on-the-left
```

```
---
```

image

将图片作为页面的主要内容进行展示。

用法

```
---
```

```
layout: image
```

```
# the image source
```

```
image: ./path/to/the/image
```

```
---
```

iframe-left

Shows a web page on the left side of the screen, the content will be placed on the right side.

Usage

```
---
```

```
layout: iframe-left
```

```
# the web page source
```

```
url: https://github.com/slidesjs/slides
```

```
# a custom class name to the content
```

```
class: my-cool-content-on-the-right
```

```
---
```

iframe-right

Shows a web page on the right side of the screen, the content will be placed on the left side.

Usage

```
---
```

```
layout: iframe-right
```

```
# the web page source
```

```
url: https://github.com/slidesjs/slides
```

```
--  
# a custom class name to the content  
class: my-cool-content-on-the-left  
---
```

iframe

Shows a web page as the main content of the page.

Usage

```
--  
layout: iframe  
  
# the web page source  
url: https://github.com/slidesjs/slides  
---
```

intro

介绍演讲稿，通常带有演讲稿标题、简述、作者等信息。

none

没有任何样式的布局。

quote

突出显示引文。

section

用来标记演讲稿的新部分的开始。

statement

将主张/声明作为主要页面内容。

two-cols

将页面内容分为两列。

用法

```
--  
layout: two-cols  
--  
  
# Left  
  
This shows on the left  
::right::
```

Right

This shows on the right

自定义布局

在你的项目根目录里创建一个 `layouts/` 文件夹，然后直接把你的自定义 Vue 组件放进去。

欲了解更多，请参阅 [自定义](#) 章节。

主题提供的布局

主题可以提供布局或者覆盖已有的。请阅读它们各自的文档，以知晓它们提供了什么。

[Go to TOC](#)

配置 KaTeX

创建一份包含以下内容的 `./setup/katex.ts` 文件：

```
import { defineKatexSetup } from '@slidev/types'

export default defineKatexSetup(() => {
    return {
        /* ... */
    }
})
```

在配置时，你可以提供一些自定义的 [KaTeX 选项](#)。关于更多配置详情，请参考其类型定义和相关文档。

配置 Mermaid

创建一份包含以下内容的 `./setup/mermaid.ts` 文件：

```
import { defineMermaidSetup } from '@slidev/types'

export default defineMermaidSetup(() => {
  return {
    theme: 'forest',
  }
})
```

在配置时，你可以为 `Mermaid` 提供一些自定义的设置。关于更多配置详情，请参考其类型定义和相关文档。

配置 Monaco

创建一份包含以下内容的 `./setup/monaco.ts` 文件：

```
import { defineMonacoSetup } from '@slidev/types'

export default defineMonacoSetup(async (monaco) => {
  // 使用 `monaco` 配置
})
```

访问 [monaco-editor](#) 了解更多关于 Monaco 配置的相关信息。

用法

如果你想在你的幻灯片中使用 Monaco，只需添加 `{monaco}` 到你的代码片段中：

```
//``js
const count = ref(1)
const plusOne = computed(() => count.value + 1)

console.log(plusOne.value) // 2
plusOne.value++ // 报错
//``
```

修改为

```
//``js {monaco}
const count = ref(1)
const plusOne = computed(() => count.value + 1)

console.log(plusOne.value) // 2
plusOne.value++ // 报错
//``
```

导出

默认情况下，Monaco 只会在开发模式开启。如果你想在导出的单页应用中使用，你需要配置你的扉页：

```
---  
monaco: true # 默认为 "dev"  
---
```

类型自动安装

当你使用 Monaco 编写 TypeScript 时，类型依赖将会自动安装到客户端。

```
//``ts {monaco}
import { ref } from 'vue'
import { useMouse } from '@vueuse/core'
```

```
const counter = ref(0)
//``
```

在上面的示例中，只需确保项目依赖（`dependencies` 或 `devDependencies`）中包含所用到的 `vue` 和 `@vueuse/core`，Slidev 将处理剩余的部分以使你的 Monaco 编辑器获得正确的类型支持。

配置主题

Monaco 主题受 Slidev 的亮色/暗色主题控制。如果你想定制主题，可以在配置函数中传入主题 id：

```
// ./setup/monaco.ts
import { defineMonacoSetup } from '@slidev/types'

export default defineMonacoSetup(() => {
  return {
    theme: {
      dark: 'vs-dark',
      light: 'vs',
    },
  }
})
```

如果你想载入自定义主题：

```
import { defineMonacoSetup } from '@slidev/types'

// 改成你的主题
import dark from 'theme-vitesse/themes/vitesse-dark.json'
import light from 'theme-vitesse/themes/vitesse-light.json'

export default defineMonacoSetup((monaco) => {
  monaco.editor.defineTheme('vitesse-light', light as any)
  monaco.editor.defineTheme('vitesse-dark', dark as any)

  return {
    theme: {
      light: 'vitesse-light',
      dark: 'vitesse-dark',
    },
  }
})
```

如果你在创建一个 Slidev 主题，在配置函数中使用动态 `import()` 以获得更好的 tree-shaking 和代码分割结果。

配置快捷键

自 v0.20 起可用

Since v0.35.6 (excluded), you decide which base shortcuts to keep (see `...base`, below).

创建一份包含以下内容的 `./setup/shortcuts.ts` 文件：

```
import type { NavOperations, ShortcutOptions } from '@slidev/types'
import { defineShortcutsSetup } from '@slidev/types'

export default defineShortcutsSetup((nav: NavOperations, base: ShortcutOptions[]) => {
  return [
    ...base, // keep the existing shortcuts
    {
      key: 'enter',
      fn: () => nav.next(),
      autoRepeat: true,
    },
    {
      key: 'backspace',
      fn: () => nav.prev(),
      autoRepeat: true,
    },
  ],
})
```

在配置时，你可以添加或者一些自定义的快捷键。例如，上面的配置为 `enter` 绑定了下一动画或幻灯片，为 `backspace` 绑定了上一动画或幻灯片。

配置函数会接收一个封装有导航函数的对象参数，返回一个快捷键配置信息的数组，你可以参考其类型定义获得详细信息。

该 `key` 仅支持字符串类型，但你也可以使用如下约定绑定多个快捷键：

```
import type { NavOperations, ShortcutOptions } from '@slidev/types'
import { defineShortcutsSetup } from '@slidev/types'

export default defineShortcutsSetup((nav: NavOperations, base: ShortcutOptions[]) => {
  return [
    ...base,
    {
      key: 'ShiftLeft+ArrowRight',
      fn: () => nav.next(),
      autoRepeat: true,
    },
  ],
})
```

关于键盘事件, 请参考 [useMagicKeys | VueUse](#)。

[Go to TOC](#)

配置 Vite

Slidev 基于 [Vite](#) 实现。这意味着你可以利用 Vite 强大的插件系统来进一步定制你的幻灯片。

如果项目中存在 `vite.config.ts` 文件，将被读取。

Slidev 已经内置了以下插件：

- [@vitejs/plugin-vue](#)
- [unplugin-vue-components](#)
- [unplugin-icons](#)
- [vite-plugin-vue-markdown](#)
- [vite-plugin-remote-assets](#)
- [vite-plugin-windicss](#)
- [unocss/vite](#)

可以在 [此处](#) 了解 Slidev 的相关预设。

配置内部插件

自 v0.21 起可用

如需对内置插件列表进行配置，先创建 `vite.config.ts`，其内容如下。请注意，Slidev 对这些插件有些预设配置，如下做法会覆盖其中一些配置，可能会导致应用崩溃。请将此功能视为**高级功能**，在继续操作前，请确保你知道自己在干什么。

```
import { defineConfig } from 'vite'

export default defineConfig({
  slidev: {
    vue: {
      /* vue options */
    },
    markdown: {
      /* markdown-it options */
      markdownItSetup(md) {
        /* custom markdown-it plugins */
        md.use(/* ... */)
      },
      /* options for other plugins */
    },
  },
})
```

欲了解更多，请参阅 [类型声明](#)。

[Go to TOC](#)

配置 Vue

Slidev 基于 [Vue 3](#) 来渲染应用。你可以针对应用进行扩展，添加自定义插件或自定义配置等操作。

创建 `./setup/main.ts` 文件，其内容如下：

```
import { defineAppSetup } from '@slidev/types'

export default defineAppSetup(({ app, router }) => {
  // Vue App
  app.use(YourPlugin)
})
```

这也可以作为你 Slidev 应用程序的主入口，在应用启动前做一些初始化操作。

了解更多：[Vue Application API](#)。

配置 Windi CSS

Markdown 天然支持 HTML。因此，你可以按照你想要的方式对你的内容进行样式设计。为了提供一些便利，我们内置了 [Windi CSS](#)，所以你可以直接使用 Windi CSS 提供的工具类对样式进行调整。

示例：

```
<div class="grid pt-4 gap-4 grids-cols-[100px,1fr]>
  ### Name
  - Item 1
  - Item 2
</div>
```

Windi CSS v3.0 中的 [属性化（Attributify）](#) 模式已默认启用。

配置

如需配置 Windi CSS，请根据如下内容创建并配置 `setup/windicss.ts`，以对内部配置进行扩展：

```
// setup/windicss.ts

import { defineWindiSetup } from '@slidev/types'

// 对内部 windicss 配置进行扩展
export default defineWindiSetup(() => ({
  shortcuts: {
    // 自定义默认背景
    'bg-main': 'bg-white text-[#181818] dark:(bg-[#121212] text-[#ddd])',
  },
  theme: {
    extend: {
      // 字体可以被替换，请记得更新 `index.html` 中的字体链接
      fontFamily: {
        sans: 'ui-sans-serif,system-ui,-apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color Emoji"',
        mono: '"Fira Code", monospace',
      },
    },
  },
}))
```

欲了解更多，请参阅 [Windi CSS 配置](#)。

项目结构

Slidev 对项目结构进行了一些约定，以尽量减少配置项，使功能扩展更加灵活直观。

基本结构如下所示：

```
your-slidev/
  |-- components/      # 自定义组件
  |-- layouts/         # 自定义布局
  |-- public/          # 静态资源
  |-- setup/           # 自定义 setup / hooks
  |-- styles/          # 自定义样式
  |-- index.html       # 注入的 index.html
  |-- slides.md        # 幻灯片主入口
  |-- vite.config.ts   # 扩展 vite 配置
```

以上所有均为可选。

组件

约定：`./components/*.{vue,js,ts,jsx,tsx,md}`

此目录中的组件可以在幻灯片的 Markdown 中直接使用，其组件名与文件名相同。

示例：

```
your-slidev/
  |-- ...
  |-- components/
    |-- MyComponent.vue
    |-- HelloWorld.ts
```

```
<!-- slides.md -->
# My Slide
<MyComponent :count="4"/>
<!-- both namings work -->
<hello-world foo="bar">
  Slot
</hello-world>
```

此特性得益于 `unplugin-vue-components`。

Slidev 还提供了一些 [内置组件](#) 供你选择。

布局

约定：`./layouts/*.{vue,js,ts,jsx,tsx}`

```
your-slidev/
  ...
  └── layouts/
    ├── cover.vue
    └── my-cool-theme.vue
```

你可以为布局文件使用任何文件名。然后只需在你的 YAML 头部使用文件名引用你的布局。

