

SLIDEV Docs - Deutsch



Sliddev

Table of contents

• De - README	4
• De - TRANSLATIONS	5
• Builtin - Components	8
• Builtin - Layouts	10
• Custom - Config katex	14
• Custom - Config mermaid	15
• Custom - Config monaco	16
• Custom - Config shortcuts	18
• Custom - Config vite	19
• Custom - Config vue	21
• Custom - Config windicss	22
• Custom - Directory structure	23
• Custom - Fonts	27
• Custom - Global layers	30
• Custom - Highlighters	32
• Custom - Index	34
• Custom - Vue context	36
• Guide - Animations	38
• Guide - Drawing	42
• Guide - Editors	44
• Guide - Exporting	47
• Guide - Faq	49
• Guide - Hosting	52
• Guide - Index	55
• Guide - Install	58
• Guide - Navigation	61
• Guide - Presenter mode	63
• Guide - Recording	64
• Guide - Syntax	65
• Guide - Why	74
• Resources - Covers	77
• Resources - Learning	78
• De - Showcases	79
• Themes - Gallery	80

- Themes - Use _____ 81
- Themes - Write a /theme _____ 83

sli.dev

Dokumentation für [Slidev](#)

Übersetzungen

	Repo	Seite	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

Server lokal starten

```
npm i -g pnpm
pnpm i
pnpm run dev
```

Und besuche <http://localhost:3000>

Oder installiere die [Vite extension for VS Code](#) um Seite an Seite zu editieren.

Hilf beim Übersetzen

Siehe [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
- `install.md` - Install 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` - Themengallery
- `use.md` - How to use Slidev themes
- `write-a-theme.md` - Write your own theme

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!

Komponenten

Built-in Komponenten

Die Dokumentation ist in diesem Bereich noch in Bearbeitung. Schaue dir doch den [Quellcode](#) selbst an, bevor die Dokumentation fertig ist.

Toc

Fügt ein Inhaltsverzeichnis ein.

Wenn Folien nicht im Inhaltsverzeichnis erscheinen sollen, muss das im Frontmatter der Folie angegeben werden:

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

Titel werden durch die `<Titles>` Komponente angezeigt.

Nutzung

```
<Toc />
```

Parameter:

- `columns (string | number , standard: 1)`: Die Anzahl an Spalten, die gezeigt werden soll
- `listClass (string | string[] , standard: '')`: Klassen, die der dem Inhaltsverzeichnis zugewiesen werden sollen
- `maxDepth (string | number , standard: Infinity)`: Die maximale Tiefe des anzuzeigenden Titels
- `minDepth (string | number , standard: 1)`: Die minimale Tiefe des anzuzeigenden Titels
- `mode ('all' | 'onlyCurrentTree' | 'onlySiblings' , standard: 'all')`:
 - `'all'` : Zeigt alle Einträge
 - `'onlyCurrentTree'` : Zeigt nur Einträge, die sich im aktuellen Pfad befinden (aktive, Eltern und Kinder der Einträge)
 - `'onlySiblings'` : Zeigt nur Einträge aus dem aktuellen Pfad und die direkten Geschwister

Link

Fügt einen Link ein, der zu einer Folie navigiert.

Nutzung

```
<Link to="42">Gehe zur Folie 42</Link>  
<Link to="42" title="Gehe zur Folie 42"/>
```

Parameter:

- `to (string | number)`: Der Pfad der Folie, zu der navigiert werden soll (Folien starten bei `1`)
- `title (string)`: Der anzugebende Titel

Titles

Fügt den Haupttitel einer Folie als geparstes HTML ein.

