SLIDEV Docs - English

Table of contents

sli.dev

Documentation for Slidev

Translations

	Repo	Site	Maintainers
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

Start Server Locally

```
npm i -g pnpm

pnpm i
pnpm run dev
```

And then visit `http://localhost:3000`

Or install the Vite extension for VS Code to edit side-by-side.

Help on Translating

See TRANSLATIONS.md

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

Use Addon

Addons are sets of additional components, layouts, styles, configuration...etc. that you can use in your presentation.

They are quite similar to themes, but in general:

- they don't affect the global styles of your slides
- you can use multiple addons in one presentation

To use addons, you have to install them manually via:

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

And then declare the addons either in your frontmatter:

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

Or in your `package.json` file:

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

Go to TOC

Write an Addon

Available since v0.32.1

Capability

An addon can contribute to the following points:

- Global styles (use with caution has it is more the role of themes)
- Provide custom layouts or override the existing one
- Provide custom components or override the existing one
- Extend Windi CSS configurations
- Configure tools like Monaco and Prism

Conventions

Addons are published to npm registry, and they should follow the conventions below:

- Package name should start with `slidev-addon-`, for example: `slidev-addon-awesome`
- Add `slidev-addon` and `slidev` in the `keywords` field of your `package.json`

Setup

Initialization

To create your addon, start by creating a directory with create a `package.json` file (you can use `npm init`).

Then, install slidev dependencies:

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

Testing

To set up the testing playground for your addon, you can create an `example.md` file with some content.

And optionally, you can also add some scripts to your `packages.json`

```
// package.json
{
  "scripts": {
    "dev": "slidev example.md",
    "build": "slidev build example.md",
    "export": "slidev export example.md",
```

```
    "screenshot": "slidev export example.md --format png"
  }
}
```

To publish your addon, simply run `npm publish` and you are good to go. There is no build process required (which means you can directly publish `.vue` and `.ts` files, Slidev is smart enough to understand them).

Addon contribution points follow the same conventions as local customization, please refer to the docs for the naming conventions.

Addon metadata

Slidev Version

If the addon is relying on a specific feature of Slidev that are newly introduced, you can set the minimal Slidev version required to have your addon working properly:

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

If users are using older versions of Slidev, an error will be thrown.

Go to TOC

Components

Built-in Components

> The documentations of this section is still working in progress. Before that, you can take a look at the source code directly.

`Toc`

Insert a Table Of Content.

If you want a slide to not appear in the `<Toc>` component, you can use in the front matter block of the slide:

```
---
hideInToc: true
---
```

Titles are displayed using the `<Titles>` component

Usage

```
<Toc />
```

Parameters:

- `columns` (`string | number` , default: `1`): The number of columns of the display
- `listClass` (`string | string[]` , default: `''`): Classes to apply to the table of contents list
- `maxDepth` (`string | number` , default: `Infinity`): The maximum depth level of title to display
- `minDepth` (`string | number` , default: `1`): The minimum depth level of title to display
- `mode` (`'all' | 'onlyCurrentTree'| 'onlySiblings'` , default: `'all'`):
  - `'all'` : Display all items
  - `'onlyCurrentTree'` : Display only items that are in current tree (active item, parents and children of active item)
  - `'onlySiblings'` : Display only items that are in current tree and their direct siblings

`Link`

Insert a link you can use to navigate to a given slide.

Usage

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

Parameters:

- `to` (`string | number`): The path of the slide to navigate to (slides starts from `1`)
- `title` (`string`): The title to display

`Titles`

Insert the main title from a slide parsed as HTML.

Titles and title levels get automatically retrieved from the first title element of each slides.

You can override this automatic behaviour for a slide by using the front matter syntax:

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

Usage

The `<Titles>` component is a virtual component you can import with:

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

Then you can use it with:

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

Parameters:

- `no` (`string | number`): The number of the slide to display the title from (slides starts from `1`)

Custom Components

Create a directory `components/` under your project root, and simply put your custom Vue components under it, then you can use it with the same name in your markdown file!

Read more in the Customization section.

Theme-provided Components

Themes can provide components as well. Please read their documentations for what they have provided.

Check more in the directory structure section.

Go to TOC

Layouts

Built-in Layouts

> As themes may override layouts behaviour, the best way to know exactly the usage, parameters and examples is referring their documentation.

`center`

Displays the content in the middle of the screen.

`cover`

Used to display the cover page for the presentation, may contain the presentation title, contextualization, etc.

`default`

The most basic layout, to display any kind of content.

`end`

The final page for the presentation.

`fact`

To show some fact or data with a lot of prominence on the screen.

`full`

Use all the space of the screen to display the content.

`image-left`

Shows an image on the left side of the screen, the content will be placed on the right side.

Usage

```
---
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

Shows an image on the right side of the screen, the content will be placed on the left side.

Usage

```
---
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

Shows an image as the main content of the page.

Usage

```
---
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/slidevjs/slidev

# 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/slidevjs/slidev
```

```
# 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/slidevjs/slidev
---
```

`intro`

To introduce the presentation, usually with the presentation title, a short description, the author, etc.

`none`

A layout without any existent styling.

`quote`

To display a quotation with prominience.

`section`

Used to mark the beginning of a new presentation section.

`statement`

Make an affirmation/statement as the main page content.

`two-cols`

Separates the page content in two columns.

Usage

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

# Left

This shows on the left

::right::
```

```
# Right

This shows on the right
```

Custom Layouts

Create a directory `layouts/` under your project root, and simply put your custom Vue layout components under it.

Read more in the Customization section.

Theme-provided Layouts

Themes can provide layouts or override existing ones. Please read their documentation for what they have provided.

Go to TOC

Configure KaTeX

Create `./setup/katex.ts` with the following content:

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

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

With the setup, you can provide the custom setting for KaTex Options. Refer to the type definitions and their documentation for more details.

Go to TOC

Configure Mermaid

Create `./setup/mermaid.ts` with the following content:

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

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

With the setup, you can provide a custom default setting for Mermaid. Refer to the type definitions and its documentation for more details.

Go to TOC

Configure Monaco

Create `./setup/monaco.ts` with the following content:

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

export default defineMonacoSetup(async (monaco) => {
  // use `monaco` to configure
})
```

Learn more about configuring Monaco.

Usage

To use Monaco in your slides, simply append `{monaco}` to your code snippets:

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

console.log(plusOne.value) // 2

plusOne.value++ // error
//```
````

To

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

console.log(plusOne.value) // 2

plusOne.value++ // error
//```
````

Exporting

By default, Monaco will ONLY work on `dev` mode. If you would like to have it available in the exported SPA, configure it in your frontmatter:

```yaml
---
monaco: true # default "dev"
---
```

Types Auto Installing

When use TypeScript with Monaco, types for dependencies will be installed to the client-side automatically.

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

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

In the example above, make sure `vue` and `@vueuse/core` are installed locally as dependencies / devDependencies, Slidev will handle the rest to get the types working for the editor automatically!

Configure Themes

The theme is controlled by Slidev based on the light/dark theme. If you want to customize it, you can pass the theme id to the setup function:

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

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

If you want to load custom themes:

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

// change to your themes
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',
    },
  }
})
```

> If you are creating a theme for Slidev, use dynamic `import()` inside the setup function to get better tree-shaking and code-splitting results.

Go to TOC

Configure and Extend the Parser

Slidev parses your presentation file (e.g. `slides.md`) in three steps:

1. A "preparsing" step is carried out: the file is split into slides using the `---` separator, and considering the possible frontmatter blocks.
2. Each slide is parsed with an external library.
3. Slidev resolves the special frontmatter property `src: ....`, which allows to include other md files.

Markdown Parser

Configuring the markdown parser used in step 2 can be done by configuring Vite internal plugins

Preparser Extensions

> Available since v0.37.0

:::warning Important: when modifying the preparser configuration, you need to stop and start slidev again (restart might not be sufficient). :::

The preparser (step 1 above) is highly extensible and allows to implement custom syntaxes for your md files. Extending the preparser is considered **an advanced feature** and is susceptible to break editor integrations due to implicit changes in the syntax.

To customize it, create a `./setup/preparser.ts` file with the following content:

```ts
import { definePreparserSetup } from '@slidev/types'

export default definePreparserSetup((filepath) => {
  return [
    {
      transformRawLines(lines) {
        for (const i in lines) {
          if (lines[i] === '@@@')
            lines[i] = 'HELLO'
        }
      },
    }
  ]
})
```

This example systematically replaces any `@@@` line by a line with `hello`. It illustrates the structure of a preparser configuration file and some of the main concepts the preparser involves:

- `definePreparserSetup` must be called with a function as parameter.
- The function receives the file path (of the root presentation file) and could use this information (e.g., enable extensions based on the presentation file).

- The function must return a list of preparser extensions.
- An extension can contain:
  - a `transformRawLines(lines)` function that runs just after parsing the headmatter of the md file and receives a list of all lines (from the md file). The function can mutate the list arbitrarily.
  - a `transformSlide(content, frontmatter)` function that is called for each slide, just after splitting the file, and receives the slide content as a string and the frontmatter of the slide as an object. The function can mutate the frontmatter and must return the content string (possibly modified, possibly `undefined` if no modifications have been done).
  - a `name`

Example Preparser Extensions

Use case 1: compact syntax top-level presentation

Imagine a situation where (part of) your presentation is mainly showing cover images and including other md files. You might want a compact notation where for instance (part of) `slides.md` is as follows:

```
@cover: /nice.jpg
# Welcome
@src: page1.md
@src: page2.md
@cover: /break.jpg
@src: pages3-4.md
@cover: https://source.unsplash.com/collection/94734566/1920x1080
# Questions?
see you next time
```

To allow these `@src:` and `@cover:` syntaxes, create a `./setup/preparser.ts` file with the following content:

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

export default definePreparserSetup((filepath) => {
  return [
    {
      transformRawLines(lines) {
        let i = 0
        while (i < lines.length) {
          const l = lines[i]
          if (l.match(/^@cover:/i)) {
            lines.splice(i, 1,
              '---',
              'layout: cover',
              `background: ${l.replace(/^@cover: */i, '')}`,
              '---',
              '')
            continue
          }
          if (l.match(/^@src:/i)) {
            lines.splice(i, 1,
              '---',
              `src: ${l.replace(/^@src: */i, '')}`,
              '---',
```

```
            '')
          continue
        }
        i++
      }
    }
  },
]
})
```

And that's it.