```
---
layout: my-cool-theme
---
```

如果你提供的布局与内置布局或主题布局重名的话，你的自定义布局将优先于内置/主题布局。优先级为 `本地 > 主题 > 内置`。

在布局组件中，你可以使用 `<slot/>` 展示幻灯片内容。比如：

```
<!-- default.vue -->
<template>
  <div class="slidev-layout default">
    <slot />
  </div>
</template>
```

静态资源

约定：`./public/*`

开发过程中，此目录中的资源文件将在 `/` 下提供，并会按原样复制到 `dist` 目录的根目录中。欲了解更多，请参阅 [Vite 的 public 目录](#)。

样式

约定：`./style.css | ./styles/index.{css,js,ts}`

遵循上述约定的文件将被注入到 App 的根目录中。如果你需要引入多个 css 入口，你可以按如下方式创建结构并自行管理引入顺序。

```
your-slidev/
  ...
  └── styles/
    ├── index.ts
    ├── base.css
    ├── code.css
    └── layouts.css
```

```
// styles/index.ts

import './base.css'
import './code.css'
import './layouts.css'
```

样式得益于 [Windi CSS](#) 和 [PostCSS](#)，因此，你拥有开箱即用的 css 嵌套和 [at-directives](#)。示例：

```
.slidev-layout {
  @apply px-14 py-10 text-[1.1rem];

  h1, h2, h3, h4, p, div {
    @apply select-none;
  }

  pre, code {
    @apply select-text;
  }

  a {
    color: theme('colors.primary');
  }
}
```

了解更多关于此语法的信息。

index.html {#index-html}

约定：`index.html`

`index.html` 提供了向主 `index.html` 中注入 `meta` 标签以及 `scripts` 标签的能力。

例如，对于以下自定义 `index.html` 来说：

```
<!-- ./index.html -->
<head>
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?
family=Fira+Code:wght@400;600&family=Nunito+Sans:wght@200;400;600&display=swap"
rel="stylesheet">
</head>

<body>
  <script src="./your-scripts"></script>
</body>
```

最终部署的 `index.html` 效果如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="icon" type="image/png"
href="https://cdn.jsdelivr.net/gh/slidedevjs/slides/assets/favicon.png">
  <!-- injected head -->
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?
family=Fira+Code:wght@400;600&family=Nunito+Sans:wght@200;400;600&display=swap"
rel="stylesheet">
</head>
<body>
  <div id="app"></div>
  <script type="module" src="__ENTRY__"></script>
  <!-- injected body -->
```

```
<script src=".your-scripts"></script>
</body>
</html>
```

全局图层

约定：`global-top.vue` | `global-bottom.vue`

了解更多：[全局图层](#)

[Go to TOC](#)

字体

自 v0.20 起可用

虽然你可以使用 HTML 和 CSS 为你的幻灯片定制你想要的字体和样式，但 Slidev 提供了另一种较为便捷的方式，可以让你轻松使用它们。

你可以在 frontmatter 中，添加如下配置：

```
---  
fonts:  
  # 基础字体  
  sans: 'Robot'  
  # 与 windicss 的 `font-serif` css 类一同使用  
  serif: 'Robot Slab'  
  # 用于代码块、内联代码等  
  mono: 'Fira Code'  
---
```

按上述修改即可完成配置。

字体将从 [Google Fonts](#) 被自动引入。这意味着你可以直接使用 Google Fonts 上的任何字体。

本地字体

默认情况下，Slidev 会认为 `fonts` 配置的所有字体均来自 Google Fonts。如果你想使用本地字体，可以指定 `fonts.local` 字段来选择不使用自动引入的字体。

```
---  
fonts:  
  # 与 css 中的 font-family 一致，你可以使用 `、` 来分割字体名，便于降级  
  sans: 'Helvetica Neue, Robot'  
  # 将 'Helvetica Neue' 作为本地字体  
  local: 'Helvetica Neue'  
---
```

weights 和斜体

默认情况下，Slidev 为每种字体引入了三种 weight 大小 `200`，`400`，`600`。你可以按如下方式配置它们：

```
---  
fonts:  
  sans: 'Robot'  
  # 默认为  
  weights: '200,400,600'  
  # 引入斜体字体，默认 `false`  
  italic: false  
---
```

这些配置适用于所有的网络字体。如果要对每种字体的 weight 进行更细粒度的控制，你需要用 [HTML](#) 和 [CSS](#) 手动引入它们。

降级字体

大多数情况下，只需指定“特殊字体”即可，Slidev 会为你提供可降级的字体。例如：

```
---  
fonts:  
  sans: 'Robot'  
  serif: 'Robot Slab'  
  mono: 'Fira Code'  
---
```

其结果为：

```
.font-sans {  
  font-family: "Robot", ui-sans-serif, system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";  
}  
.font-serif {  
  font-family: "Robot Slab", ui-serif, Georgia, Cambria, "Times New Roman", Times, serif;  
}  
.font-mono {  
  font-family: "Fira Code", ui-monospace, SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace;  
}
```

如需禁用降级字体，请按如下方式配置：

```
---  
fonts:  
  mono: 'Fira Code, monospace'  
  fallback: false  
---
```

Providers

- 选项：`google` | `none`
- 默认值：`google`

目前，仅针对于 Google Fonts 进行了支持，我们计划在未来添加更多的字体服务整合。当此字段指定为 `none` 时，自动导入功能将被完全禁用，同时将所有字体视为本地字体。

```
---  
fonts:  
  provider: 'none'  
---
```

[Go to TOC](#)

全局图层

自 v0.17 起可用

全局图层允许你拥有**持续存在的**跨幻灯片自定义组件。这对于有页脚、跨幻灯片动画、全局特效等来说可能很有用。

Slidev 为这种用法提供了三种图层，在你的项目根目录下创建 `global-top.vue`、`global-bottom.vue` 或 `custom-nav-controls.vue` 文件，它们会被自动识别。

图层关系：

- 全局顶层 (`global-top.vue`)
- 幻灯片
- 全局底层 (`global-bottom.vue`)
- 导航控件
 - 自定义导航控件 (`custom-nav-controls.vue`)

示例

```
<!-- global-bottom.vue -->
<template>
  <footer class="absolute bottom-0 left-0 right-0 p-2">Your Name</footer>
</template>
```

文字 `Your Name` 将出现在你所有幻灯片中。

```
<!-- custom-nav-controls -->
<template>
  <button class="icon-btn" title="Next" @click="$slidev.nav.next">
    <carbon:arrow-right />
  </button>
</template>
```

`Next` 按钮会出现在导航栏中。

如需有条件地启用它，你可以用 [Vue 全局上下文](#)。

```
<!-- 在第四页时隐藏页脚 -->
<template>
  <footer
    v-if="$slidev.nav.currentPage !== 4"
    class="absolute bottom-0 left-0 right-0 p-2">
    Your Name
  </footer>
</template>
```

```
<!-- "cover" 布局的情况下隐藏页脚 -->
<template>
  <footer>
    <v-if="$slidev.nav.currentLayout !== 'cover'">
      <div>Your Name</div>
    </v-if>
  </footer>
</template>
```

```
<!-- 一个显示页数的页脚示例 -->
<template>
  <footer>
    <v-if="$slidev.nav.currentLayout !== 'cover'">
      <div>{{ $slidev.nav.currentPage }} / {{ $slidev.nav.total }}</div>
    </v-if>
  </footer>
</template>
```

```
<!-- custom-nav-controls -->
<!-- 在演讲者模式隐藏按钮 -->
<template>
  <button v-if="!$slidev.nav.isPresenter" class="icon-btn" title="Next">
    @click="$slidev.nav.next"
    <carbon:arrow-right />
  </button>
</template>
```

语法高亮器

Slidev 内建了两种语法高亮器供你使用：

- [Prism](#)
- [Shiki](#)

Prism 是最受欢迎的语法高亮器之一。它在代码文本中插入标签包裹需要高亮的元素并通过 CSS 文件来设置高亮样式。你可以直接使用 Prism [官方预设的主题](#)，或者通过 `prism-theme-vars` 快速创建自己的高亮主题。

Shiki，一个基于 TextMate 语法的代码高亮器。它直接生成带样式的包裹元素，所以不需要引入额外的 CSS 文件。因为基于 TextMate 语法，所以生成的高亮区块非常准确，效果类似于 VS Code。Shiki 也提供了 [很多预设主题](#)。不过 Shiki 需要通过 TextMate 主题（与 VS Code 主题相兼容）来自定义高亮，这相对来说会比较麻烦。

Slidev 主题通常会同时支持 Prism 和 Shiki，不过需要注意有些主题可能只会支持其中一种。

你可以参考以下描述来选择使用哪种高亮器：

- **Prism** 更容易自定义样式
- **Shiki** 生成的高亮区块更加准确

默认情况下 Slidev 使用 Prism，你可以在 frontmatter 中修改设置：

```
---  
highlighter: shiki  
---
```

配置 Prism

直接引入预设主题或者使用 `prism-theme-vars` 就可以配置 Prism 高亮，更多细节可以参阅相应文档。

配置 Shiki

创建 `./setup/shiki.ts` 文件并添加以下代码：

```
/* ./setup/shiki.ts */
import { defineShikiSetup } from '@slidev/types'

export default defineShikiSetup(() => {
  return {
    theme: {
      dark: 'min-dark',
      light: 'min-light',
    },
  }
})
```

所有可选的主题请参阅 [Shiki 文档](#)。

如果想使用自己的主题可以将配置文件改为：

```
/* ./setup/shiki.ts */

import { defineShikiSetup } from '@slidev/types'

export default defineShikiSetup(async({ loadTheme }) => {
  return {
    theme: {
      dark: await loadTheme('path/to/theme.json'),
      light: await loadTheme('path/to/theme.json'),
    },
  }
})
```

自定义

从样式到工具，Slidev 都是完全可以自定义的。你可以对以下这些工具进行自定义配置（[Vite](#)， [Windi CSS](#)， [Monaco](#)， 等等。）

扉页配置

你可以在 Slidev 的第一张幻灯片扉页处（文件顶部）进行各项配置，以下是各项的默认参数。

```
---  
# 主题id 或 主题包名称  
# 了解更多：https://sli.dev/themes/use.html  
theme: 'default'  
# 幻灯片的总标题，如果没有指定，那么将以第一张拥有标题的幻灯片的标题作为总标题  
title: 'Slidev'  
# titleTemplate for the webpage, `%s` will be replaced by the page's title  
titleTemplate: '%s - Slidev'  
# information for your slides, can be a markdown string  
info: false  
  
# 在单页(SPA) 构建中启用 pdf 下载，也可以指定一个自定义 url  
download: false  
# 要导出文件的文件名称  
exportFilename: 'slidev-exported'  
# 语法高亮设置，可以使用 'prism' 或 'shiki' 方案  
highlighter: 'prism'  
# 在代码块中显示行号  
lineNumbers: false  
# 启用 monaco 编辑器，可以是 boolean, 'dev' 或者 'build'  
monaco: 'dev'  
# 使用 vite-plugin-remote-assets 在本地下载远程资源，可以是 boolean, 'dev' 或者 'build'  
remoteAssets: false  
# 控制幻灯片中的文本是否可以选择  
selectable: true  
# 启用幻灯片录制，可以是 boolean, 'dev' 或者 'build'  
record: 'dev'  
  
# 幻灯片的配色方案，可以使用 'auto', 'light' 或者 'dark'  
colorSchema: 'auto'  
# vue-router 模式，可以使用 'history' 或 'hash' 模式  
routerMode: 'history'  
# 幻灯片的长宽比  
aspectRatio: '16/9'  
# canvas 的真实宽度，单位为 px  
canvasWidth: 980  
# 用于主题定制，会将属性 `x` 注入根样式 `--slidev-theme-x`  
themeConfig:  
  primary: '#5d8392'  
  
# favicon 可以是本地文件路径，也可以是一个 URL  
favicon: 'https://cdn.jsdelivr.net/gh/slidevjs/slidev/assets/favicon.png'  
# 用于渲染图表的 PlantUML 服务器的 URL  
plantUmlServer: 'https://www.plantuml.com/plantuml'  
# 字体将从 Google 字体自动导入  
# 了解更多：https://sli.dev/custom/fonts  
fonts:  
  sans: 'Roboto'
```

```
serif: 'Roboto Slab'  
mono: 'Fira Code'  
  
# 为所有幻灯片添加默认的 frontmatter  
defaults:  
  layout: 'default'  
  # ...  
  
# 绘制选项  
# 了解更多：https://sli.dev/guide/drawing.html  
drawings:  
  enabled: true  
  persist: false  
  presenterOnly: false  
  syncAll: true  
---
```