Titel und Titelebenen werden automatisch aus dem ersten Titelement jeder Folie abgerufen.

```
---  
title: Eine Tolle Folie  
level: 2  
---
```

Nutzung

Die `<Titles>` Komponente ist eine virtuelle Komponente, die wie folgend importiert werden kann:

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

Dann kann man sie nutzen:

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

Parameter:

- `no (string | number)`: Die Nummer der Folie, von der der Titel gezeigt werden soll (Folien starten bei `1`)

Eigene Komponenten

Erstelle einen Ordner `components/` im Ursprung deines Projektverzeichnisses. Dort können eigene Vue Komponenten erstellt werden, die dann ganz einfach in der Markdown Datei genutzt werden können.

Erfahre mehr über [Individualisierung](#)

Komponenten bereitgestellt von Themen

Themen können auch eigene Komponenten bereitstellen. Diese sind in der Dokumentation der jeweiligen Themen aufgelistet und erklärt.

Mehr Informationen über die [Ordner Struktur](#)

[Go to TOC](#)

Layouts

Eingebaute Layouts

Da Themen eventuell Layouts überschreiben können, ist es am besten, die genaue Verwendung, Parameter und Beispiele in der jeweiligen Dokumentation nachzulesen.

center

Zeigt den Inhalt in der Mitte des Bildschirms an.

cover

Wird benutzt, um das Deckblatt der Präsentation anzuzeigen. Diese Folie kann auch den Titel und weitere Kontextualisierungen oder anderes enthalten.

default

Das Standard Layout, um jegliche Art von Inhalt anzuzeigen.

end

Die Abschlussfolie der Präsentation.

fact

Zeigt einen Fakt oder Daten mit sehr viel Aufmerksamkeit an.

full

Nutzt den gesamten Platz auf dem Bildschirm für die Darstellung von Inhalten aus.

image-left

Zeigt ein Bild auf der linken Seite des Bildschirms und rechts den Inhalt.

Verwendung

```
---  
layout: image-left  
  
# die Bildquelle  
image: ./pfad/zum/bild  
  
# ein benutzerdefinierter Klassename für den Inhalt  
class: mein-toller-inhalt-rechts  
---
```

image-right

Zeigt ein Bild auf der rechten Seite des Bildschirms und links den Inhalt.

Verwendung

```
---
layout: image-right
# die Bildquelle
image: ./pfad/zum/bild

# ein benutzerdefinierter Klassename für den Inhalt
class: mein-toller-inhalt-links
---
```

image

Zeigt ein Bild als Hauptinhalt der Seite an.

Verwendung

```
---
layout: image
# die Bildquelle
image: ./pfad/zum/bild
---
```

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

Als Einstieg in die Präsentation, wird normalerweise mit Titel, kurzer Beschreibung, Autor, usw. benutzt.

none

Ein Layout ohne jegliche Styles.

quote

Um ein Zitat an hervorgehobener Stelle anzuzeigen.

section

Wird benutzt, um den Anfang eines neuen Abschnitts in der Präsentation zu markieren.

statement

Mache eine Affirmation/Aussage zum Hauptinhalt der Seite.

two-cols

Trennt den Seiteninhalt in zwei Spalten.

Verwendung

```
---  
layout: two-cols  
---  
  
# Links  
  
Das wird links angezeigt  
::right::
```

Rechts

Das wird rechts angezeigt

Eigene Layouts

Erstelle einen Ordner `layouts/` im Ursprung deines Projektverzeichnisses. Dort können eigene Vue Layout Komponenten erstellt werden.

Erfahre mehr unter [Individualisierung](#)

Layouts bereitgestellt von Themen

Themen können auch eigene Layouts bereitstellen oder bestehende ändern. Diese sind in der Dokumentation der jeweiligen Themen aufgelistet und erklärt.

[Go to TOC](#)

Konfiguriere KaTeX

Erstelle eine `./setup/katex.ts` Datei mit dem folgendem Inhalt:

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

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

Mit dem Setup kann man benutzerdefinierte Einstellungen für [KaTeX](#) bereitstellen. Weitere Informationen sind in den Typdefinitionen und Dokumentationen von [KaTeX](#) zu finden.