Use case 2: using custom frontmatter to wrap slides

Imagine a case where you often want to scale some of your slides but still want to use a variety of existing layouts so create a new layout would not be suited. For instance, you might want to write your `slides.md` as follows:

```
---
layout: quote
_scale: 0.75
---

# Welcome

> great!

---
_scale: 4
---
# Break

---

# Ok

---
layout: center
_scale: 2.5
---
# Questions?
see you next time
```

Here we used an underscore in `_scale` to avoid possible conflicts with existing frontmatter properties (indeed, the case of `scale`, without underscore would cause potential problems).

To handle this `_scale: ...` syntax in the frontmatter, create a `./setup/preparser.ts` file with the following content:

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

export default definePreparserSetup((filepath) => {
  return [
    {
      transformSlide(content, frontmatter) {
```

```
        if ('_scale' in frontmatter) {
          return [
            `<Transform :scale=${frontmatter['_scale']}>`,
            '',
            content,
            '',
            '</Transform>'
          ].join('\n')
        }
      },
    },
  ]
})
```

And that's it.

Go to TOC

Configure Shortcuts

Available since v0.20

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

Getting started

Create `./setup/shortcuts.ts` with the following content:

```ts
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,
    },
  ]
})
```

With the setup, you can provide the custom setting for shortcuts mentioned in Navigation The above configuration binds next animation or slide to `enter` and previous animation or slide to `backspace`.

The configuration function receives an object with some navigation methods, and returns an array containing some shortcut configuration. Refer to the type definitions for more details.

Advanced key binding

The `key` type only allows for strings, but you can still bind multiple keys by using following convention:

```ts
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,
    }
  ]
})
```

Advanced navigation features

The `nav` navigation operations allows you to access some functionalities than basic *next slide* or *previous slide*. See the following for use-cases:

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

export default defineShortcutsSetup((nav: NavOperations) => {
  return [
    {
      key: 'e',

      // Set the `e` keyboard shortcut to be used as a bookmark
      // or quick-access of sorts, to navigate specifically to
      // slide number 42
      fn: () => nav.go(42),
      autoRepeat: true,
    }
  ]
})
```

Refer to useMagicKeys | VueUse for more details about key pressed event.

Go to TOC

Configure Vite

Slidev is powered by Vite under the hood. This means you can leverage Vite's great plugin system to customize your slides even further.

The `vite.config.ts` will be respected if you have one.

Slidev has the following plugins preconfigured:

- @vitejs/plugin-vue
- unplugin-vue-components
- unplugin-icons
- vite-plugin-vue-markdown
- vite-plugin-remote-assets
- vite-plugin-windicss
- unocss/vite

Learn more about the pre-configurations here.

Configure Internal Plugins

> Available since v0.21

To configure the built-in plugins list above, create `vite.config.ts` with the following content. Please note Slidev has some preconfigure options for those plugins, this usage will override some of them, which could potentially cause the app to break. Please treat this as **an advanced feature**, make sure you know what you are doing before moving on.

```ts
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 */
  },
})
```

See the type declarations for more options.

Go to TOC

Configure Vue

Slidev uses Vue 3 to render the application on the client side. You can extend the app to add custom plugins or configurations.

Create `./setup/main.ts` with the following content:

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

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

This could also be used as the main entrance of your Slidev app to do some initializations before the app starts.

Learn more: Vue Application API.

Go to TOC

Configure Windi CSS

Markdown naturally supports embedded HTML markups. You can therefore style your content the way you want. To provide some convenience, we have Windi CSS built-in, so you can style markup directly using class utilities.

For example:

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

### Name

- Item 1
- Item 2

</div>
```

The Attributify Mode in Windi CSS v3.0 is enabled by default.

Configurations

To configure Windi CSS, create `setup/windicss.ts` with the following content to extend the builtin configurations

```
// setup/windicss.ts

import { defineWindiSetup } from '@slidev/types'

// extending the builtin windicss configurations
export default defineWindiSetup(() => ({
  shortcuts: {
    // custom the default background
    'bg-main': 'bg-white text-[#181818] dark:(bg-[#121212] text-[#ddd])',
  },
  theme: {
    extend: {
      // fonts can be replaced here, remember to update the web font links in `index.html`
      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',
      },
    },
  },
}))
```

Learn more about Windi CSS configurations

Go to TOC

Directory Structure

Slidev employs some directory structure conventions to minimize the configuration surface and to make the functionality extensions flexible and intuitive.

The basic structure is as follows:

```
your-slidev/
  ├── components/       # custom components
  ├── layouts/          # custom layouts
  ├── public/           # static assets
  ├── setup/            # custom setup / hooks
  ├── styles/           # custom style
  ├── index.html        # injections to index.html
  ├── slides.md         # the main slides entry
  └── vite.config.ts    # extending vite config
```

All of them are optional.

Components

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

Components inside this directory can be directly used in the slides Markdown with the same component name as the file name.

For example:

```
your-slidev/
  ├── ...
  └── components/
      ├── MyComponent.vue
      └── HelloWorld.ts
```

```
<!-- slides.md -->

# My Slide

<MyComponent :count="4"/>

<!-- both namings work -->

<hello-world foo="bar">
  Slot
</hello-world>
```

This feature is powered by `unplugin-vue-components`, learn more there.

Slidev also provides some built-in components for you to use.

Layouts

Conventions: `./layouts/*.{vue,js,ts,jsx,tsx}`

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

You can use any filename for your layout. You then reference your layout in you YAML header using the filename.

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

If the layout you provide has the same name as a built-in layout or a theme layout, your custom layout will take precedence over the built-in/theme layout. The priority order is `local > theme > built-in`.

In the layout component, use `<slot/>` for the slide content. For example:

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

Public

Conventions: `./public/*`

Assets in this directory will be served at root path `/` during dev, and copied to the root of the dist directory as-is. Read more about Vite's `public` directory.

Style

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

Files following this convention will be injected to the App root. If you need to import multiple css entries, you can create the following structure and managing the import order yourself.

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

```ts
// styles/index.ts

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

Styles will be processed by Windi CSS and PostCSS, so you can use css nesting and at-directives out-of-box. For example:

```ts
.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');
  }
}
```

Learn more about the syntax.

index.html

Conventions: `index.html`

The `index.html` provides the ability to inject meta tags and/or scripts to the main `index.html`

For example, for the following custom `index.html`:

```ts
<!-- ./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>
```

The final hosted `index.html` will be:

```ts
<!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/slidevjs/slidev/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>
```

Global Layers

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

Learn more: Global Layers

Go to TOC

Fonts

Available since v0.20

While you can use HTML and CSS to custom the fonts and style for your slides as you want, Slidev also provides a convenient way to use them effortlessly.

In your frontmatter, configure as following

```
---
fonts:
  # basically the text
  sans: 'Robot'
  # use with `font-serif` css class from windicss
  serif: 'Robot Slab'
  # for code blocks, inline code, etc.
  mono: 'Fira Code'
---
```

And that's all.

Fonts will be **imported automatically from Google Fonts**. That means you can use any fonts available on Google Fonts directly.

Local Fonts

By default, Slidev assumes all the fonts specified via `fonts` configurations come from Google Fonts. If you want to use local fonts, specify the `fonts.local` to opt-out the auto-importing.

```
---
fonts:
  # like font-family in css, you can use `,` to separate multiple fonts for
fallback
  sans: 'Helvetica Neue,Robot'
  # mark 'Helvetica Neue' as local font
  local: 'Helvetica Neue'
---
```

Weights & Italic

By default, Slidev imports three weights `200`, `400`, `600` for each font. You can configure them by:

```
---
fonts:
  sans: 'Robot'
  # default
  weights: '200,400,600'
```

```
  # import italic fonts, default `false`
  italic: false
---
```

This configuration applies to all web fonts. For more fine-grained controls of each font's weights, you will need to manually import them with HTML and CSS.

Fallback Fonts

For most of the scenarios, you only need to specify the "special font" and Slidev will append the fallback fonts for you, for example:

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

will result in

```
.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;
}
```

If you want to disable the fallback fonts, configure as following

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

Providers

- Options: `google` | `none`
- Default: `google`

Currently, only Google Fonts is supported, we are planned to add more providers in the future. Specify to `none` will disable the auto-importing feature entirely and treat all the fonts local.

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

Go to TOC

Global Layers

Available since v0.17

Global layers allow you to have custom components that **persistent** across slides. This could be useful for having footers, cross-slides animations, global effects, etc.

Slidev provides three layers for this usage, create `global-top.vue`, `global-bottom.vue` or `custom-nav-controls.vue` under your project root and it will pick up automatically.

Layers relationship:

- Global Top (`global-top.vue`)
- Slides
- Global Bottom (`global-bottom.vue`)
- NavControls
  - Customized Navigation Controls (`custom-nav-controls.vue`)

Example

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

The text `Your Name` will appear to all your slides.

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

The button `Next` will appear in NavControls.