你可以从 [类型定义](#) 获取到更多的配置信息。

目录结构

Slidev 使用特定的目录结构来减少配置的复杂度，并使功能扩展更加灵活和直观。

具体请参考 [目录结构](#) 章节。

Config Tools

- [语法高亮器](#)
- [配置 Vue](#)
- [配置 Vite](#)
- [配置 Windi CSS](#)
- [配置 Monaco](#)
- [配置 KaTeX](#)
- [配置 Mermaid](#)

[Go to TOC](#)

Vue 全局上下文

Slidev 注入了一个 全局的 Vue 上下文 `$slidev`，它用于高级的条件判断或导航控制。

用法

你可以在你的 markdown 文件以及 Vue 模板的任何位置使用 "Mustache" 语法 访问它。

```
<!-- slides.md -->
# Page 1
Current page is: {{ $slidev.nav.currentPage }}
```

```
<!-- Foo.vue -->
<template>
  <div>Title: {{ $slidev.configs.title }}</div>
  <button @click="$slidev.nav.next">Next Page</button>
</template>
```

属性

`$slidev.nav` {#slidev-nav}

一个响应式对象，它拥有幻灯片导航的属性以及控制权。例如：

```
$slidev.nav.next() // 执行下一步
$slidev.nav.nextSlide() // 跳转下一张幻灯片 (忽略 v-clicks)
$slidev.nav.go(10) // 去到幻灯片的第 10 页
```

```
$slidev.nav.currentPage // 获取当前幻灯片的页数
$slidev.nav.currentLayout // 当前的布局 id
$slidev.nav.clicks // 目前的点击次数
```

欲了解更多可用属性，请参阅 [nav.ts](#) 的 exports。

`$slidev.configs` {#slidev-configs}

一个响应式对象，它存储着你 `slides.md` 中解析后的 第一个 frontmatter 中的配置

```
---
title: My First Slidev!
---
```

```
{{ $slidev.configs.title }} // 'My First Slidev!'
```

\$slidev.themeConfigs {#slidev-themeconfigs}

一个响应式对象，它存储着解析后的主题配置。

```
---  
title: My First Slidev!  
themeConfig:  
  primary: #213435  
---
```

```
{} $slidev.themeConfigs.primary }} // '#213435'
```

动画

点击动画

v-click

如需为元素添加“点击动画”，你可以使用 `v-click` 指令或 `<v-click>` 组件

```
# Hello

<!-- 组件用法：在你按下“下一步”之前，这是不可见的 -->
<v-click>

Hello World

</v-click>

<!-- 指令用法：在你第二次按下“下一步”之前，这是不可见的 -->
<div v-click class="text-xl p-2">

Hey!

</div>
```

v-after

`v-after` 和 `v-click` 用法类似，但是 `v-after` 会在上一个 `v-click` 触发后使元素可见。

```
<div v-click>Hello</div>
<div v-after>World</div>
```

当你点击了“下一步”按钮之后，`Hello` 和 `World` 会同时出现。

v-click-hide

与 `v-click` 相同，但不是让元素出现，而是让元素在点击后不可见。

```
<div v-click-hide>Hello</div>
```

v-clicks

`v-clicks` 仅作为组件提供。它可以快速将其子元素全部添加 `v-click` 指令。它在列表中尤为实用。

```
<v-clicks>
  - Item 1
  - Item 2
  - Item 3
  - Item 4
</v-clicks>
```

每次你点击“下一步”按钮时，元素会逐条依次出现。

自定义点击数量

默认情况下，Slidev 会计算进入下一张幻灯片之前需要执行多少步。你可以在 frontmatter 选项中设置 `clicks` 来覆盖该设置：

```
---  
# 在你进入下一页之前，需要点击 10 次  
clicks: 10  
---
```

排序

通过传递点击索引，你可以自定义展示的顺序

```
<div v-click>1</div>  
<div v-click>2</div>  
<div v-click>3</div>
```

```
<!-- 顺序颠倒了 -->  
<div v-click="3">1</div>  
<div v-click="2">2</div>  
<div v-click="1">3</div>
```

```
---  
clicks: 3  
---  
  
<!-- 三次点击后可见 -->  
<v-clicks at="3">  
  <div>Hi</div>  
</v-clicks>
```

元素过渡

当你在元素中应用 `v-click` 指令时，它会给该元素添加名为 `slidev-vclick-target` 的类。当元素隐藏时，还加上了 `slidev-vclick-hidden` 类。例如：

```
<div class="slidev-vclick-target slidev-vclick-hidden">Text</div>
```

点击后，会变成：

```
<div class="slidev-vclick-target">Text</div>
```

默认情况下，这些类都应用了半透明的过渡效果：

```
// the default  
  
.slidev-vclick-target {  
  transition: opacity 100ms ease;  
}  
  
.slidev-vclick-hidden {
```

```
    opacity: 0;
    pointer-events: none;
}
```

你可以在你的自定义样式表中自定义过渡效果来覆盖默认配置。

例如，你可以通过以下方式实现缩放过渡效果：

```
// styles.css

.slidev-vclick-target {
  transition: all 500ms ease;
}

.slidev-vclick-hidden {
  transform: scale(0);
}
```

只为某页幻灯片或布局指定动画：

```
.slidev-page-7,
.slidev-layout.my-custom-layout {
  .slidev-vclick-target {
    transition: all 500ms ease;
  }

  .slidev-vclick-hidden {
    transform: scale(0);
}
```

欲了解更多详细信息，请参阅 [自定义样式](#)

运动

Slidev 内置了 `@vueuse/motion`。你可以对任何元素应用 `v-motion` 指令，以对它们施加运动效果。例如：

```
<div
  v-motion
  :initial="{ x: -80 }"
  :enter="{ x: 0 }">
  Slidev
</div>
```

文本 `Slidev` 将从其初始化位置 `-80px` 移至其原始位置。

注意：Slidev 会预加载下一张幻灯片以提高性能，这意味着动画可能会在你导航到该页面之前就开始了。为了使其正常工作，你可以禁用指定幻灯片的预加载

```
---  
preload: false  
---
```

或者使用 `v-if` 控制元素的生命周期来获得对组件更细粒度的控制

```
<div  
  v-if="$slidev.nav.currentPage === 7"  
  v-motion  
  :initial="{ x: -80 }"  
  :enter="{ x: 0 }">  
  Slidev  
</div>
```

学习模式：[Demo](#) | [@vueuse/motion](#) | [v-motion](#) | [Presets](#)

页面过渡

当前版本尚未提供对幻灯片页面过渡功能的内置支持。我们计划在下一个主版本中增加对其的支持。在此之前，你仍然可以使用自定义样式和工具库来实现页面过渡效果。

[Go to TOC](#)

绘图与批注

自 v0.23 起可用

我们基于 [drauu](#) 实现了绘图和批注的功能，可用进一步增强你的演示效果。

你可以通过点击工具栏上的 图标来启用。它也可以在 [演讲者模式](#) 中使用。你创建的绘图和批注都会实时自动同步起来。

与触控笔一同使用

当在平板电脑上使用触控笔时（例如，带有 Apple Pencil 的 iPad），Slidev 可以智能地检测输入类型。你可以直接用笔在幻灯片上绘图，而无需打开绘图模式，同时你的手指或鼠标可以控制导航。

对绘图进行持久化

frontmatter 中的配置可以把你得到绘图作为 SVG 保存在 `.slidev/drawings` 目录下，并把它们放入你导出的 pdf 或者部署的网站中。

```
---  
drawings:  
  persist: true  
---
```

禁用绘图

完全禁用：

```
---  
drawings:  
  enabled: false  
---
```

仅在开发环境可用：

```
---  
drawings:  
  enabled: dev  
---
```

仅在演讲者模式可用：

```
---  
drawings:  
  presenterOnly: true  
---
```

绘图同步

默认情况下， Slidev 会在所有实例中同步你的绘图。如果你在和他人共享幻灯片，你可能会需要通过以下方式禁用同步：

```
---  
drawings:  
  syncAll: false  
---
```

通过这个配置，只有来自演讲者实例的绘图会和其他实例同步。

编辑器整合

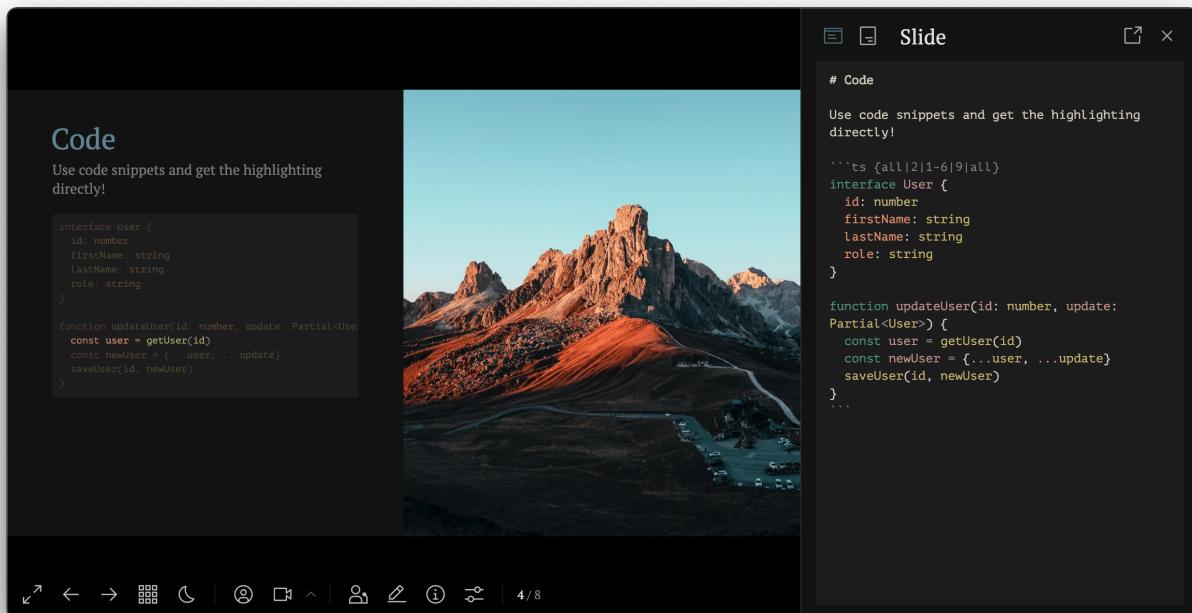
因为 Slidev 使用 Markdown 作为源文件，所以你可以使用任何你喜欢的编辑器。