Konfiguriere Mermaid

Erstelle eine `./setup/mermaid.ts` Datei mit dem folgendem Inhalt:

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

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

Mit dem Setup kann man benutzerdefinierte Einstellungen für [Mermaid](#) bereitstellen. Weitere Informationen sind in den Typdefinitionen und Dokumentationen von [Mermaid](#) zu finden.

Konfiguriere Monaco

Erstelle eine `./setup/monaco.ts` Datei mit dem folgendem Inhalt:

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

export default defineMonacoSetup(async (monaco) => {
  // nutzte `monaco` zum konfigurieren
})
```

Erfahre mehr über die [Konfiguration von Monaco](#).

Verwendung

Um Monaco in den Folien zu nutzen, muss nur `{monaco}` zu einem Code Snipped hinzugefügt werden.

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

console.log(plusOne.value) // 2

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

zu

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

console.log(plusOne.value) // 2

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

Exportieren

Standardmäßig ist Monaco nur im Development-Modus aktiviert. Damit Monaco auch in einer SPA funktioniert, muss `monaco: true` in den Frontmatter Konfigurationen gesetzt werden:

```
---
monaco: true # standart: "dev"
---
```

Automatische Typen Installation

Wenn TypeScript mit Monaco genutzt wird, werden Typen für Abhängigkeiten automatisch auf der Clientseite installiert.

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

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

Im obigen Beispiel muss man nur sicher gehen, dass `vue` und `@vueuse/core` lokal installiert sind (dependencies oder devDependencies) und Slidev handelt den Rest, damit der Editor funktioniert!

Themen Konfigurieren

Das Thema wird von Slidev basierend auf dem Hell-/Dunkelmodus gesteuert. Wenn man es anpassen möchte, kann die Themen-ID in der Setup-Funktion übergeben werden.

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

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

Wenn man benutzerdefinierte Themen laden möchte:

```
import { defineMonacoSetup } from '@slidev/types'
// ändere dein Thema
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',
    },
  }
})
```

Wenn man ein Thema für Slidev erstellt, kann man den dynamischen `import()` innerhalb der Setup-Funktion nutzen, um bessere Ergebnisse bei der Code-Aufteilung zu erhalten.

Konfiguriere Shortcuts

Available since v0.20

Erstelle eine `./setup/shortcuts.ts` Datei mit dem folgendem Inhalt:

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

export default defineShortcutsSetup((nav: NavOperations) => {
  return [
    {
      key: 'enter',
      fn: () => nav.next(),
      autoRepeat: true,
    },
    {
      key: 'backspace',
      fn: () => nav.prev(),
      autoRepeat: true,
    },
  ],
})
```

Mit diesem Setup können eigene Einstellungen für Tastenkürzel aus [Navigationen](#) definiert werden. Das obige Beispiel führt die nächste Animation aus, wenn `enter` gedrückt wird und die letzte Animation, wenn `backspace` gedrückt wird.

Die Konfigurations-Funktion bekommt ein Objekt mit den Navigationsmethoden und gibt einen Array, welcher die Tastenkürzel und Konfigurationen enthält zurück. Weitere Informationen sind in den Typdefinitionen zu finden.

Siehe [useMagicKeys | VueUse](#) für mehr Informationen über das `keyPressed` Event.

[Go to TOC](#)

Konfiguriere Vite

Slidev läuft mit [Vite](#) unter der Haube. Somit kann das großartige Plugin-System von Vite genutzt werden, um Präsentationen noch besser anzupassen.

Die `vite.config.ts` Datei wird genutzt, falls eine vorhanden ist.

Slidev hat die folgenden Plugins vorkonfiguriert:

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

Erfahre mehr über [Vorkonfigurationen hier](#).