To enable it conditionally, you can apply it with the Vue Global Context.

```vue
<!-- hide the footer from Page 4 -->
<template>
  <footer
    v-if="$slidev.nav.currentPage !== 4"
    class="absolute bottom-0 left-0 right-0 p-2"
  >
    Your Name
  </footer>
</template>
```

```
<!-- hide the footer from "cover" layout -->
<template>
  <footer
    v-if="$slidev.nav.currentLayout !== 'cover'"
    class="absolute bottom-0 left-0 right-0 p-2"
  >
    Your Name
  </footer>
</template>
```

```
<!-- an example footer for pages -->
<template>
  <footer
    v-if="$slidev.nav.currentLayout !== 'cover'"
    class="absolute bottom-0 left-0 right-0 p-2"
  >
    {{ $slidev.nav.currentPage }} / {{ $slidev.nav.total }}
  </footer>
</template>
```

```
<!-- custom-nav-controls -->
<!-- hide the button in Presenter model -->
<template>
  <button v-if="!$slidev.nav.isPresenter" class="icon-btn" title="Next"
@click="$slidev.nav.next">
    <carbon:arrow-right />
  </button>
</template>
```

Go to TOC

Highlighters

Slidev comes with two syntax highlighter for you to choose from:

- Prism
- Shiki

Prism is one of the most popular syntax highlighters. The highlighting is done by adding token classes to the code and it's colored using CSS. You can browse through their official themes, or create/customize one yourself very easily using `prism-theme-vars`.

Shiki, on the other hand, is a TextMate grammar-powered syntax highlighter. It generates colored tokens, so there is no additional CSS needed. Since it has great grammar support, the generated colors are very accurate, just like what you will see in VS Code. Shiki also comes with a bunch of built-in themes. The downside of Shiki is that it also requires TextMate themes (compatible with VS Code theme) to do the highlighting, which can be a bit harder to customize.

Slidev themes usually support both Prism and Shiki, but depending on the theme you are using, it might only support one of them.

When you have the choice, the tradeoff is basically:

- **Prism** for easier customization
- **Shiki** for more accurate highlighting

By default, Slidev uses Prism. You can change it by modifying your frontmatter:

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

Configure Prism

To configure your Prism, you can just import the theme css or use `prism-theme-vars` to configure themes for both light and dark mode. Refer to its docs for more details.

Configure Shiki

Create `./setup/shiki.ts` file with the following content

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

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

```
    },
  }
})
```

Refer to Shiki's docs for available theme names.

Or if you want to use your own theme:

```ts
/* ./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'),
    },
  }
})
```

Go to TOC

Customizations

Slidev is fully customizable, from styling to tooling configurations. It allows you to configure the tools underneath (Vite, Windi CSS, Monaco, etc.)

Frontmatter Configures

You can configure Slidev in the frontmatter of your first slide, the following shows the default value for each option.

```yaml
---
# theme id or package name
# Learn more: https://sli.dev/themes/use.html
theme: 'default'
# title of your slide, will auto infer from the first header if not specified
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

# enabled pdf downloading in SPA build, can also be a custom url
download: false
# filename of the export file
exportFilename: 'slidev-exported'
# syntax highlighter, can be 'prism' or 'shiki'
highlighter: 'prism'
# show line numbers in code blocks
lineNumbers: false
# enable monaco editor, can be boolean, 'dev' or 'build'
monaco: 'dev'
# download remote assets in local using vite-plugin-remote-assets, can be boolean,
'dev' or 'build'
remoteAssets: false
# controls whether texts in slides are selectable
selectable: true
# enable slide recording, can be boolean, 'dev' or 'build'
record: 'dev'

# force color schema for the slides, can be 'auto', 'light', or 'dark'
colorSchema: 'auto'
# router mode for vue-router, can be "history" or "hash"
routerMode: 'history'
# aspect ratio for the slides
aspectRatio: '16/9'
# real width of the canvas, unit in px
canvasWidth: 980
# used for theme customization, will inject root styles as `--slidev-theme-x` for
attribute `x`
themeConfig:
  primary: '#5d8392'

# favicon, can be a local file path or URL
favicon: 'https://cdn.jsdelivr.net/gh/slidevjs/slidev/assets/favicon.png'
# URL of PlantUML server used to render diagrams
plantUmlServer: 'https://www.plantuml.com/plantuml'
```

```yaml
# fonts will be auto imported from Google fonts
# Learn more: https://sli.dev/custom/fonts
fonts:
  sans: 'Roboto'
  serif: 'Roboto Slab'
  mono: 'Fira Code'

# default frontmatter applies to all slides
defaults:
  layout: 'default'
  # ...

# drawing options
# Learn more: https://sli.dev/guide/drawing.html
drawings:
  enabled: true
  persist: false
  presenterOnly: false
  syncAll: true
---
```

Check out the type definitions for more options.

Directory Structure

Slidev uses directory structure conventions to minimalize the configuration surface and make extensions in functionality flexible and intuitive.

Refer to the Directory Structure section.

Config Tools

- Highlighters
- Configure Vue
- Configure Vite
- Configure Windi CSS
- Configure Monaco
- Configure KaTeX
- Configure Mermaid

Go to TOC

Vue Global Context

Slidev injected a global Vue context `$slidev` for advanced conditions or navigation controls.

Usage

You can access it anywhere in your markdown and Vue template, with the "Mustache" syntax.

```
<!-- 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>
```

Properties

`$slidev.nav`

A reactive object holding the properties and controls of the slides navigation. For examples:

```
$slidev.nav.next() // go next step

$slidev.nav.nextSlide() // go next slide (skip v-clicks)

$slidev.nav.go(10) // go slide #10
```

```
$slidev.nav.currentPage // current slide number

$slidev.nav.currentLayout // current layout id

$slidev.nav.clicks // current clicks count
```

For more properties available, refer to the nav.ts exports.

`$slidev.configs`

A reactive object holding the parsed configurations in the first frontmatter of your `slides.md`. For example

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

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

`$slidev.themeConfigs`

A reactive object holding the parsed theme configurations.

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

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

Go to TOC

Animations

Click Animations

`v-click`

To apply "click animations" for elements, you can use the `v-click` directive or `<v-click>` components

```
# Hello

<!-- Component usage: this will be invisible until you press "next" -->
<v-click>

Hello World

</v-click>

<!-- Directive usage: this will be invisible until you press "next" the second
time -->
<div v-click class="text-xl p-2">

Hey!

</div>
```

`v-after`

The usage of `v-after` is similar to `v-click` but it will turn the element visible when the previous `v-click` is triggered.

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

When you click the "next" button, both `Hello` and `World` will show up together.

`v-click-hide`

Same as `v-click` but instead of making the element appear, it makes the element invisible after clicking.

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

`v-clicks`

`v-clicks` is only provided as a component. It's a shorthand to apply the `v-click` directive to all its child elements. It is especially useful when working with lists.

```
<v-clicks>

- Item 1
- Item 2
- Item 3
```

```
- Item 4

</v-clicks>
```

An item will become visible each time you click "next".

Custom Clicks Count

By default, Slidev counts how many steps are needed before going to the next slide. You can override this setting by passing the `clicks` frontmatter option:

```
---
# 10 clicks in this slide, before going to the next
clicks: 10
---
```

Ordering

Passing the click index to your directives, you can customize the order of the revealing

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

```
<!-- the order is reversed -->
<div v-click="3">1</div>
<div v-click="2">2</div>
<div v-click="1">3</div>
```

```
---
clicks: 3
---

<!-- visible after 3 clicks -->
<v-clicks at="3">
  <div>Hi</div>
</v-clicks>
```

Element Transitions

When you apply the `v-click` directive to your elements, it will attach the class name `slidev-vclick-target` to it. When the elements are hidden, the class name `slidev-vclick-hidden` will also be attached. For example:

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

After a click, it will become

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

By default, a subtle opacity transition is applied to those classes:

```
// the default

.slidev-vclick-target {
  transition: opacity 100ms ease;
}

.slidev-vclick-hidden {
  opacity: 0;
  pointer-events: none;
}
```

You can override them to customize the transition effects in your custom stylesheets.