如果你想对幻灯片进行一些高级管理，我们为你提供了以下编辑器集成！

编辑器集成

Slidev 带有一个集成的 [CodeMirror](#) 编辑器，可以立即重新加载并保存更改到文件中。

点击 按钮来打开它。



VS Code 插件



Slidev for VS Code

 VS Code Marketplace v0.4.1

downloads 26k

VS Code 插件提供一些特性，这些特性可以帮你更好的组织幻灯片并且可以快速浏览。

特性

- 在侧边面板中查看幻灯片
- 下一页 / 上一页按钮
- 对幻灯片重新排序
- 幻灯片折叠
- 将 Markdown 转为 HTML

The screenshot shows a code editor window titled "slides.md — demo". The left sidebar contains a tree view of files under "SLIDEV: SLIDES", including "Composable Vue", "Anthony Fu", "Sponsors", "Composable Vue", "VueUse", "Composition API", "Ref", "Ref Auto Unwrapping", "unref - Oppsite of Ref", "Patterns & Tips", "What's Composable Func...", "Think as "Connections"" (highlighted), "One Thing at a Time", "Passing Refs as Argumen...", "MaybeRef <MarkerTips/>", "Make it Flexible <MarkerP...", "'useTitle' <Marker class=...>", "'Reuse' Ref <MarkerCore...>", "'ref' / 'unref' <MarkerTi...>", "Object of Refs <MarkerPa...>", "Async to "Sync" <Marker...>", "'useFetch' <Marker clas...>", and "Side-effects Self Cleanup...". The main editor area displays the following code:

```
168 # Ref Auto Unwrapping <MarkerCore />
170
    Get rid of `value` for most of the time.
171
        <div class="grid grid-cols-2 gap-x-4">
172            <v-clicks :every='2'>
173                - `watch` accepts ref as the watch target, and returns the
174                    unwrapped value in the callback
175
176                ````ts
177
178        const counter = ref(0)
179
180        watch(counter, count => {
181            console.log(count) // same as `counter.value`
182        })
183
184        - Ref is auto unwrapped in the template
185
186        ````html
187
188        <template>
189            <button @click="counter += 1">
```

Markdown

[Go to TOC](#)

导出

PDF

导出为 PDF 或 PNG 的功能基于 [Playwright](#) 实现渲染。因此，使用此功能前需要安装 `playwright-chromium`。如果你需要在 CI 环境下进行导出，那么阅读 [playwright CI 指南](#) 会对你有所启发。

安装 `playwright-chromium`：

```
$ npm i -D playwright-chromium
```

接着，使用如下命令即可将你的幻灯片导出为 PDF：

```
$ slidev export
```

稍作等待，即可在 `./slides-export.pdf` 路径下看到你幻灯片的 PDF 文件。

如果你想要导出使用暗色主题的幻灯片，请使用 `--dark` 选项：

```
$ slidev export --dark
```

导出点击步骤

自 v0.21 起可用

默认情况下，Slidev 会将每张幻灯片导出为 1 页，并忽略点击动画。如果你想将多个步骤的幻灯片，分解为多个页面，请使用 `--with-clicks` 选项。

```
$ slidev export --with-clicks
```

PNGs

当为命令传入 `--format png` 选项时，Slidev 会将每张幻灯片导出为 PNG 图片格式。

```
$ slidev export --format png
```

单页应用(SPA)

请参阅 [静态部署](#) 章节。

[Go to TOC](#)

FAQ

Grids

由于 Slidev 基于 Web 运行，因此你可以使用任何想使用的布局方式。比如 [CSS Grids](#), [flexboxes](#), 甚至是 [Masonry](#), 都可以完美兼容。

由于我们内置了 [Windi CSS](#), 你也可以参考使用如下方式：

```
<div class="grid grid-cols-2 gap-4">
<div>

The first column

</div>
<div>

The second column

</div>
</div>
```

你甚至可以定制每一列的大小，比如：

```
<div class="grid grid-cols-[200px,1fr,10%] gap-4">
<div>

The first column (200px)

</div>
<div>

The second column (auto fit)

</div>
<div>

The third column (10% width to parent container)

</div>
</div>
```

欲了解更多，请参考 [Windi CSS 的 Grids 布局](#)。

定位

幻灯片被定义为固定尺寸（默认为 `980x552px`），并会跟随用户屏幕进行缩放。你可以安全地在你的幻灯片中使用绝对定位，因为它们会随着屏幕的缩放而变化。

例如：

```
<div class="absolute left-30px bottom-30px">
  This is a left-bottom aligned footer
</div>
```

如需改变 canvas 的实际尺寸，你可以在第一张幻灯片的 frontmatter 中传递 `canvasWidth` 选项：

```
---  
  canvasWidth: 800  
---
```

Font Size

如果你觉得幻灯片的字体过小，你可以通过如下方式进行调整：

覆盖本地样式

你可以通过内联的 `<style>` 标签来覆盖每张幻灯片的样式。

```
# Page 1
<style>
h1 {
  font-size: 10em;
}
</style>
---

# Page 2
This will not be affected.
```

了解更多：[内联样式](#)

覆盖全局样式

你可以通过创建 `./style.css` 文件的方式来提供自定义全局样式，例如：

```
/* style.css */
h1 {
  font-size: 10em !important;
}
```

了解更多：[全局样式](#)

Canvas 缩放

改变画布的实际尺寸将缩放所有内容（文本、图片、组件等）以及幻灯片。

```
---  
# default: 980
# 由于画布变小，视觉尺寸也会变大
  canvasWidth: 800
---
```

使用 Transform

我们提供了内置的 `<Transform />` 组件，它针对 CSS 的 transform 属性进行了简易封装。

```
<Transform :scale="1.4">  
  - Item 1  
  - Item 2  
</Transform>
```

[Go to TOC](#)

静态部署

构建单页应用(SPA)

你还可以将幻灯片构建成可部署的单页应用 (SPA) :

```
$ slidev build
```

生成的应用程序会保存在 `dist/` 目录下，然后你可以将该目录部署在 [GitHub Pages](#), [Netlify](#), [Vercel](#), 等你想部署的任何地方。接着，就可以将你幻灯片的链接分享给任何人。

配置基础路径

如果你需要将幻灯片部署在网站的子路由下，你可以使用 `--base` 选项来进行修改。例如：

```
$ slidev build --base /talks/my-cool-talk/
```

欲了解更多，请参阅 [Vite 的文档](#)。

提供可下载的 PDF

你可以向浏览幻灯片单页应用的观众提供一个可下载的 PDF。你可以通过如下配置来启用它：

```
---  
download: true  
---
```

配置好后，Slidev 将生成一个 PDF 文件，并在单页应用中展示下载按钮。

你也可以为 PDF 提供一个自定义的 URL。在这种情况下，PDF 的渲染过程将被忽略。

```
---  
download: 'https://myside.com/my-talk.pdf'  
---
```

示例

下面是几个导出为单页应用的示例：

- [Starter Template](#)
- [Composable Vue by Anthony Fu](#)

欲了解更多，请参阅 [Showcases](#)。

部署

我们建议使用 `npm init slidev@latest` 来为你初始化你的项目，它包含了部署服务开箱即用的配置文件。

Netlify

- [Netlify](#)

在你项目的根目录创建 `netlify.toml` 文件，其内容如下：

```
[build.environment]
  NODE_VERSION = "14"

[build]
  publish = "dist"
  command = "npm run build"

[[redirects]]
  from = "/"
  to = "/index.html"
  status = 200
```

接着，去 Netlify 的仪表盘，选择对应仓库并创建新的站点。

Vercel

- [Vercel](#)

在你项目的根目录创建 `vercel.json` 文件，其内容如下：

```
{
  "rewrites": [
    { "source": "/(.*)", "destination": "/index.html" }
  ]
}
```

接着，去 Vercel 的仪表盘，选择对应仓库并创建新的站点。

GitHub Pages

- [GitHub Pages](#)

将你的幻灯片部署到 GitHub Pages：

- 上传你的仓库里该项目的全部文件（例如，名字为 `name_of_repo`）
- 创建 `.github/workflows/deploy.yml` 文件，并包含如下内容。然后通过 Github Action 将你的幻灯片部署到 Github Pages。在该文件中，用 `name_of_repo` 替换 `<name_of_repo>`。

```
name: Deploy pages
on: push
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - name: Install dependencies
```

```
run: npm install
- name: Install slidev
  run: npm i -g @slidev/cli
- name: Build
  run: slidev build --base <name_of_repo>
- name: Deploy pages
  uses: crazy-max/ghaction-github-pages@v2
  with:
    build_dir: dist
  env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

- 在你的仓库里，选择“Settings > Pages”。在“Build and deployment”下，选择“Deploy from a branch”，选择“gh-pages”和“root”，点击保存。
- 最终，在全部工作流执行之后，在“Settings > Pages”下会出现幻灯片的链接。

开始使用

总览

Slidev (slide + dev, /slaidiv/) 是基于 Web 的幻灯片制作和演示工具。它旨在让开发者专注在 Markdown 中编写内容，同时拥有支持 HTML 和 Vue 组件的能力，并且能够呈现像素级完美的布局，还在你的演讲稿中内置了互动的演示样例。

它使用了功能丰富的 markdown 文件来生成精美的幻灯片，具有即时重载的体验。它还拥有很多内置的集成功能，如实时编码、导出 PDF、演讲录制等。由于 Slidev 是由 web 驱动的，因此你可以使用它进行任何操作 —— 具有无限的可能性。

你可以在 [为什么选 Slidev](#) 部分了解更多关于本项目的设计初衷。

功能

- 📝 **Markdown 支持** —— 使用你最喜欢的编辑器和工作流编写 Markdown 文件
- 💻 **开发者友好** —— 内置代码高亮、实时编码等功能
- 🎨 **可定制主题** —— 以 npm 包的形式共享、使用主题
- 🌈 **灵活样式** —— 使用 Windi CSS 按需使用的实用类和 UnoCSS
- 🤖 **可交互** —— 无缝嵌入 Vue 组件
- 🎙 **演讲者模式** —— 可以使用另一个窗口，甚至是你的手机来控制幻灯片
- 🖌 **绘图** - 在你的幻灯片上进行绘图和批注
- TeX **LaTeX 支持** —— 内置了对 LaTeX 数学公示的支持
- 📊 **图表支持** —— 使用文本描述语言创建图表
- ⭐ **图标** —— 能够直接从任意图标库中获取图标
- 💻 **编辑器** —— 集成的编辑器，或者使用 VS Code 扩展
- 🎥 **演讲录制** —— 内置录制功能和摄像头视图
- 🌐 **跨平台** —— 能够导出 PDF、PNG 文件，甚至是一个可以托管的单页应用
- ⚡ **快速** —— 基于 Vite 的即时重载
- 🔧 **可配置** —— 支持使用 Vite 插件、Vue 组件以及任何的 npm 包

技术栈

Slidev 使用了如下的工具和技术：

- Vite —— 一款极速的前端工具
- 基于 Vue 3 的 Markdown —— 专注内容的同时，具备 HTML 和 Vue 组件的能力
- Windi CSS 或 UnoCSS —— 按需、实用类优先的 CSS 框架，轻松定制你的幻灯片样式
- Prism, Shiki, Monaco Editor —— 具有实时编码能力的一流代码片段支持
- RecordRTC —— 内置录制功能和摄像头视图
- VueUse 家族 —— `@vueuse/core`、`@vueuse/head`、`@vueuse/motion` 等
- Iconify —— 图标库集合
- Drauu - 支持绘图和批注
- KaTeX —— LaTeX 数学渲染
- Mermaid —— 文本描述语言创建图表

搭建你的第一个演讲稿

在线试用

[sli.dev/new](#)



[Open in StackBlitz](#)

本地创建

使用 NPM：