Konfiguriere Interne Plugins

Verfügbar seit v0.21

Um integrierte Plugins zu konfigurieren, muss eine `vite.config.ts` Datei mit folgendem Inhalt erstellt werden. Bitte beachte, dass Slidev einige Vorkonfigurationsoptionen für diese Plugins hat. Diese Verwendung überschreibt einige von den Vorkonfigurationsoptionen, was möglicherweise dazu führen kann, dass die App beschädigt wird. Das ist **eine erweiterte Funktion**, die mit Vorsicht beachtet werden sollte und nur genutzt werden sollte, wenn man weiß, was man tut.

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

Weitere Optionen sind in den [Typdeklarationen](#) zu finden.

[Go to TOC](#)

Konfiguriere Vue

Slidev nutzt [Vue 3](#) um die Präsentation auf der Clientseite zu render. Man kann die App erweitern und benutzerdefinierte Plugins und Extensions nutzen.

Erstelle eine `./setup/main.ts` Datei mit dem folgendem Inhalt:

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

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

Diese Funktion kann auch als Eintrittspunkt der Slidev App genutzt werden, um Initialisierungen durchzuführen, bevor die App startet.

Erfahre mehr: [Vue Application API](#).

Konfiguriere Windi CSS

Markdown unterstützt eingebettete HTML-Markups. So können Inhalte nach Belieben gestylt werden. Der Einfachheit halber haben wir [Windi CSS](#) integriert, sodass dieses Markup direkt mit utility-Klassen gestylt werden kann.

Zum Beispiel:

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

Der [Attributify-Modus](#) in [Windi CSS v3.0](#) ist standardmäßig aktiviert.

Konfigurationen

Erstelle eine `./setup/windicss.ts` Datei mit dem folgendem Inhalt, um Windi CSS zu konfigurieren:

```
// setup/windicss.ts

import { defineWindiSetup } from '@slidev/types'

// Erweiterung der eingebauten Windicss-Konfigurationen
export default defineWindiSetup(() => {
  shortcuts: {
    // eigener Standardhintergrund
    'bg-main': 'bg-white text-[#181818] dark:(bg-[#121212] text-[#ddd])',
  },
  theme: {
    extend: {
      // Schriftarten können hier ersetzt werden, Web-Font-Links müssen der
      // `index.html` Datei hinzugefügt werden.
      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',
      },
    },
  },
})
```

Erfahre mehr über [Windi CSS Konfigurationen](#)

[Go to TOC](#)

Ordner Struktur

Slidev verwendet einige Verzeichnisstruktur-Konventionen, um die Konfigurationsoberfläche zu minimieren und die Erweiterungen flexibel und intuitiv zu gestalten.

Der Grundaufbau ist wie folgt:

```
your-slidev/
  └── components/      # eigene Komponenten
  └── layouts/         # eigene Layouts
  └── public/          # statische Assets
  └── setup/           # eigenes Setup / Hooks
  └── styles/          # eigene Styles
  └── index.html        # injektionen in index.html
  └── slides.md         # Haupt-Eingangspunkt
  └── vite.config.ts    # vite-Konfiguration erweitern
```

Alle sind optional.

Components

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

Komponenten innerhalb dieses Ordners können direkt in den Folien Markdown mit dem gleichen Komponentennamen wie der Dateiname verwendet werden.

Zum Beispiel:

```
your-slidev/
  └── ...
  └── components/
      └── MeineKomponente.vue
      └── HalloWelt.ts
```

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

# My Slide

<MeineKomponente :count="4"/>

<!-- beide Namen funktionieren -->

<hallo-welt foo="bar">
  Slot
</hallo-welt>
```

Diese Funktion wird von `vite-plugin-components` bereitgestellt, erfahre dort mehr darüber.

Slidev stellt auch einige [eingebaute Komponenten](#) zur Verfügung, die man verwenden kann.

Layouts

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

```
your-slidev/
  ...
  └── layouts/
    ├── cover.vue
    └── mein-tolles-thema.vue
```

Man kann einen beliebigen Dateinamen für das Layout verwenden. Verwende das Layout dann in dem YAML-Header über den Dateinamen.