For example, you can achieve the scaling up transitions by:

```
// styles.css

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

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

To specify animations for only certain slide or layout

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

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

Learn more about customizing styles

Motion

Slidev has @vueuse/motion built-in. You can use the `v-motion` directive to any elements to make apply motion on them. For example

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

The text `Slidev` will move from `-80px` to its original position on initialization.

Note: Slidev preloads the next slide for performance, which means the animations might start before you navigate to the page. To get it works properly, you can disable the preloading for the particular slide

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

Or control the element life-cycle with `v-if` to have fine-grained controls

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

Learn mode: Demo | @vueuse/motion | v-motion | Presets

Pages Transitions

Built-in support for slides is NOT YET provided in the current version. We are planning to add support for them in the next major version. Before that, you can still use your custom styles and libraries to do that.

Go to TOC

Drawing & Annotations

> Available since v0.23

We have drauu built-in for drawing and annotation that could enhance your presentation further.

To start, click the icon in the toolbar and start drawing. It's also available in the Presenter Mode. Drawings and annotations you created will be **synced up** automatically across all instances in real-time.

Use with Stylus Pen

When using a stylus pen on a tablet (for example, iPad with Apple Pencil), Slidev could smartly detect the input type. You can directly draw on your slides with the pen without turning on the drawing mode, while having your fingers or mouse control the navigation.

Persist Drawings

The following frontmatter configuration allows you to persist your drawings as SVGs under `.slidev/drawings` directory and have them inside your exported pdf or hosted site.

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

Disable Drawings

Entirely:

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

Only in Development:

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

Only in Presenter Mode:

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

Drawing Syncing

By default, Slidev syncs up your drawings across all instances. If you are sharing your slides with others, you might want to disable the syncing by:

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

With this config, only the drawing from the presenter instance will be able to sync with others.

Go to TOC

Editor Support

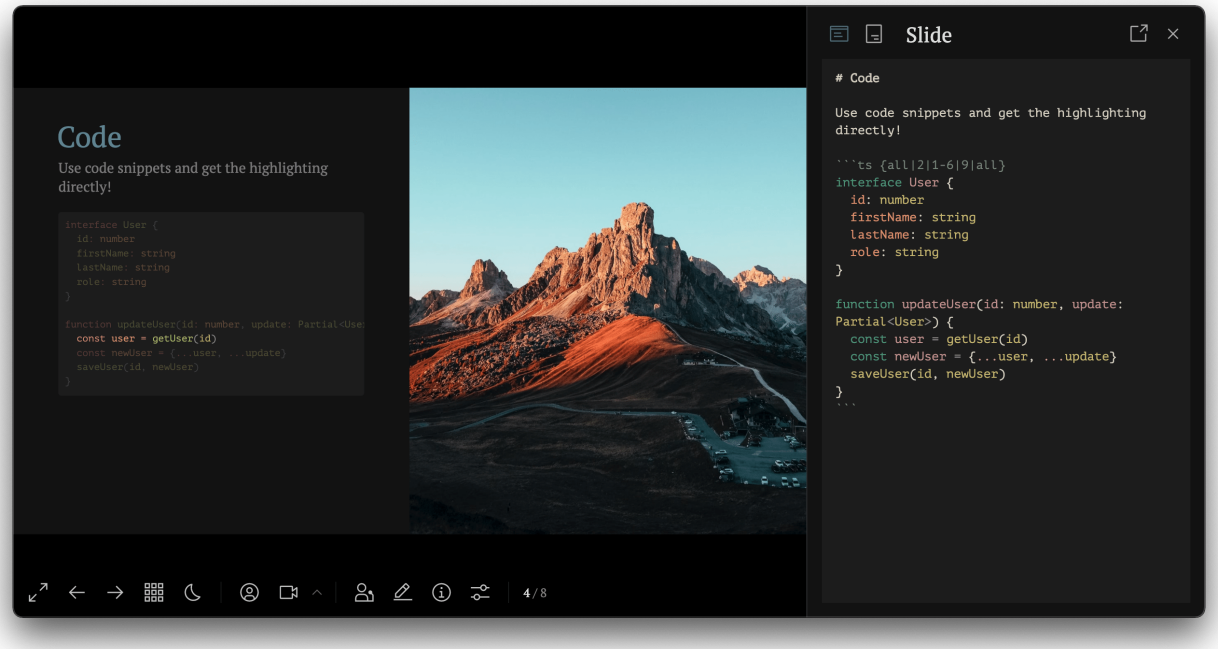
Since Slidev is using Markdown as the source entry, you can use ANY editors you love to write it.

If you want some high-level management to your slides, we have provided the following editor integrations for you!

Integrated Editor

Slidev comes with an integrated CodeMirror editor that will instantly reload and save the changes to your file.

Click the button to open it.

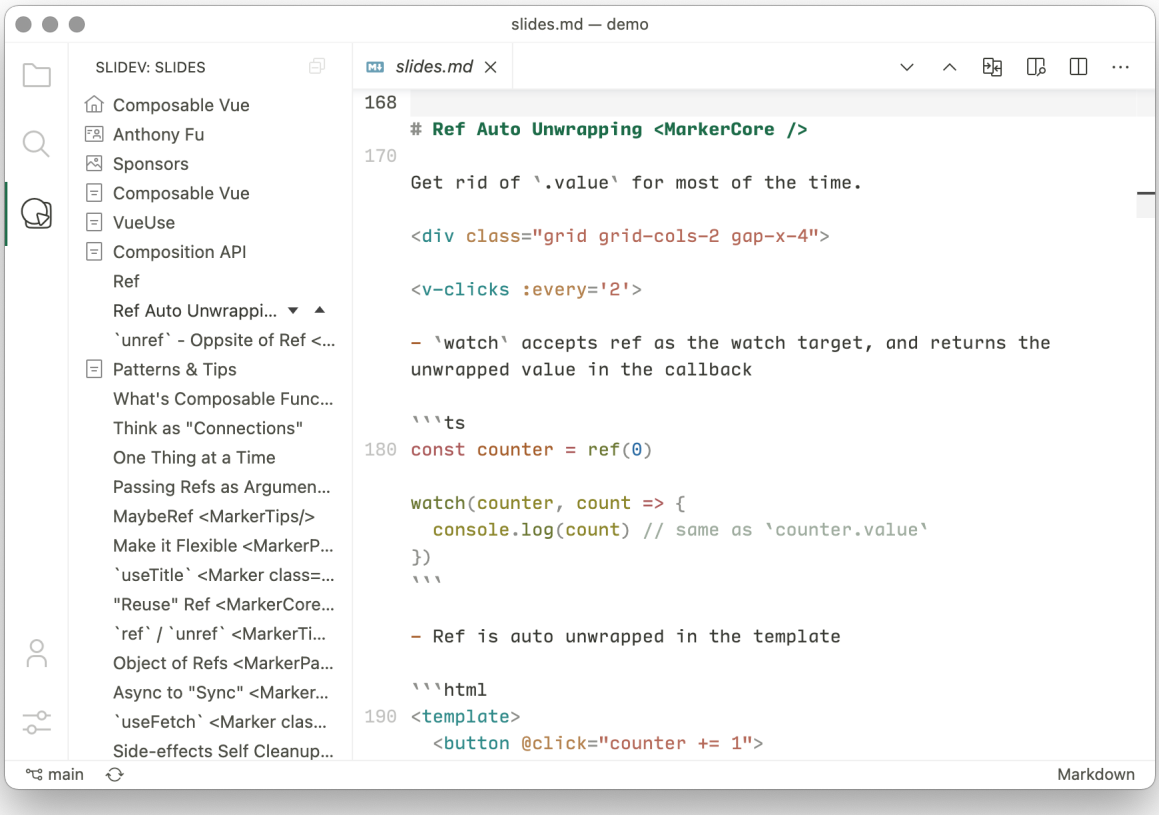


VS Code Extension

The VS Code extension provides some features to help you better organize your slides and have a quick overview of them.

Features

- View slides in the side panel
- Go to next / prev buttons
- Re-ordering slides
- Folding for slide blocks
- Convert Markdown to HTML


slides.md — demo
SLIDEV: SLIDES
Composable Vue
Anthony Fu
Sponsors
Composable Vue
VueUse
Composition API
Ref
Ref Auto Unwrappi...
`unref` - Oppsite of Ref <...
Patterns & Tips
What's Composable Func...
Think as "Connections"
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...
Side-effects Self Cleanup...
slides.md
168 # Ref Auto Unwrapping <MarkerCore />
170 Get rid of `.value` for most of the time.

<v-clicks :every='2'>
- `watch` accepts ref as the watch target, and returns the unwrapped value in the callback
```ts
180 const counter = ref(0)
watch(counter, count => {
console.log(count) // same as `counter.value`
})
```
- Ref is auto unwrapped in the template
```html
190 <template>
<button @click="counter += 1">
main
Markdown

Go to TOC
```

Exporting

Slides

PDF

> Exporting to PDF or PNG relies on Playwright for rendering. You will therefore need to install `playwright-chromium` to use this feature. If you are doing exporting in a CI environment, the playwright CI guide can be helpful.

Install `playwright-chromium`

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

Now export your slides to PDF using the following command

```
$ slidev export
```

After a few seconds, your slides will be ready at `./slides-export.pdf`.

In case you want to export your slides using the dark version of the theme, use the `--dark` option:

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

Export Clicks Steps

> Available since v0.21

By default, Slidev exports one page per slide with clicks animations disabled. If you want export slides with multiple steps into multiple pages, pass the `--with-clicks` options.

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

PNGs

When passing in the `--format png` option, Slidev will export PNG images for each slide instead of a PDF.

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

Single-Page Application (SPA)

See Static Hosting.

Presenter notes

> Available since v0.36.8

Export only the presenter notes (the last comment block for each slide) into a text document in PDF.

```
$ slidev export-notes
```

Go to TOC

FAQ

Grids

Since Slidev is based on the Web, you can apply any grid layouts as you want. CSS Grids, flexboxes, or even Masonry, you get the full controls.

Since we have Windi CSS built-in, here is one simple way for you to reference:

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

The first column

</div>
<div>

The second column

</div>
</div>
```

Go further, you can customize the size of each column like:

```
<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>
```

Learn more about Windi CSS Grids.

Positioning

Slides are defined in fixed sizes (default `980x552px`) and scale to fit with the user screen. You can safely use absolute position in your slides as they will scale along with the screen.

For example:

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

To change the canvas' actual size, you can pass the `canvasWidth` options in your first frontmatter:

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

Font Size

If you feel the font size in your slides are too small, you can adjust it in a few ways:

Override Local Style

You can override styles for each slide with the inlined `<style>` tag.

```
# Page 1

<style>
h1 {
  font-size: 10em;
}
</style>

---

# Page 2

This will not be affected.
```

Learn more: Embedded Styles

Override Global Style

You can provide custom global styles by creating `./style.css`, for example

```
/* style.css */

h1 {
  font-size: 10em !important;
}
```

Learn more: Global Style

Scale the Canvas

Changing the canvas' actual size will scale all your contents(text, images, components, etc.) and slides

```
---
# default: 980
# since the canvas gets smaller, the visual size will become larger
canvasWidth: 800
---
```

Use Transform

We provide a built-in component `<Transform />`, which is a thin wrapper of CSS transform property.

```
<Transform :scale="1.4">

- Item 1
- Item 2

</Transform>
```

Go to TOC

Static Hosting

Build Single Page Applications (SPA)

You can also build the slides into a self-hostable SPA:

```
$ slidev build
```

The generated application will be available under `dist/` and then you can host it on GitHub Pages, Netlify, Vercel, or whatever you want. Now you can share your slides with the rest of the world with a single link.