```
$ npm init slidev
```

使用 Yarn：

```
$ yarn create slidev
```

根据提示开始创建你的幻灯片吧！想要了解更多关于 Markdown 的语法，请阅读 [语法指南](#)。

命令行界面

在安装了 Slidev 的项目里，你可以在你的 npm scripts 里使用 `slidev` 命令。

```
{
  "scripts": {
    "dev": "slidev", // 启动 dev server
    "build": "slidev build", // 构建生产环境的单页面应用
    "export": "slidev export" // 将幻灯片导出为 pdf 格式
  }
}
```

或者，你也可以使用 `npx`

```
$ npx slidev
```

执行 `slidev --help` 命令获取更多选项的详细信息。

Markdown 语法

Slidev 会读取位于项目根目录的 `slides.md` 文件，并将其转换为幻灯片。每当你修改 Markdown 文件，幻灯片的内容都会立刻随之更新。例如：

```
# Slidev
Hello World
---
# 第 2 页
```

直接使用代码块，能够实现代码高亮

```
//```ts
console.log('Hello, World!')
//```
```

第 3 页

请阅读 [语法指南](#) 获取更多关于 Slidev Markdown 语法的内容。

[Go to TOC](#)

安装

初始模板

Slidev 需要 [Node.js](#) 的版本 **>=14.0.0**

快速开始最好的方式就是使用官方的初始模板。

使用 NPM :

```
$ npm init slidev@latest
```

使用 Yarn :

```
$ yarn create slidev
```

跟随命令行的提示，它将自动为你打开幻灯片，网址是 <http://localhost:3030/>。

同时包含了一些基本配置和简单的 demo，为你说明如何开始使用 Slidev。

手动安装

如果你倾向于手动安装 Slidev，或者想把它集成到你已有的项目中，你可以执行如下操作：

```
$ npm install @slidev/cli @slidev/theme-default
```

```
$ touch slides.md
```

```
$ npx slidev
```

请注意，如果你使用的是 [pnpm](#)，请先启用 `shamefully-hoist` 选项，才能使得 Slidev 正常工作。

```
$ echo 'shamefully-hoist=true' >> .npmrc
```

全局安装

自 v0.14 开始可用

你可以使用如下命令在全局安装 Slidev：

```
$ npm i -g @slidev/cli
```

然后即可在任何地方使用 `slidev`，而无需每次都创建一个项目。

```
$ slidev
```

如果在本地的 `node_modules` 目录下找到了 `@slidev/cli`，此命令也同样有效。

在 Docker 上安装

如果你需要快速的在容器上部署你的演示文稿，你可以使用由 `tangramor` 维护的预构建 `docker` 镜像，或者自行构建。

在你的工作目录下运行下面的命令：

```
docker run --name slidev --rm -it \
--user node \
-v ${PWD}:/slidev \
-p 3030:3030 \
tangramor/slidev:latest
```

如果你的工作目录为空，容器会在目录下自动创建 `slides.md` 文件和其它相关文件，并基于 `3030` 端口启动 `slidev` 服务。

你可以通过 `http://localhost:3030/` 访问你的幻灯片。

构建可部署镜像

你也可以把你的 `slidev` 幻灯片构建到一个 `docker` 镜像里来进行部署，`Dockerfile` 如下：

```
FROM tangramor/slidev:latest
ADD . /slidev
```

使用命令 `docker build -t myppt .` 来构建镜像。

执行 `docker run --name myslides --rm --user node -p 3030:3030 myppt` 命令来运行镜像。

这时你就可用通过 `http://localhost:3030/` 来打开你的幻灯片了。

构建单网页应用

在前面启动的 `slidev` 容器上运行命令 `docker exec -i slidev npx slidev build` 就可以在 `dist` 目录下将你的幻灯片生成静态 HTML 文件。

使用 Github Pages 托管

你可以在静态 Web 站点上托管生成的静态文件，比如 [Github pages](#) 或 [Gitlab pages](#)。

由于 `Github pages` 的 URL 可能包含二级目录，所以你需要修改生成的 `index.html`，把 `href="/assets/xxx"` 改为 `href=". /assets/xxx"`（即使用相对路径）。或者你可以用 `vite` 的 `--base=/<subfolder>/` 选项来指定二级目录，例如：`docker exec -i slidev npx slidev build --base=/slidev_docker/`。

为了防止触发 `Jekyll` 构建流程，你需要在静态站根目录下添加一个名为 `.nojekyll` 的空文件

使用 docker 托管

你当然也可以使用 docker 容器来托管生成的静态文件：

```
docker run --name myslides --rm -p 80:80 -v ${PWD}/dist:/usr/share/nginx/html
nginx:alpine
```

或者使用下面的 Dockerfile 来构建一个静态站点的容器镜像：

```
FROM nginx:alpine
COPY dist /usr/share/nginx/html
```

运行 `docker build -t mystaticppt .` 来构建镜像

执行 `docker run --name myslides --rm -p 80:80 mystaticppt` 命令来启动容器。

此时你就可以通过 `http://localhost/` 来访问你的幻灯片了。

关于容器的更多详细信息, 请参考 [tangramor/slides_docker 仓库](#)。

Command Line Interface (CLI)

`@slidev/cli` 暴露了一些命令, 你可以通过 `npx slidev ...` 或者在你的 `package.json` 中添加 `script` 来使用它们。

```
{
  "script": {
    "dev": "slidev"
  }
}
```

在这种情况下, 你可以通过 `npm run dev` 来运行。

你可以向任何命令传参：

- `boolean` 选项如果存在则为 `true`, 否则为 `false` (例如: `slidev --open`)
- 一些选项可以在选项后添加数值, 或者使用 `=` 字符 (例如: `slidev --port 8080` 或者 `slidev --port=8080`)

如果你使用 npm 的 `script`, 别忘了在 npm 命令后加上 `--` :