```
---
```

`layout: mein-tolles-thema`

```
--
```

Wenn das eigene Layout den gleichen Namen, wie ein integriertes Layout oder ein Themenlayout hat, dann hat das benutzerdefinierte Layout den Vorrang vor dem Integrierten- oder Themenlayout. Die Prioritätsreihenfolge ist `Lokal > Thema > Integriert`.

Verwende `<slot />` in der Layoutkomponente für den Folieninhalt. Zum Beispiel:

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

Public

Konventionen: `./public/*`

Assets in diesem Verzeichnis werden nach der Umwandlung in eine SPA im Wurzelpfad `/` bereitgestellt und in das Wurzelverzeichnis des Dist-Verzeichnisses kopiert. Lesen mehr über [Vites public -Verzeichnis](#).

Style

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

Dateien, die dieser Konvention folgen, werden in das Stammverzeichnis der App injiziert. Wenn man mehrere css-Einträge importieren möchte, kann man die folgende Struktur erstellen und die Importreihenfolge selbst verwalten.

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

```
// styles/index.ts

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

Stile werden von [Windi CSS](#) und [PostCSS](#) verarbeitet, so dass man css-Schachtelungen und [at-Direktiven](#) out-of-box verwenden kann. Zum Beispiel:

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

Erfahre mehr über den [Syntax](#).

index.html

Konventionen: `index.html`

Die `index.html` bietet die Möglichkeit, Meta-Tags und/oder Skripte in die Haupt-`index.html` zu injizieren.

Zum Beispiel für die folgende eigene `index.html`:

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

Die fertig gehostete `index.html` wird sein:

```
<!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>
```

Globale Ebenen

Konventionen: [global-top.vue](#) | [global-bottom.vue](#)

Erfahre mehr: [Globale Ebenen](#)

[Go to TOC](#)

Schriftarten

Verfügbar seit v0.20

Während man HTML und CSS verwenden kann, um die Schriftarten und den Stil der Folien nach Belieben anzupassen, bietet Slidev auch eine bequeme Möglichkeit, Schriftarten und Stile mühelos zu verwenden.

Konfiguriere den Frontmatter wie folgt:

```
---  
  fonts:  
    # grundsätzlicher Text  
    sans: 'Robot'  
    # Verwendung mit `font-serif` css-Klasse aus windicss  
    serif: 'Robot Slab'  
    # für Code-Blöcke, Inline-Code, etc.  
    mono: 'Fira Code'  
---
```

Und das ist alles.

Schriftarten werden **automatisch von Google Fonts** importiert. Das heißt, man kann alle bei Google Fonts verfügbaren Schriftarten direkt verwenden.

Lokale Schriftarten

Standardmäßig nimmt Slidev an, dass alle über die `fonts`-Konfigurationen angegebenen Schriftarten von Google Fonts stammen. Wenn man lokale Schriftarten verwenden möchte, gibt man `fonts.local` an, um den Auto-Import zu deaktivieren.

```
---  
  fonts:  
    # wie font-family in css kann ein Komma (`, `) genutzt werden um Fallback  
    # Schriftarten anzugeben  
    sans: 'Helvetica Neue, Robot'  
    # "Helvetica Neue" als lokale Schrift markieren  
    local: 'Helvetica Neue'  
---
```

Gewichte & Kursivschrift

Standardmäßig importiert Slidev drei Gewichte `200`, `400`, `600` für jede Schriftart. Man kann diese konfigurieren, indem man:

```
---  
  fonts:  
    sans: 'Robot'  
    # Standard
```

```

weights: '200,400,600'
# importiere kursive Schriftarten, Voreinstellung `false`
italic: false
---
```

Diese Konfiguration gilt für alle Webfonts. Für eine feinere Steuerung der Gewichte, müssen diese manuell mit [HTML](#) und CSS importiert werden.