Base Path

To deploy your slides under sub-routes, you will need to pass the `--base` option. For example:

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

Refer to Vite's documentation for more details.

Provide Downloadable PDF

You can provide a downloadable PDF to the viewers of your SPA with the following config:

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

Slidev will generate a PDF file along with the build, and a download button will be displayed in the SPA.

You can also provide a custom URL for the PDF. In that case, the rendering process will be skipped.

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

Examples

Here are a few examples of the exported SPA:

- Starter Template
- Composable Vue by Anthony Fu

For more, check out Showcases.

Hosting

We recommend to use `npm init slidev@latest` to scaffold your project, which contains the necessary configuration files for hosting services out-of-the-box.

Netlify

- Netlify

Create `netlify.toml` in your project root with the following content.

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

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

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

Then go to your Netlify dashboard and create a new site with the repository.

Vercel

- Vercel

Create `vercel.json` in your project root with the following content.

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

Then go to your Vercel dashboard and create a new site with the repository.

GitHub Pages

- GitHub Pages

To deploy your slides on GitHub Pages:

- upload all the files of the project in your repo (i.e. named `name_of_repo`)
- create `.github/workflows/deploy.yml` with following content to deploy your slides to GitHub Pages via GitHub Actions. In this file, replace `<name_of_repo>` with `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 }}
```

- In your repository, go to Settings>Pages. Under "Build and deployment", select "Deploy from a branch", select "gh-pages" and "root". Click on save.
- Finally, after all workflows are executed, a link to the slides should appear under Settings>Pages.

Go to TOC

Getting Started

Overview

Slidev (slide + dev, `/slʌɪdɪv/`) is a web-based slides maker and presenter. It's designed for developers to focus on writing content in Markdown while also having the power of HTML and Vue components to deliver pixel-perfect layouts and designs with embedded interactive demos in your presentations.

It uses a feature-rich markdown file to generate beautiful slides with an instant reloading experience, along with many built-in integrations such as live coding, PDF exporting, presentation recording, and so on. Since it's powered by the web, you can do anything with Slidev - the possibilities are endless.

You can learn more about the rationale behind the project in the Why Slidev section.

Features

- 📝 **Markdown-based** - use your favorite editors and workflow
- 👨‍💻 **Developer Friendly** - built-in syntax highlighting, live coding, etc.
- 🎨 **Themable** - theme can be shared and used with npm packages
- 🌈 **Stylish** - on-demand utilities via Windi CSS or UnoCSS.
- 🤹 **Interactive** - embedding Vue components seamlessly
- 🎙 **Presenter Mode** - use another window, or even your phone to control your slides
- 🎨 **Drawing** - draw and annotate on your slides
- 🧱 **LaTeX** - built-in LaTeX math equations support
- 📰 **Diagrams** - creates diagrams with textual descriptions
- 🌟 **Icons** - Access to icons from any iconset directly
- 💻 **Editors** - integrated editor, or extension for VS Code
- 📹 **Recording** - built-in recording and camera view
- 📩 **Portable** - export into PDF, PNGs, or even a hostable SPA
- ⚡ **Fast** - instant reloading powered by Vite
- 🛠 **Hackable** - using Vite plugins, Vue components, or any npm packages

Tech Stack

Slidev is made possible by combining these tools and technologies.

- Vite - An extremely fast frontend tooling
- Vue 3 powered Markdown - Focus on the content while having the power of HTML and Vue components whenever needed
- Windi CSS or UnoCSS - On-demand utility-first CSS framework, style your slides at ease
- Prism, Shiki, Monaco Editor - First-class code snippets support with live coding capability
- RecordRTC - Built-in recording and camera view
- VueUse family - `@vueuse/core`, `@vueuse/head`, `@vueuse/motion`, etc.
- Iconify - Iconsets collection.
- Drauu - Drawing and annotations support

- KaTeX - LaTeX math rendering.
- Mermaid - Textual Diagrams.

Scaffolding Your First Presentation

Try it Online

sli.dev/new

Open in StackBlitz

Create Locally

With NPM:

```
$ npm init slidev
```

With Yarn:

```
$ yarn create slidev
```

Follow the prompts and start making your slides now! For more details about the markdown syntax, read through the syntax guide.

Command Line Interface

In a project where Slidev is installed, you can use the `slidev` binary in your npm scripts.

```
{
  "scripts": {
    "dev": "slidev", // start dev server
    "build": "slidev build", // build for production SPA
    "export": "slidev export" // export slides to pdf
  }
}
```

Otherwise, you can use it with `npx`

```
$ npx slidev
```

Run `slidev --help` for more options available.

Markdown Syntax

Slidev reads your `slides.md` file under your project root and converts them into slides. Whenever you made changes to it, the content of the slides will be updated immediately. For example:

```
# Slidev

Hello World

---

# Page 2

Directly use code blocks for highlighting

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

---

# Page 3
```

Read more about the Slidev Markdown syntax in the syntax guide.

Go to TOC

Installation

Starter Template

Slidev requires **Node.js >=14.0**

The best way to get started is using our official starter template.

With NPM:

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

With Yarn:

```
$ yarn create slidev
```

Follow the prompts and it will open up the slideshow at `http://localhost:3030/` automatically for you.

It also contains the basic setup and a short demo with instructions on how to get started with Slidev.

Install Manually

If you still prefer to install Slidev manually or would like to integrate it into your existing projects, you can do:

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

```
$ touch slides.md
```

```
$ npx slidev
```

Please note if you are using pnpm, you will need to enable shamefully-hoist option for Slidev to work properly:

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

Install Globally

Available since v0.14

You can install Slidev globally with the following command

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

And then use `slidev` everywhere without creating a project every time.

```
$ slidev
```

This command will also try to use local `@slidev/cli` if it has been found in the `node_modules`.

Install on Docker

If you need a rapid way to run a presentation with containers, you can use the prebuilt docker image maintained by tangramor, or build your own.

Just run following command in your work folder:

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

If your work folder is empty, it will generate a template `slides.md` and other related files under your work folder, and launch the server on port `3030`.

You can access your slides from `http://localhost:3030/`

Build deployable image

Or you can create your own slidev project to a docker image with Dockerfile:

```
FROM tangramor/slidev:latest

ADD . /slidev
```

Create the docker image: `docker build -t myppt .`

And run the container: `docker run --name myslides --rm --user node -p 3030:3030 myppt`

You can visit your slides from `http://localhost:3030/`

Build hostable SPA (Single Page Application)

Run command `docker exec -i slidev npx slidev build` on the running container `slidev`. It will generate static HTML files under `dist` folder.

Host on Github Pages

You can host `dist` in a static web site such as Github Pages or Gitlab Pages.

Because in Github pages the url may contain subfolder, so you need to modify the generated `index.html` to change `href="/assets/xxx` to `href="./assets/xxx`. Or you may use `--base=/<subfolder>/` option during the build process, such as: `docker exec -i slidev npx slidev build --base=/slidev_docker/`.

And to avoid Jekyll build process, you need to add an empty file `.nojekyll`.

Host by docker

You can also host it by yourself with docker:

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

Or create a static image with following Dockerfile:

```
FROM nginx:alpine

COPY dist /usr/share/nginx/html
```

Create the docker image: `docker build -t mystaticppt .`

And run the container: `docker run --name myslides --rm -p 80:80 mystaticppt`

You can visit your slides from http://localhost/

Refer to the tangramor/slidev_docker for more details.

Command Line Interface (CLI)

`@slidev/cli` Expose a few commands you can use with `npx slidev ...` or by adding scripts in your `package.json`:

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

In that case you will be able to run `npm run dev`.

You can pass options to any commands:

- boolean option are `true` if they are present, false otherwise (example: `slidev --open`)
- some options can have values you can add just after the option or by using the `=` character (example: `slidev --port 8080` or `slidev --port=8080`)

If you use npm scripts, don't forget to add `--` after the npm command:

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

`slidev [entry]`

Start a local server for Slidev.

- `[entry]` (`string`, default: `slides.md`): path to the slides markdown entry.

Options:

- `--port`, `-p` (`number`, default: `3030`): port number.
- `--open`, `-o` (`boolean`, default: `false`): open in browser.
- `--remote [password]` (`string`): listen to public host and enable remote control, if a value is passed then the presenter mode is private and only accessible by passing the given password in the URL query `password` parameter.
- `--log` (`'error', 'warn', 'info', 'silent'`, default: `'warn'`): Log level.
- `--force`, `-f` (`boolean`, default `false`): force the optimizer to ignore the cache and re-bundle.
- `--theme`, `-t` (`string`): override theme.

`slidev build [entry]`

Build hostable SPA.

- `[entry]` (`string`, default: `slides.md`): path to the slides markdown entry.

Options:

- `--watch`, `-w` (`boolean`, default: `false`): build watch.
- `--out`, `-o` (`string`, default: `dist`): output dir.
- `--base` (`string`, default: `/`): base URL (see https://cli.vuejs.org/config/#publicpath)
- `--download` (`boolean`, default: `false`): allow to download the slides as PDF inside the SPA.
- `--theme`, `-t` (`string`): override theme.

`slidev export [entry]`

Export slides to PDF (or other format).

- `[entry]` (`string`, default: `slides.md`): path to the slides markdown entry.

Options:

- `--output` (`string`, default: use `exportFilename` (see https://sli.dev/custom/#frontmatter-configures) or use `[entry]-export`): path to the output.
- `--base` (`'pdf', 'png', 'md'`, default: `'pdf'`): output format.
- `--timeout` (`number`, default: `30000`): timeout for rendering the print page (see https://playwright.dev/docs/api/class-page#page-goto).
- `--range` (`string`): page ranges to export (example: `'1,4-5,6'`).
- `--dark` (`boolean`, default: `false`): export as dark theme.
- `--with-clicks`, `-c` (`boolean`, default: `false`): export pages for every clicks (see https://sli.dev/guide/animations.html#click-animations).

- `--theme`, `-t` (`string`): override theme.

`slidev format [entry]`

Format the markdown file.

- `[entry]` (`string`, default: `slides.md`): path to the slides markdown entry.

`slidev theme [subcommand]`

Theme related operations.

Subcommands:

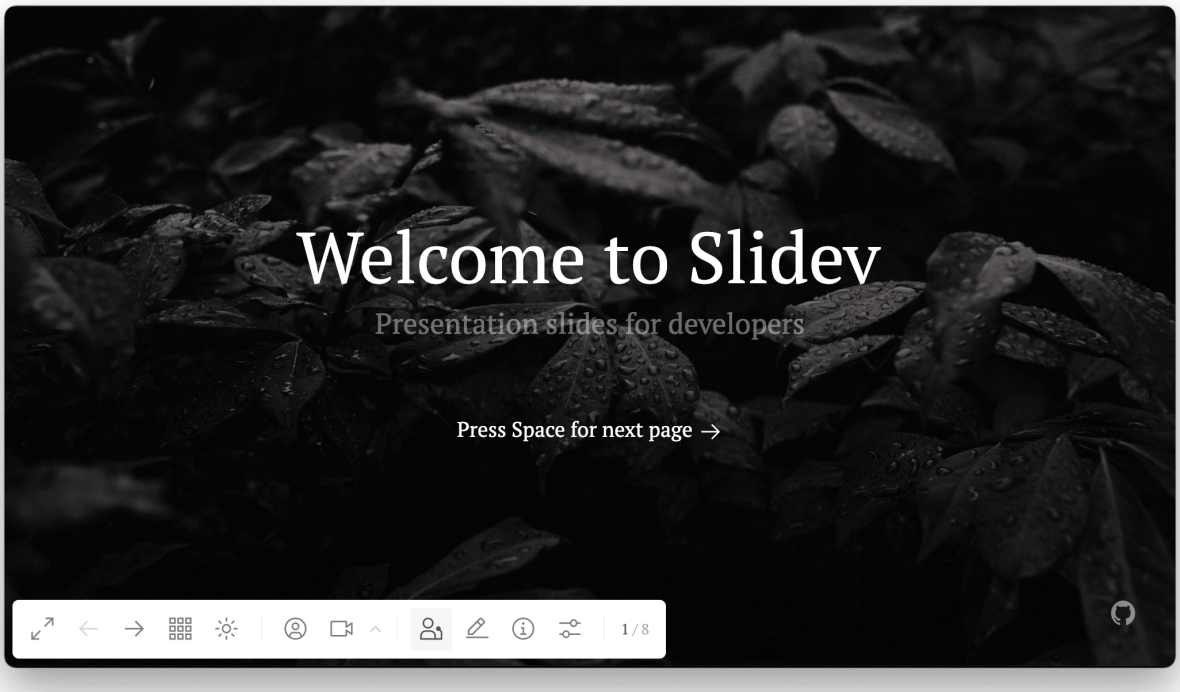
- `eject [entry]`: Eject current theme into local file system
  - `[entry]` (`string`, default: `slides.md`): path to the slides markdown entry.
  - Options: `--dir` *(`string`, default: `theme`): output dir.* `--theme`, `-t` (`string`): override theme.

Go to TOC

Navigation

Navigation Bar

Move your mouse to the bottom left corner of Slidev page to make the navigation bar appear.

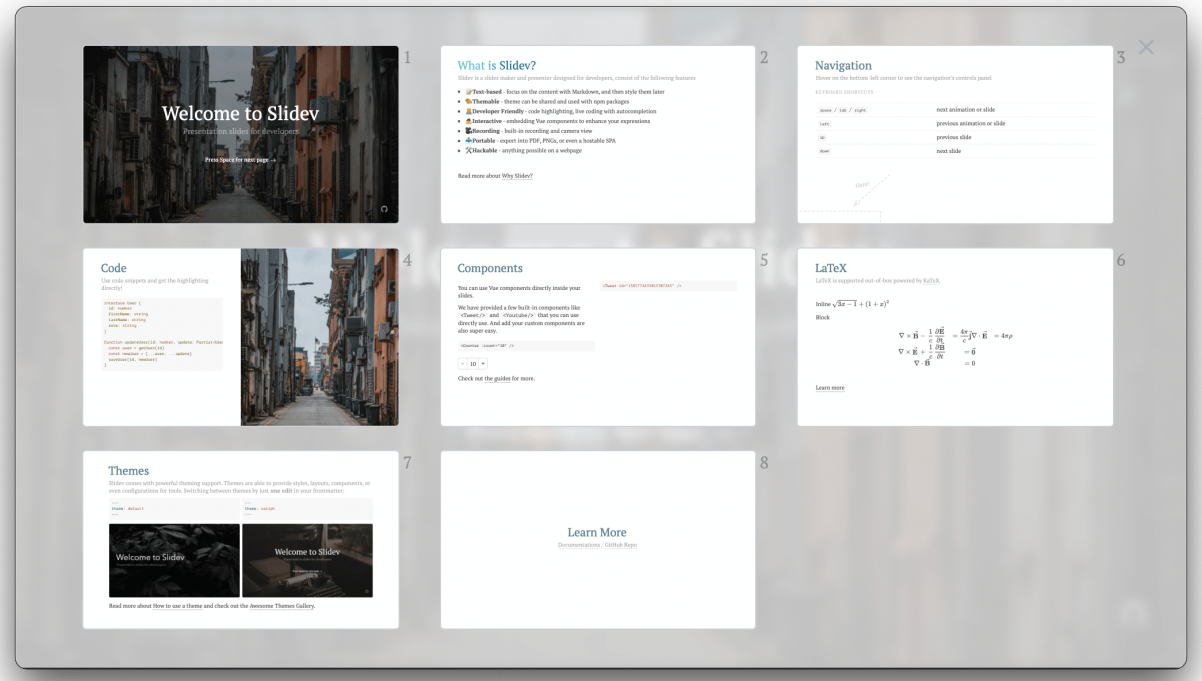


Shortcuts	Button	Description
`f`		toggle fullscreen
`right` / `space`		next animation or slide
`left`		previous animation or slide
`up`	-	previous slide
`down`	-	next slide
`o`		toggle slides overview
`d`		toggle dark mode
-		toggle camera view
-		recording
-		enter presenter mode

Shortcuts	Button	Description
-		toggle integrated editor
-		download slides (only appear in SPA build
-		show information about the slides
-		show settings menu
g	-	show goto...

Slides Overview

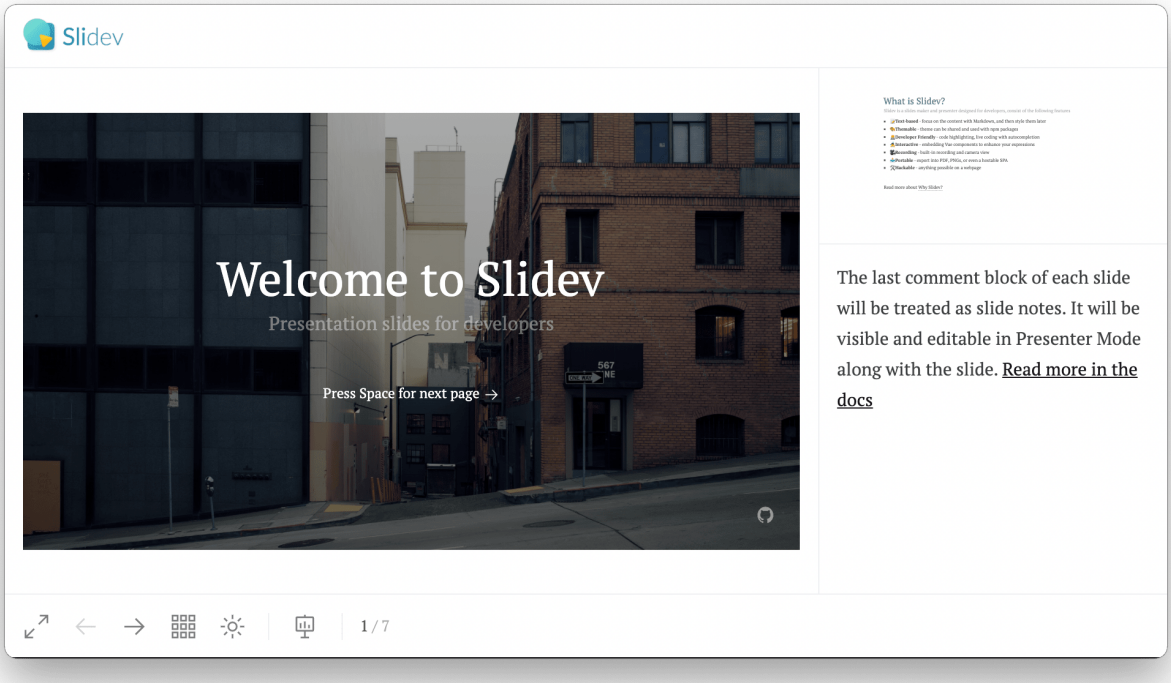
By pressing o or clicking the button in the navigation bar, you can have the overview of your slides so you can jump between them easily.



Go to TOC

Presenter Mode

Click the button in the navigation panel, or visit `http://localhost:3030/presenter` manually, to enter the presenter mode. Whenever you enter the presenter mode, other page instances will automatically stay in sync with the presenter.



Go to TOC

Recording

Slidev has a built-in recording and camera view. You can use them to record your presentation easily in one place.

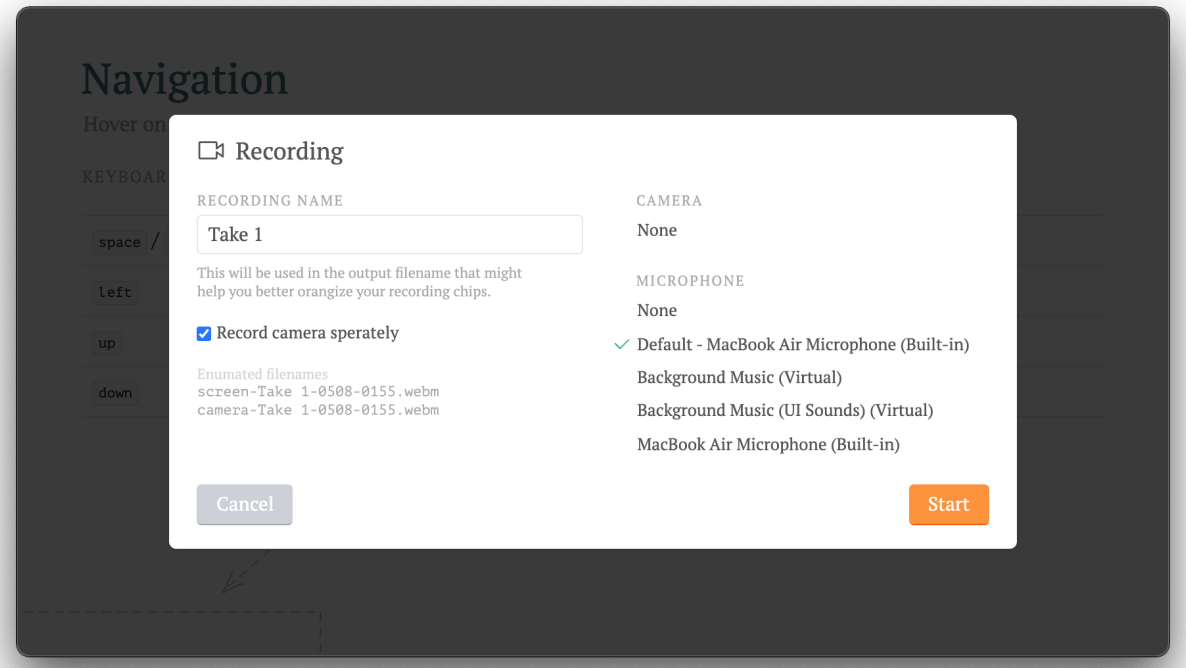
Camera View

Click the button in the navigation panel to show your camera view in the presentation. You can drag to move it, and use the handler on the right bottom corner to resize it. The size and position will persist in `localStorage` and will therefore be consistent across multiple refreshes, so no need to worry about that.

Recording

Clicking the button in the navigation panel will bring up a dialog for you. Here you can choose to either record your camera embedded in your slides or to separate them into two video files.

This feature is powered by RecordRTC and uses the WebRTC API.



Go to TOC

Markdown Syntax

Slides are written within **a single markdown file** (by default `./slides.md`).

You can use the Markdown features as you normally would, with the additional support of inlined HTML and Vue Components. Styling using Windi CSS is also supported. Use `---` padded with a new line to separate your slides.