```
npm run slidev -- --open
```

slidev [entry] {#slidev-entry}

为 Slidev 启动一个本地服务器。

- `[entry] (string, 默认值: slides.md)`: 幻灯片 markdown 的入口文件。

选项：

- `--port, -p (number, 默认值: 3030)`: 端口号。
- `--open, -o (boolean, 默认值: false)`: 在浏览器打开。

- `--remote [password] (string)` : 监听公共主机并启用远程控制, 如果传递了该值, 那么演讲者模式是私有的, 只有通过在 URL 查询 `password` 参数中给定的密码才能访问。
- `--log ('error', 'warn', 'info', 'silent', 默认值: 'warn')` : 日志级别。
- `--force, -f (boolean, 默认值: false)` : 强制优化器忽略缓存, 并重新构建。
- `--theme, -t (string)` : 覆盖主题。

slidev build [entry] {#slidev-build-entry}

建立可托管的 SPA。

- `[entry] (string, 默认值: slides.md)` : 幻灯片 markdown 的入口文件。

选项：

- `--watch, -w (boolean, 默认值: false)` : 构建观察。
- `--out, -o (string, 默认值: dist)` : 要输出到的目标文件夹。
- `--base (string, 默认值: /)` : base URL (参阅 <https://cli.vuejs.org/config/#publicpath>)
- `--download (boolean, 默认值: false)` : 允许在 SPA 内下载 PDF 格式的幻灯片。
- `--theme, -t (string)` : 覆盖主题。

slidev export [entry] {#slidev-export-entry}

将幻灯片导出为 PDF (或者其他格式)。

- `[entry] (string, 默认值: slides.md)` : 幻灯片 markdown 的入口文件。

Options:

- `--output (string, 默认值: use exportFilename (参阅 https://sli.dev/custom/#frontmatter-configures)) 或使用 [entry]-export)` : 输出的路径。
- `--base ('pdf', 'png', 'md', 默认值: 'pdf')` : 输出的格式。
- `--timeout (number, 默认值: 30000)` : 渲染打野页面的超时时间 (参阅 <https://playwright.dev/docs/api/class-page#page-goto>)。
- `--range (string)` : 输出的页面范围 (例如: '1,4-5,6')。
- `--dark (boolean, 默认值: false)` : 导出黑暗主题色。
- `--with-clicks, -c (boolean, 默认值: false)` : 输出每次点击的页面 (参阅 <https://sli.dev/guide/animations.html#click-animations>)。
- `--theme, -t (string)` : 覆盖主题。

slidev format [entry] {#slidev-format-entry}

格式化 markdown 文件。

- `[entry] (string, 默认值: slides.md)` : 幻灯片 markdown 的入口文件。

slidev theme [subcommand] {#slidev-theme-subcommand}

与主题相关的业务。

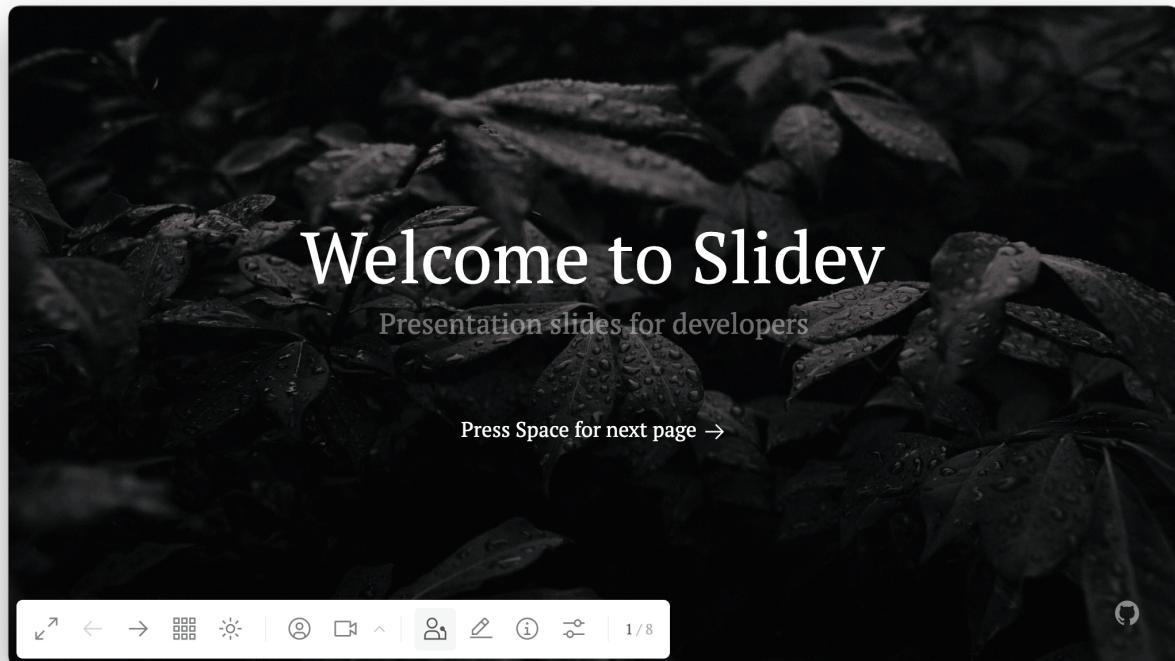
子命令：

- `eject [entry]` : 将当前主题弹出到本地文件系统中
 - `[entry] (string, 默认值: slides.md)` : 幻灯片 markdown 的入口文件。
 - 选项: `--dir (string, 默认值: theme)` : 要输出到的目标文件夹。 `--theme`, `-t (string)` : 覆盖主题。

导航

导航栏

将鼠标移至窗口左下角以显示导航栏。

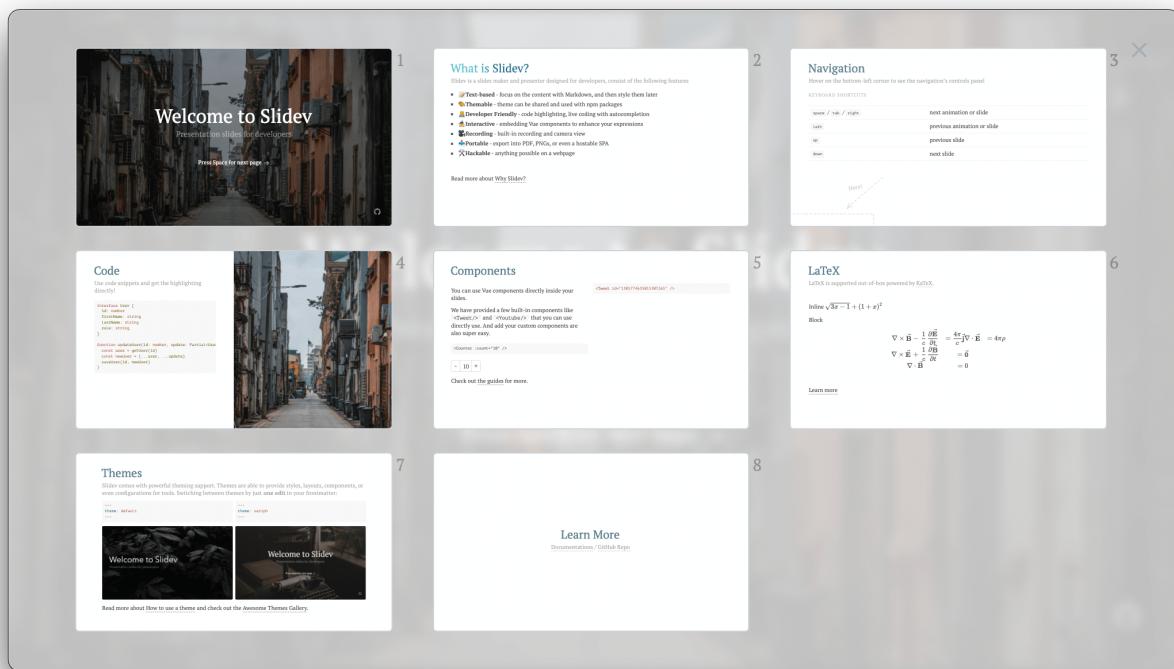


快捷键	按钮	说明
f		切换全屏
right / space		下一动画或幻灯片
left		上一动画或幻灯片
up	-	上一张幻灯片
down	-	下一张幻灯片
o		切换 幻灯片总览
d		切换 暗黑模式
-		切换 摄像头视图
-		演讲录制
-		进入 演讲者模式
-		切换 集成编辑器

快捷键	按钮	说明
-		下载幻灯片 (仅在 单页 (SPA) 构建 中支持)
-		显示该演示文稿的信息
-		显示设置菜单
g	-	显示“前往...”

幻灯片总览

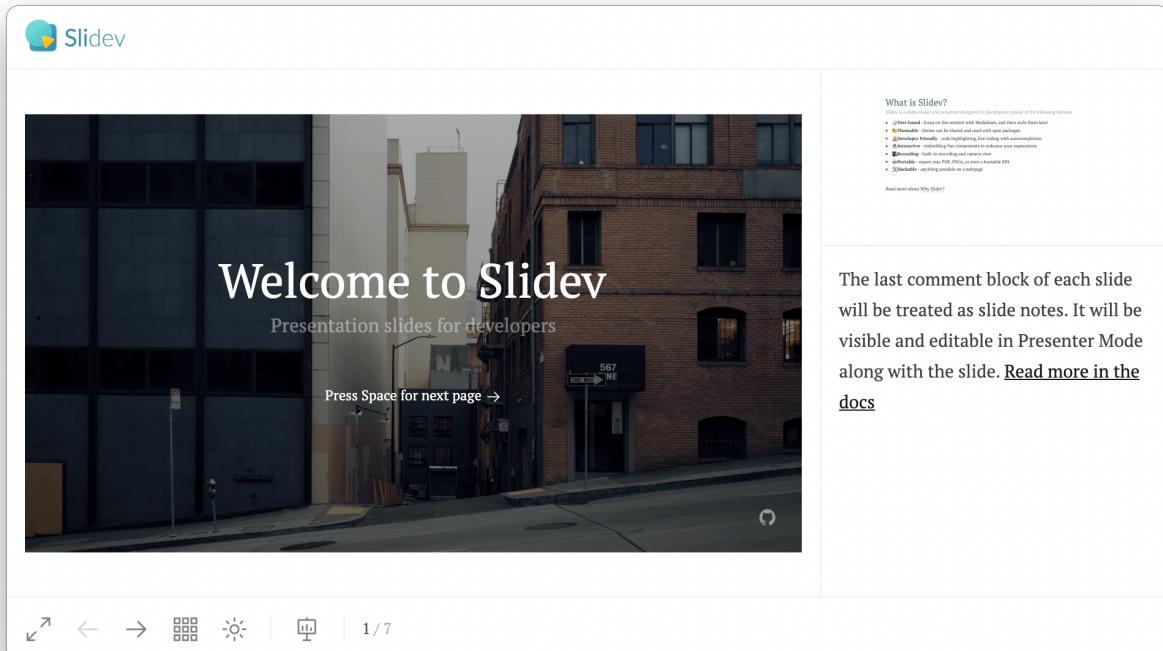
通过按下键盘按键 o 或者鼠标点击导航栏中的 按钮，你就可以查看全部幻灯片并方便地在它们之间跳转。



[Go to TOC](#)

演讲者模式

点击导航面板上的 按钮，或者手动访问 `http://localhost:3030/presenter` 地址，即可进入演讲者模式。每当你进入演示者模式，其他页面实例会自动与演示者保持同步。



[Go to TOC](#)

演讲录制

Slidev 拥有内置的演讲录制和摄像头视图。你可以使用它们轻松实现你的演讲录制。

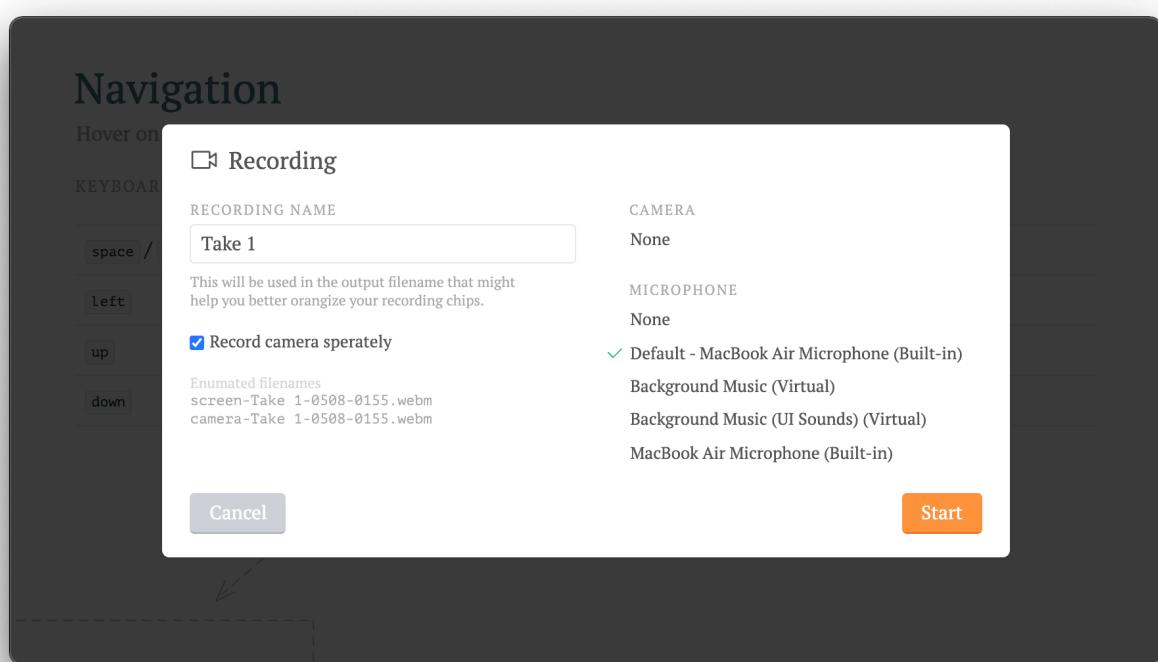
摄像头视图

点击导航面板上的 按钮，将在演示文稿中显示你的摄像头视图。你可以拖动它，并使用右下角的把手来调整大小。尺寸和位置将持久化存储在 `localStorage` 中，因此，可以保证多次刷新后的展示一致，无需担心位置和大小丢失的问题。

录制

点击导航面板上的 按钮，将会弹出一个对话框。在此对话框中，你可以选择将你的摄像头视图嵌入到幻灯片中进行录制，也可以将它们分成两个视频文件。

此功能的实现得益于 [RecordRTC](#) 与 [WebRTC API](#)。



[Go to TOC](#)

Markdown 语法

幻灯片通过 **一个 markdown 文件** 编写而成 (默认会使用 `./slides.md`)。

你可以像平时编写 markdown 一样使用 [Markdown 的相关特性](#), 同时还支持内联的 HTML 和 Vue 组件。也支持使用 [Windi CSS](#) 来编写样式。使用 `---` 添加分隔符来分隔你的幻灯片。

```
# Slidev

Hello, World!

---

# Page 2

Directly use code blocks for highlighting

//```ts
console.log('Hello, World!')
//```

---

# Page 3

You can directly use Windi CSS and Vue components to style and enrich your slides.

<div class="p-3">
  <Tweet id="20" />
</div>
```

扉页及布局

你可以通过将分隔符转换为 [扉页块 \(front matter\)](#), 为每张幻灯片指定布局 (layout) 和其他元数据。每个扉页信息都以分隔符 `---` 开始, 以另一个分隔符 `---` 结束。两个分隔符之间的文本是 [YAML](#) 格式的数据对象。具体示例如下:

```
---
layout: cover
---

# Slidev

This is the cover page.

---
layout: center
background: './images/background-1.png'
class: 'text-white'
---

# Page 2

This is a page with the layout `center` and a background image.