Fallback Schriftarten

Für die meisten Szenarien muss man nur die "spezielle Schriftart" angeben, und Slidev fügt die Fallback-Schriften automatisch dazu, zum Beispiel:

```

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

wird

```

.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;
}
```

Wenn man die Fallback-Schriftarten deaktivieren möchte, muss das ganze wie folgt konfiguriert werden:

```

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

Anbieter

- Optionen: `google` | `none`
- Standard: `google`

Derzeit wird nur Google Fonts unterstützt, es ist geplant, in Zukunft weitere Anbieter hinzuzufügen. Mit der Angabe "none" wird die Funktion des automatischen Imports vollständig deaktiviert und alle Schriften werden lokal behandelt.

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

Globale Ebenen

Verfügbar seit v0.17

Globale Ebenen erlauben es, Komponenten zu nutzen, die über Folien **bestehend** bleiben. Solche können zu Beispiel für Kopf- oder Fußzeilen, Folienübergreifende Animationen oder Globale Effekte nützlich sein.

Slidev bietet dafür 3 Ebenen. Erstelle eine `global-top.vue`, `global-bottom.vue` oder `custom-nav-controls.vue` Datei unter dem Projektstamm und die Ebenen werden automatisch aufgenommen.

Ebenenbeziehung:

- Global Top (`global-top.vue`)
- Slides
- Global Bottom (`global-bottom.vue`)
- NavControls
 - Individualisierte Navigationsoberfläche (`custom-nav-controls.vue`)

Beispiel

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

Der Text `Dein Name` wird auf allen Folien erscheinen.

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

Der Button `Next` erscheint in der Navigationsoberfläche.

Der [Vue Global Context](#) kann angewendet werden, um diese bedingt zu aktivieren.

```
<!-- Fußzeile von Seite 4 ausblenden -->
<template>
  <footer
    v-if="$slidev.nav.currentPage !== 4"
    class="absolute bottom-0 left-0 right-0 p-2">
    >
    Dein Name
    </footer>
  </template>
```

```
<!-- Fußzeile vom "cover"-Layout ausblenden -->
<template>
  <footer>
    <v-if="$slidev.nav.currentLayout !== 'cover'">
      <div>Dein Name</div>
    </v-if>
  </template>
```

```
<!-- eine Beispiel-Fußzeile für Folien -->
<template>
  <footer>
    <v-if="$slidev.nav.currentLayout !== 'cover'">
      <div>{{ $slidev.nav.currentPage }} / {{ $slidev.nav.total }}</div>
    </v-if>
  </template>
```

```
<!-- custom-nav-controls -->
<!-- verstecke den Button im Präsentationsmodus -->
<template>
  <button v-if="!$slidev.nav.isPresenter" class="icon-btn" title="Next">
    <carbon:arrow-right />
  </button>
</template>
```

Highlighters

Slidev kommt mit zwei Syntax-Highlighter, aus denen man wählen kann:

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

Prism ist einer der beliebtesten Syntax-Highlighter. Die Hervorhebung erfolgt durch Hinzufügen von Token-Klassen zum Code und wird mit CSS gefärbt. Man kann die [offiziellen Themen](#) durchstöbern oder mit `prism-theme-vars` ganz einfach selbst ein Thema erstellen/anpassen.

Shiki hingegen ist ein TextMate-Grammatik-gestützter Syntax-Highlighter. Er generiert farbige Token, so dass kein zusätzliches CSS erforderlich ist. Da Shiki eine großartige Grammatikunterstützung hat, sind die generierten Farben sehr akkurat, fast genau wie das, was in VS Code zu sehen ist. Shiki kommt auch mit [einer Reihe von eingebauten Themen](#). Der Nachteil von Shiki ist, dass es TextMate-Themen (kompatibel mit VS Code-Themen) benötigt, um die Hervorhebung zu machen, was etwas schwieriger anzupassen sein kann.