```
# 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>
```

Front Matter & Layouts

Specify layouts and other metadata for each slide by converting the separators into front matter blocks. Each frontmatter starts with a triple-dash and ends with another. Texts between them are data objects in YAML format. For example:

```
---
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.
```

Refer to customization for more details.

Code Blocks

One big reason I am building Slidev is needing to make my code look just right in the slides. So just as you expected, you can use Markdown flavored code block to highlight your code.

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

We support Prism and Shiki as syntax highlighters. Refer to the highlighters section for more details.

Line Highlighting

To highlight specific lines, simply add line numbers within bracket `{}`. Line numbers start counting from 1.

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

To change the highlight in multiple steps, you can use `|` to separate them. For example

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

This will first highlight `a: Ref<number> | number` and `b: Ref<number> | number`, and then `return computed(() => unref(a) + unref(b))` after one click, and lastly, the whole block. Learn more in the clicks animations guide.

To skip highlighting any lines, you can set the line number to `0`. For example

```
//```ts {0}
function add(
  a: Ref<number> | number,
  b: Ref<number> | number
) {
```

````
  return computed(() => unref(a) + unref(b))
}
//```
````

If the code doesn't fit into one slide, you can pass an extra maxHeight option which will set fixed height and enable scrolling

````
//```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 Editor

Whenever you want to do some modification in the presentation, simply add `{monaco}` after the language id — it turns the block into a fully-featured Monaco editor!

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

Learn more about configuring Monaco.

Embedded Styles

You can use `<style>` tag in your Markdown directly to override styles for the **current slide**.

```
# This is Red

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

---

# Next slide is not affected
```

`<style>` tag in Markdown is always scoped. To have global style overrides, check out the customization section

Powered by Windi CSS, you can directly use nested css and directives (e.g. `@apply`)

```
# Slidev

> Hello `world`

<style>
```

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

Static Assets

Just like you would do in markdown, you can use images pointing to a remote or local url.

For remote assets, the built-in `vite-plugin-remote-assets` will cache them into the disk at the first run so you can have instant loading even for large images later on.

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

For local assets, put them into the `public` folder and reference them with **leading slash**.

```
![Local Image](pic.png)
```

For you want to apply custom sizes or styles, you can convert them to the `<img>` tag

```
<img src="/pic.png" class="m-40 h-40 rounded shadow" />
```

Notes

You can also take notes for each slide. They will show up in Presenter Mode for you to reference during presentations.

In Markdown, the last comment block in each slide will be treated as a note.

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

# Page 1

This is the cover page.

<!-- This is a note -->

---

# Page 2

<!-- This is NOT a note because it precedes the content of the slide -->

The second page

<!--
This is another note
-->
```

Icons

Slidev allows you to have the accessing to almost all the popular open-source iconsets **directly** in your markdown. Powered by `unplugin-icons` and Iconify.

The naming follows Iconify's conversion `{collection-name}-{icon-name}`. For example:

- `<mdi-account-circle />` - from Material Design Icons
- `<carbon-badge />` - from Carbon
- `<uim-rocket />` - from Unicons Monochrome
- `<twemoji-cat-with-tears-of-joy />` - from Twemoji
- `<logos-vue />` - from SVG Logos
- And much more...

Browse and search for all the icons available with Icônes.

Styling Icons

You can style the icons just like other HTML elements. For example:

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

Slots

> Available since v0.18

Some layouts can provide multiple contributing points using Vue's named slots.

For example, in `two-cols` layout, you can have two columns left (`default` slot) and right (`right` slot) side by side.

```
---
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

We also provide a shorthand syntax sugar `::name::` for slot name. The following example works exactly the same as the previous one.

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

# Left

This shows on the left

::right::

# Right

This shows on the right
```

You can also explicitly specify the default slot and provide in the custom order

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

::right::

# Right

This shows on the right

::default::

# Left

This shows on the left
```

Configurations

All configurations needed can be defined in the Markdown file. For example:

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

# Slidev

This is the cover page.
```

Learn more about frontmatter configurations

LaTeX

Slidev comes with LaTeX support out-of-box, powered by KaTeX.

Inline

Surround your LaTeX with a single `$` on each side for inline rendering.

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

Block

Use two (`$$`) for block rendering. This mode uses bigger symbols and centers the result.

```
$$
\begin{array}{c}

\nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}
{\partial t} &
= \frac{4\pi}{c}\vec{\mathbf{j}}    \nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho
\\

\nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}
{\partial t} & = \vec{\mathbf{0}} \\

\nabla \cdot \vec{\mathbf{B}} & = 0

\end{array}
$$
```

Learn more: Demo | KaTeX | `markdown-it-katex`

Diagrams

You can also create diagrams / graphs from textual descriptions in your Markdown, powered by Mermaid.

Code blocks marked as `mermaid` will be converted to diagrams, for example:

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

You can further pass an options object to it to specify the scaling and theming. The syntax of the object is a JavaScript object literal, you will need to add quotes (`'`) for strings and use comma (`,`) between keys.