---
```

```
# Page 3
```

```
This is a default page without any additional metadata.
```

欲了解更多，请参阅 [自定义](#) 章节。

代码块

建立 Slidev 一个非常重要的原因就是为了让代码在幻灯片中拥有正确的高亮。如你所见，你可以使用 Markdown 风格的代码块，以使得你的代码高亮。

```
//``ts
console.log('Hello, World!')
//``
```

我们支持 [Prism](#) 和 [Shiki](#) 作为语法高亮器。请参阅 [语法高亮器](#) 获取更多细节。

特定行高亮

如需针对特定行进行高亮展示，只需在 `{}` 内添加对应的行号。行号从 1 开始计算。

```
//``ts {2,3}
function add(
  a: Ref<number> | number,
  b: Ref<number> | number
) {
  return computed(() => unref(a) + unref(b))
}
//``
```

如果要在多个步骤中改变高亮，你可以用 `|` 分隔它们。比如：

```
//``ts {2-3|5|all}
function add(
  a: Ref<number> | number,
  b: Ref<number> | number
) {
  return computed(() => unref(a) + unref(b))
}
//``
```

这段代码会先对 `a: Ref<number> | number` 和 `b: Ref<number> | number` 进行高亮展示，当你点击幻灯片后，会高亮展示 `return computed(() => unref(a) + unref(b))`，最后，会对整个块进行高亮展示。你可以在 [动画指南](#) 中了解更多。

你可以使用行号 `0` 来跳过高亮。比如：

```
//``ts {0}
function add(
  a: Ref<number> | number,
  b: Ref<number> | number
) {
  return computed(() => unref(a) + unref(b))
}
//``
```

如果代码在一张幻灯片展示不下，你可以传递一个额外的 `maxHeight` 选项，该选项将为代码段设置固定高度并启用滚动：

```
//``ts {2|3|7|12} {maxHeight:'100'}
function add(
  a: Ref<number> | number,
  b: Ref<number> | number
) {
  return computed(() => unref(a) + unref(b))
}
/// ...as many lines as you want
const c = add(1, 2)
//``
```

Monaco 编辑器

当你需要在演示文稿中做修改时，只需在语言 id 后添加 `{monaco}` —— 即可将该代码块变为一个功能齐全的 Monaco 编辑器。

```
//``ts {monaco}
console.log('HelloWorld')
//``
```

欲了解更多，请参阅 [配置 Monaco](#)。

内联样式

你可以在 Markdown 中直接使用 `<style>` 标签来覆盖当前幻灯片的样式。

```
# This is Red

<style>
h1 {
  color: red
}
</style>

---

# Next slide is not affected
```

Markdown 中的 `<style>` 标签均为 `scoped`。如果想覆盖全局样式，请查阅 [项目结构](#)

在 [Windi CSS](#) 的支持下，你可以直接使用嵌套的 CSS 和 [指令集](#)。(例如，`@apply`)

```
# Slides

> Hello `world`

<style>
blockquote {
  code {
    @apply text-teal-500 dark:text-teal-400;
  }
}
</style>
```

静态资源

和编写 Markdown 的方式一样，你可以使用本地或远程的 URL 的图片。

如果是远程资源，内置的 `vite-plugin-remote-assets` 将在第一次运行时把它们缓存到磁盘中，即便是大图也能实现立即加载。

 ! [Remote Image](https://sli.dev/favicon.png)

如果是本地资源，请将资源放置到 `public` 文件夹 中并使用 / 开头的 URL 来引用它们。

 ! [Local Image](pic.png)

如果你想使用自定义的尺寸或样式，可以使用 `` 标签



备注

你也可以为每张幻灯片编写备注。它们将展示在 [演讲者模式](#) 中，供你在演示时参考。

在 Markdown 中，每张幻灯片中的最后一个注释块将被视为备注。

```
---
layout: cover
---

# 第 1 页

This is the cover page.

<!-- 这是一条备注 -->

---

# 第 2 页

<!-- 这不是一条备注，因为它在幻灯片内容前 -->

The second page

<!--
这是另一条备注
-->
```

图标

Slidev 允许你在 Markdown 中直接访问几乎所有的开源的图标集。这得益于 `unplugin-icons` 和 [Iconify](#)。

图标 ID 遵循 [Iconify](#) 的命名规则 `{collection-name}-{icon-name}`。例如：

- 使用 [Material Design Icons](#)，其规则为 `<mdi-account-circle />` -
- 使用 [Carbon](#)，其规则为 `<carbon-badge />` -

- 使用 [Unicons Monochrome](#), 其规则为 `<uim-rocket />` -
- 使用 [Twemoji](#), 其规则为 `<twemoji-cat-with-tears-of-joy />` -
- 使用 [SVG Logos](#), 其规则为 `<logos-vue />` -
- 还有更多...

你可以通过 [Icônes](#) 来浏览访问所有可用的图标。

调整图标样式

你可以像其他 HTML 元素一样对图标的样式进行修改。例如：

```
<uim-rocket />
<uim-rocket class="text-3xl text-red-400 mx-2" />
<uim-rocket class="text-3xl text-orange-400 animate-ping" />
```

插槽

自 v0.18 开始可用

一些布局可以使用 [Vue](#) 的具名插槽。

例如，在 `two-cols` 布局中，你可以采用左（`default` 插槽）右（`right` 插槽）两列的布局方式。

```
---  
layout: two-cols  
---  
  
<template v-slot:default>  
  # Left  
  
  This shows on the left  
  
</template>  
<template v-slot:right>  
  # Right  
  
  This shows on the right  
</template>
```

Left

This shows on the left

Right

This shows on the right

我们还为具名插槽提供了一个语法糖 `:::name:::`。下述示例与上述示例的工作原理完全相同。

```
---
layout: two-cols
---

# Left

This shows on the left

::right::

# Right

This shows on the right
```

你也可以明确的指定默认插槽，并按自定义顺序展示。

```
---
layout: two-cols
---

::right::

# Right

This shows on the right

::default::

# Left

This shows on the left
```

配置

依赖的所有配置都可以在 Markdown 文件中定义，比如：

```
---
theme: serif
layout: cover
background: 'https://source.unsplash.com/1600x900/?nature,water'
---

# Sliderv

This is the cover page.
```

欲了解更多，请参阅 [扉页配置](#) 章节。

LaTeX

Slidev 开箱即有对 LaTeX 的支持，得益于 [KaTeX](#)。

内联

在你的 LaTeX 语法左右各加一个 `$`，用于内联渲染。

```
$\sqrt{3x-1} + (1+x)^2$
```

块

当使用两个 (`$$`) 时，会进行块级渲染。这种模式会使用更大的符号，并将结果居中。

```
$$
\begin{array}{c}
\nabla \times \vec{\mathbf{B}} - \frac{1}{c} \nabla \frac{\partial \vec{\mathbf{E}}}{\partial t} \\
= \frac{4\pi}{c} \vec{c} \cdot \vec{\mathbf{E}} \quad \nabla \cdot \vec{\mathbf{E}} = 4\pi \rho \\
\\
\nabla \times \vec{\mathbf{E}} + \frac{1}{c} \nabla \frac{\partial \vec{\mathbf{B}}}{\partial t} \\
= \vec{\mathbf{0}} \\
\end{array}
$$
```

了解更多：[Demo](#) | [KaTeX](#) | [markdown-it-katex](#)

图表

你也可以在 Markdown 的文本描述中创建图形或图表，得益于 [Mermaid](#)。

被标记为 `mermaid` 的代码块将被转换为图形，例如：

```
//```mermaid
sequenceDiagram
    Alice->John: Hello John, how are you?
    Note over Alice,John: A typical interaction
//````
```

你可以向它传递一个配置项来指定缩放和主题。该对象的语法是 JavaScript 的对象字面量，你需要对字符串添加引号 (`'`)，并在键与键之间使用(`,`)。

```
//```mermaid {theme: 'neutral', scale: 0.8}
graph TD
    B[Text] --> C{Decision}
    C -->|One| D[Result 1]
    C -->|Two| E[Result 2]
//````
```

了解更多：[Demo](#) | [Mermaid](#)

多个入口点

自 v0.15 开始可用

这意味着你可以将 `slides.md` 分割成多个文件，并可以按照你的需求组织它们。

`slides.md` :

```
# Page 1
This is a normal page
---
src: ./subpage2.md
---
<!-- this page will be loaded from './subpage2.md' -->
Inline content will be ignored
```

`subpage2.md` :

```
# Page 2
This page is from another file
```

合并 Frontmatter

你可以为主入口点和外部 markdown 页面提供 frontmatter。如果其中有相同的 key，**主入口点的 key 拥有更高的优先级**。例如：

`slides.md` :

```
---
src: ./cover.md
background: https://sli.dev/bar.png
class: text-center
---
```

`cover.md` :

```
---
layout: cover
background: https://sli.dev/foo.png
---
# Cover
Cover Page
```

其效果最终与下述页面相同：

```
---
layout: cover
background: https://sli.dev/bar.png
class: text-center
---
# Cover
Cover Page
```

页面复用

有了多入口点的加持，对页面进行重用变得很容易。例如：

```
---  
src: ./cover.md  
---  
  
---  
src: ./intro.md  
---  
  
---  
src: ./content.md  
---  
  
---  
# reuse  
src: ./content.md  
---
```

[Go to TOC](#)

为什么选择 Slidev

有很多功能丰富的、通用的、所见即所得的幻灯片制作工具，例如 [微软 PowerPoint](#) 或 [苹果 Keynote](#)。它们在制作带有动画、图表和许多其他漂亮的幻灯片方面效果相当好，同时非常直观和容易学习。那么，为什么要费心制作 Slidev 呢？

Slidev 旨在为开发者提供灵活性和交互性，通过使用他们已经熟悉的工具和技术，使他们的演示文稿更加有趣、更具表现力和吸引力。

当使用所见即所得的编辑器时，人们很容易被样式选项所干扰。Slidev 通过分离内容和视觉效果来弥补这一点。这使你能够一次专注于一件事，同时也能够重复使用社区中的主题。Slidev 并不寻求完全取代其他幻灯片制作工具。相反，它专注于迎合开发者社区的需求。

Slidev



以下是 Slidev 的一些最酷的功能：

支持 Markdown 语法

Slidev 使用一种扩展的 Markdown 格式，在一个纯文本文件中存储和组织你的幻灯片。这让你专注于制作内容。而且由于内容和样式是分开的，这也使得在不同的主题之间切换变得更加容易。

欲了解更多，请参阅 [Slidev 的 Markdown 语法](#)。

可定制主题

Slidev 的主题可以通过 `npm` 包的形式来分享和安装。你只需要使用一行配置就可以应用它们。

点击查看 [主题库](#) 或者 [学习如何制作一个主题](#)。

对开发者友好

Slidev 为开发者提供了一流的代码片段支持。它同时支持 `Prism` 和 `Shiki` 以获得像素级的完美语法高亮，并且能够随时修改代码。通过内置的 `Monaco 编辑器`，它还能让你在演示文稿中进行现场编码/演示，并支持自动补全、类型悬停、甚至是 `TypeScript` 类型检查。

欲了解更多，请参阅 [语法高亮](#) 和 [Monaco 配置](#)。

快速

Slidev 得益于 `Vite`, `Vue 3` 和 `Windi CSS`, 为你带来了最美妙的创作体验。你所做的每一个改变都会立即反映到你的幻灯片上。

查找更多关于 [技术栈](#)

互动性 & 直观表达

你可以编写自定义的 `Vue` 组件并直接在你的 `MarkDown` 文件中使用它们。你也可以在演示文稿中与它们互动，以更深入和直观的方式表达你的想法。

支持录制

Slidev 提供了内置的录音和摄像头视图。你可以将你的演示文稿与你的相机视图一起分享，或者为你的屏幕和相机分别录制并保存。所有这些都是内置的，不需要额外的工具。

欲了解更多，请参阅 [录制](#)。

可移植性

用一个命令就可以将你的幻灯片导出为 `PDF` 或 `PNG`，甚至是可托管的单页应用程序（`SPA`），并在任何地方分享它们。

欲了解更多，请参阅 [导出文档](#)。

可配置

由于 Slidev 基于 Web 技术，任何可以在 Web 应用中完成的事情，Slidev 也可以做到。例如，`WebGL`、`API 请求`、`ifrarnes`，甚至是实时共享。完全取决于你的想象力！

即刻体验

开始一个 Slidev 项目无需长篇大论。只需要一个命令即可：