```
//```mermaid {theme: 'neutral', scale: 0.8}
graph TD
B[Text] --> C{Decision}
```

```
C -->|One| D[Result 1]
C -->|Two| E[Result 2]
//```
```

Learn more: Demo | Mermaid

Multiple Entries

> Available since v0.15

You can split your `slides.md` into multiple files and organize them as you want.

`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
```

Frontmatter Merging

You can provide frontmatters from both your main entry and external markdown pages. If there are the same keys in them, the ones from the **main entry have the higher priority**. For example

`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
```

They will end up being equivalent of the following page:

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

# Cover

Cover Page
```

Page Reusing

With the multi-entries support, reusing pages could be straightforward. For example:

```
---
src: ./cover.md
---

---
src: ./intro.md
---

---
src: ./content.md
---

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

Go to TOC

Why Slidev

There are a lot of feature-rich, general-purpose, WYSIWYG slides makers like Microsoft PowerPoint and Apple Keynote. They work pretty well for making nice slides with animations, charts, and many other things, while being very intuitive and easy to learn. So why bother making Slidev?

Slidev aims to provide the flexibility and interactivity for developers to make their presentations even more interesting, expressive, and attractive by using the tools and technologies they are already familiar with.

When working with WYSIWYG editors, it is easy to get distracted by the styling options. Slidev remedies that by separating the content and visuals. This allows you to focus on one thing at a time, while also being able to reuse the themes from the community. Slidev does not seek to replace other slide deck builders entirely. Rather, it focuses on catering to the developer community.

Slidev

Here are a few of the coolest features of Slidev:

Markdown-based

Slidev uses an extended Markdown format to store and organize your slides in a single plain text file. This lets you focus on making the content. And since the content and styles are separated, this also made it possible to switch between different themes effortlessly.

Learn more about Slidev's Markdown Syntax.

Themable

Themes for Slidev can be shared and installed using npm packages. You then apply them with only one line of config.

Check out the theme gallery or learn how to write a theme.

Developer Friendly

Slidev provides first-class support for code snippets for developers. It supports both Prism and Shiki to get pixel perfect syntax highlighting, while still being able to modify the code at any time. With Monaco editor built-in, it also empowers you to do live coding / demonstration in your presentation with autocompletion, type hovering, and even TypeScript type check support.

Learn more about highlighters and Monaco configuration.

Fast

Slidev is powered by Vite, Vue 3 and Windi CSS, which give you the most wonderful authoring experience. Every change you made will reflect to your slides **instantly**.

Find more about our tech stack

Interactive & Expressive

You can write custom Vue components and use them directly inside your markdown file. You can also interact with them inside the presentation to express your idea in a more interesting and intuitive way.

Recording Support

Slidev provides built-in recording and camera view. You can share your presentation with your camera view inside, or record and save them separately for your screen and camera. All with one go, no additional tools are needed.

Learn more about recording here.

Portable

Export your slides into PDF, PNGs, or even a hostable Single-page Application (SPA) with a single command, and share them anywhere.

Read more about that in the exporting docs.

Hackable

Being powered by web technologies, anything that can be done in a web app is also possible with Slidev. For example, WebGL, API requests, iframes, or even live sharing. It's up to your imagination!

Give it a Try

Playing around with Slidev will tell you more than a thousand words. You are just one command away:

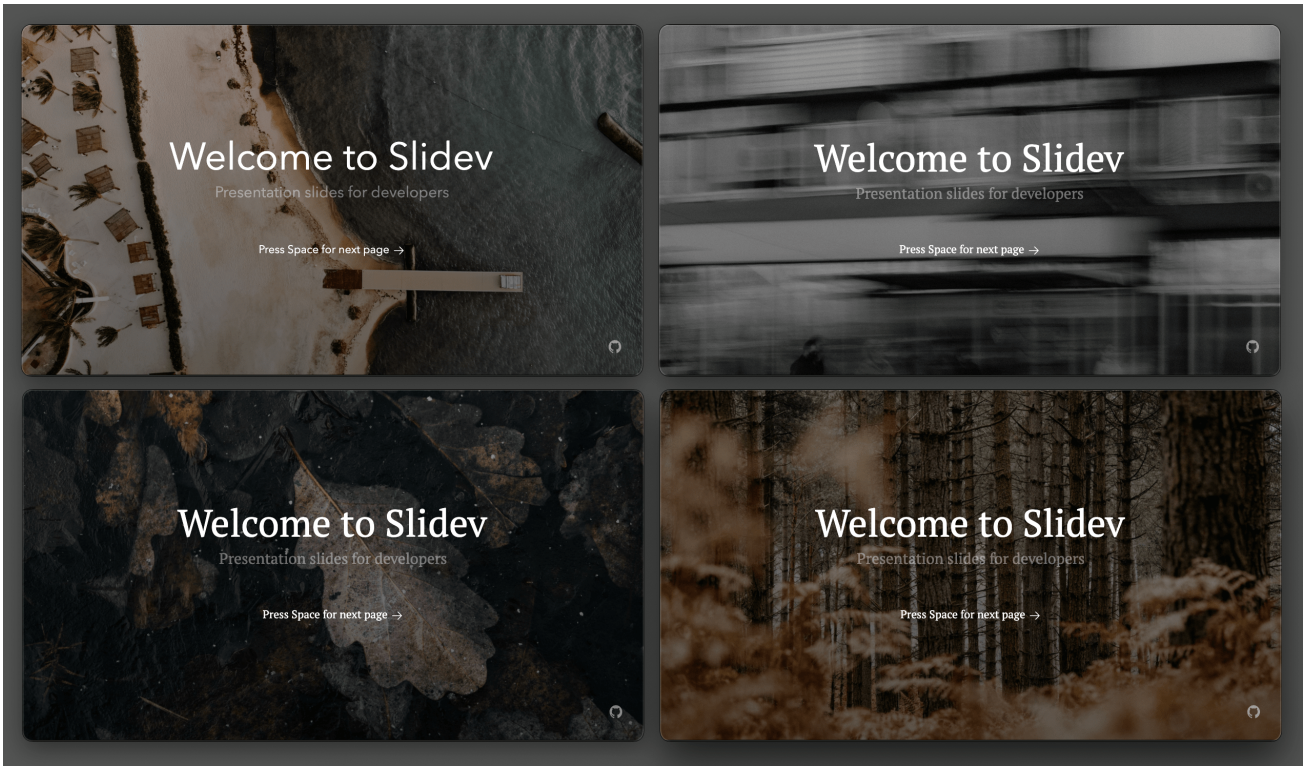
```
$ npm init slidev
```

Or have a quick preview of it:

Go to TOC

Curated Covers

We curated a few cover images to demonstrate our starter template.




```
---
# random image from the curated collection
background: https://source.unsplash.com/collection/94734566/1920x1080
---
```

If you enjoy any of them, check out our Unsplash collection and find out their authors.

Go to TOC

Learning Resources

English

Videos

Articles

- 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

日本語

- 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

Showcases

Talks / Presentations using Slidev.

Go to TOC

Themes Gallery

Browse awesome themes available for Slidev here.

Read more about how to use a theme, or how to write your own and share with the community!

Official Themes

Community Themes

Here are the curated themes made by the community.

More Themes

Find all the themes available on NPM.

Go to TOC

Use Theme

Changing the theme in Slidev is surprisingly easy. All you need to do is to add the `theme:` field in your frontmatter.

```
---
theme: seriph
---
```

You can start the server, which will prompt you to install the theme automatically

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

or install the theme manually via

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

And that's all, enjoy the new theme! For more details about the usage, you can refer to the theme's README.

Want to share your theme? Learn about how to write a theme.

Eject Theme

If you want to get full control of the current theme, you can **eject** it to your local file system and modify it as you want. By running the following command

```
$ slidev theme eject
```

It will eject the theme you are using currently into `./theme`, and changed your frontmatter to

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

This could also be helpful if you want to make a theme based on an existing one. If you do, remember to mention the original theme and the author :)

Local Theme

As you probably found out from the previous section, you can have a local theme for your project. By having a **relative path** in your theme field.

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

Refer to how to write a theme for more details.

Go to TOC

Write a Theme

To get started, we recommend you use our generator for scaffolding your first theme

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

Then you can modify and play with it. You can also refer to the official themes as examples.

Capability

A theme can contribute to the following points:

- Global styles
- Provide default configurations (fonts, color schema, highlighters, etc.)
- Provide custom layouts or override the existing one
- Provide custom components or override the existing one
- Extend Windi CSS configurations
- Configure tools like Monaco and Prism

Conventions

Themes are published to npm registry, and they should follow the conventions below:

- Package name should start with `slidev-theme-`, for example: `slidev-theme-awesome`
- Add `slidev-theme` and `slidev` in the `keywords` field of your `package.json`

Setup

To set up the testing playground for your theme, you can create `example.md` with the following frontmatter, to tell Slidev you are using the current directory as a theme.

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

Optionally, you can also add some scripts to your `packages.json`

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

To publish your theme, simply run `npm publish` and you are good to go. There is no build process required (which means you can directly publish `.vue` and `.ts` files, Slidev is smart enough to understand them).

Theme contribution points follow the same conventions as local customization, please refer to the docs for the naming conventions.

Default Configurations

> Available since v0.19

A theme can provide default configurations via `package.json`.

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

Fonts will be auto imported from Google Fonts.

Learn more about fonts and frontmatter configurations

Theme Metadata

Color Schema

By default, Slidev assumes themes support both light mode and dark mode. If you only want your theme be presented in a designed color schema, you will need to specify it explicitly in `package.json`

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

To access the dark mode when creating your theme styles, you can wrap the dark-mode-specific css inside a `dark` class:

```
/* general css here */

html:not(.dark) {
  /* light mode css here */
}

html.dark {
  /* dark mode css here */
}
```

Slidev toggles a `dark` class on the page's `html` element for switching color schema.

Highlighter

Syntax highlighting colors are also provided in the theme. We support both Prism and Shiki. For more information please refer to the syntax highlighting docs.

You can support either one of them, or both. Refer to the default theme for configurations examples `./styles/code.css` / `./setup/shiki.ts`.

Also, remember to specify the supported highlighters in your `package.json`

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

Slidev Version

If the theme is relying on a specific feature of Slidev that are newly introduced, you can set the minimal Slidev version required to have your theme working properly:

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

If users are using older versions of Slidev, an error will be thrown.

Go to TOC

Colophon

This book is created by using the following sources:

- Slidev - English
- GitHub source: slidevjs/docs
- Created: 2022-11-27
- Bash v5.2.2
- Vivliostyle, https://vivliostyle.org/
- By: @shinokada
- GitHub repo: https://github.com/shinokada/markdown-docs-as-pdf