```
$ npm init slidev
```

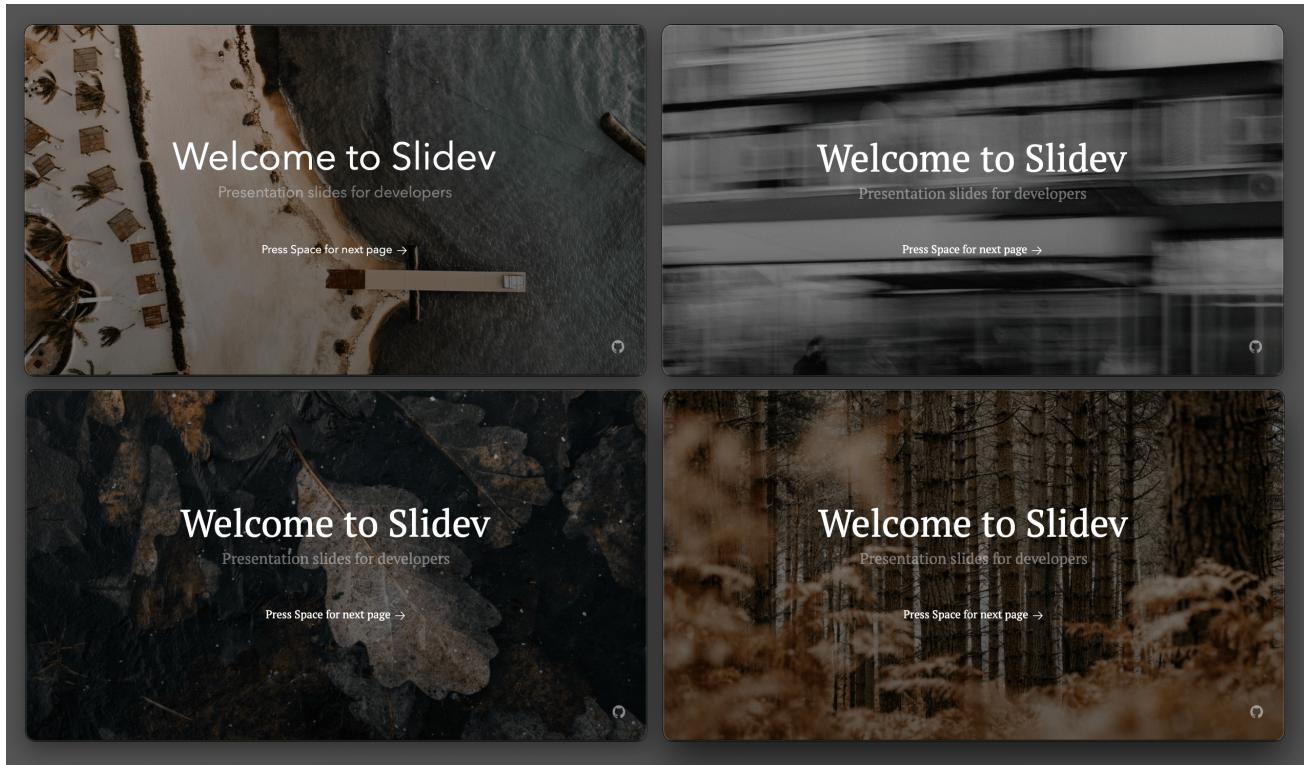
或者你也可以通过下方的视频进行快速预览：

Sliderv First Preview Demo



精选封面

我们精心挑选了一些封面图片，用以展示在我们的初始模板中。



来自精选集的随机图片

`background: https://source.unsplash.com/collection/94734566/1920x1080`

如果你喜欢它们的设计，可以参阅我们的 [Unsplash 系列](#)，同时还能了解它们的作者。

[Go to TOC](#)

学习资源

英语

视频

Slidev - one of the best presentation software and it is free!



文章

- [Tips To Turn R Markdown Into Slidev Presentation](#) by Hiroaki Yutani

中文

- [Slidev : 一个用Markdown写slides的神器](#) by 梦里风林
- [神器！这款开源项目可以让你使用 Markdown 来做 PPT！](#) by [Github掘金计划](#)
- [【用 markdown 写 Slide!】神器 Slidev 的安装及 bug 解决](#) by HaloHoohoo

日语

- [開発者のためのスライド作成ツール Slidev がすごい](#) by ryo_kawamata
- [Markdownでオシャレなスライドを作るSli.dev](#) by Nobuko YAMADA

[Go to TOC](#)

案例展示

使用 Slidev 制作的演示文档/演讲。

[Go to TOC](#)

主题合集

此处你将看到超棒的主题合集。

阅读更多关于 [如何使用主题](#), 或 [如何编写属于你自己的主题](#) 并与社区分享 !

官方主题

社区主题

下列是精选的社区主题。

更多主题

查找 [NPM 上所有可用的主题](#)。

使用主题

在 Slidev 中更换主题非常简单。在 frontmatter 中添加 `theme:` 配置即可。

```
---  
theme: serif  
---
```

在服务启动后，它会自动提示你是否安装该主题：

```
? The theme "@slidev/theme-serif" was not found in your project, do you want to install it now? › (Y/n)
```

或者你也可以手动安装：

```
$ npm install @slidev/theme-serif
```

现在你就可以使用全新的主题了！你可以查阅相应主题的 README 文档以了解更多的使用细节。

想要分享你自己的主题吗？来看看如何 [编写主题](#)。

弹出主题

如果你想对当前的主题拥有完全的掌控，你可以将它 弹出（eject）到本地的文件系统，并且随心所欲地修改它。可以使用以下命令：

```
$ slidev theme eject
```

它会将你当前使用的主题弹出到 `./theme` 目录下，然后请将你的 frontmatter 修改为：

```
---  
theme: ./theme  
---
```

如果你想在现有的主题上制作主题，这样会很方便。当然如果你这么做了，记得标明原主题和作者哦：）

本地主题

通过上面的描述，你可能已经发现了：在一个项目里是可以使用本地主题的。你只需要在主题说明中引入相对路径：

```
---  
theme: ./path/to/theme  
---
```

欲了解更多详细信息，请参阅 [编写主题](#)。

[Go to TOC](#)

编写主题

首先，我们推荐你使用预设的生成器来快速搭建一个主题：

```
$ npm init slidev-theme
```

之后便可以尝试对主题进行改动并使用。你也可以参阅 [官方主题](#) 中的案例。

主题能力

一个主题可以自定义以下功能：

- 全局样式
- 提供默认配置（字体、配色方案、语法高亮器等）
- 自定义布局或者重写现有布局
- 自定义组件或者重写现有组件
- 扩展 Windi CSS 配置
- 配置 Monaco, Prism 等工具

约定

主题发布到 npm，需遵循以下约定：

- 包名应该以 `slidev-theme-` 开头，例如：`slidev-theme-awesome`
- 在主题 `package.json` 的 `keywords` 中添加 `slidev-theme` 和 `slidev` 关键词

配置说明

如果想要测试自己编写的主题，你可以新建 `example.md` 并在 `frontmatter` 中增加以下代码，以告知 Slidev 你正在使用当前目录作为一个主题。

```
---  
theme: ./  
---
```

你还可以在 `package.json` 增加一些脚本以方便测试：

```
// package.json
{
  "scripts": {
    "dev": "slidev example.md",
    "build": "slidev build example.md",
    "export": "slidev export example.md",
    "screenshot": "slidev export example.md --format png"
  }
}
```

你只需在命令行中执行 `npm publish` 就可以发布自己的主题，并不需要额外的构建过程（这意味着你可以直接发布 `.vue` 和 `.ts` 文件，Slidev 可以直接识别它们）。

主题可以定制的范围与本地自定义相一致，可以参阅 [自定义文档](#)。

默认配置

自 v0.19 起可用

主题可以通过 `package.json` 提供默认的 [配置](#)

```
// package.json
{
  "slidev": {
    "default": {
      "aspectRatio": "16/9",
      "canvasWidth": 980,
      "fonts": {
        "sans": "Robot",
        "mono": "Fira Code"
      }
    }
  }
}
```

字体将从 [Google Fonts](#) 自动导入。

欲了解更多，请参阅 [fonts](#) 和 [frontmatter](#) 配置

主题元数据

配色方案

默认情况下，Slidev 假定主题会同时支持亮色与暗色两种模式。如果希望自己的主题只以某种预设的配色方案展现，你需要在 `package.json` 中显式指定：

```
// package.json
{
  "name": "slidev-theme-my-cool-theme",
  "keywords": [
    "slidev-theme",
    "slidev"
  ],
  "slidev": {
    "colorSchema": "light" // 还可选 "dark" 或 "both"
  }
}
```

当在编写自己的主题样式时，如果需要设置暗色模式下的样式，你可以将特定使用在暗色模式下的样式放置在指定的 `dark` 类下：

```
/* 共通 CSS 样式 */

html:not(.dark) {
    /* 亮色模式 CSS 样式 */
}

html.dark {
    /* 暗色模式 CSS 样式 */
}
```

在切换为暗色模式时 `Slidev` 会为页面中的 `html` 元素添加 `dark` 类。

语法高亮器

主题中也可以设置代码高亮配色，我们同时支持 `Prism` 和 `Shiki`。欲了解更多，请参阅 [语法高亮文档](#)。

你可以选择使用其中任意一种或同时使用。可以参考默认主题配置示例中的 `./styles/code.css` 和 `./setup/shiki.ts`。

另外，不要忘记在 `package.json` 中指定想要支持的高亮工具：

```
// package.json
{
    "slidev": {
        "highlighter": "shiki" // or "prism" or "all"
    }
}
```

Slidev 版本

如果主题依赖于 `Slidev` 的某项新特性，你可以为主题设置最小的 `Slidev` 版本，以使你的主题可以正常工作：

```
// package.json
{
    "engines": {
        "slidev": ">=0.19.3"
    }
}
```

如果用户使用的是旧版本的 `Slidev`，将会抛出错误。

Colophon

This book is created by using the following sources:

- Slidev - 简体中文
- GitHub source: [slidevjs/docs-cn](https://github.com/slidesjs/docs-cn)
- Created: 2022-11-27
- Bash v5.2.2
- Vivliostyle, <https://vivliostyle.org/>
- By: @shinokada
- GitHub repo: <https://github.com/shinokada/markdown-docs-as-pdf>