Slidev-Themen unterstützen normalerweise sowohl Prism als auch Shiki, aber je nach verwendetem Thema kann es sein, dass nur eines davon unterstützt wird.

Falls man die Wahl hat, ist der einzige Unterschied:

- **Prism** für einfachere Anpassung
- **Shiki** für eine genauere Hervorhebung

Standardmäßig verwendet Slidev Prism. Man kan dies aber ändern, indem man den Frontmatter modifiziert:

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

Konfiguriere Prism

Um Prism zu konfigurieren, kann man einfach das Thema-Css importieren oder `prism-theme-vars` verwenden, um Themen für den hellen und dunklen Modus zu konfigurieren. Weitere Details findet man in den entsprechenden Dokumenten.

Konfiguriere Shiki

Erstelle eine `./setup/shiki.ts` Datei mit folgendem Inhalt:

```
/* ./setup/shiki.ts */  
import { defineShikiSetup } from '@sliderv/types'  
  
export default defineShikiSetup(() => {  
  return {  
    theme: {
```

```
        dark: 'min-dark',
        light: 'min-light',
    },
})
```

Die verfügbaren Themennamen findet man in [Shikis Dokumentation](#).

Oder wenn man sein eigenes Thema verwenden möchten:

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

Anpassungen

Slidev ist vollständig anpassbar, vom Styling bis zur Tools-konfiguration. Slidev ermöglicht darunter liegende Tools zu konfigurieren ([Vite](#), [Windi CSS](#), [Monaco](#), etc.)

Frontmatter Konfigurationen

Slidev kann im Frontmatterblock der ersten Folie konfiguriert werden. Folgende Liste zeigt die Standartwerte:

```
---  
# themen-id oder package name  
theme: 'default'  
# Titel der Folie (wird automatisch aus der ersten Überschrift abgeleitet, wenn  
nicht angegeben)  
title: 'Slidev'  
# titleTemplate für die Webseite, `'%s` wird mit dem Folientitel ersetzt  
titleTemplate: '%s - Slidev'  
# information for your slides, can be a markdown string  
info: false  
# erlaube das Herunterladen einer PDF aus der SPA, kann auch ein eigener URL sein  
download: false  
# Dateiname der exportierten Datei  
exportFilename: 'slidev-exported.pdf'  
# Syntaxhervorheber, entweder 'prism' oder 'shiki'  
highlighter: 'prism'  
# Zeilennummern in Codeblöcken anzeigen  
lineNumbers: false  
# Monaco-Editor aktivieren, standardmäßig nur dev  
monaco: 'dev'  
# Remote-Assets lokal mit vite-plugin-remote-assets herunterladen, kann ein  
boolean, 'dev' oder 'build' sein  
remoteAssets: false  
# gibt an, ob Text in den Folien auswählbar ist  
selectable: true  
# Folienaufzeichnung aktivieren, kann ein boolean, 'dev' oder 'build' sein  
record: 'dev'  
  
# Farbschema für die Folien erzwingen, kann 'auto', 'light' oder 'dark' sein  
colorSchema: 'auto'  
# Router-Modus für Vue-Router, entweder "history" oder "hash"  
routerMode: 'history'  
# Seitenverhältnis der Folien  
aspectRatio: '16/9'  
# tatsächliche Breite des Canvases (Einheit in px)  
canvasWidth: 980  
# passe Theme Designs an, fügt Root-Styles `--slidev-theme-x` für das Attribut `x`  
ein  
themeConfig:  
  primary: '#5d8392'  
  
# favicon: Kann ein lokaler Pfad oder eine URL sein.  
favicon: 'https://cdn.jsdelivr.net/gh/slidevjs/slidev/assets/favicon.png'  
  
# URL von einem PlantUML Server, der zum render von Diagrammen genutzt werden soll  
plantUmlServer: 'https://www.plantuml.com/plantuml'
```

```

# Schriften werden automatisch von Google Fonts importiert
# Erfahre mehr: https://de.sli.dev/custom/fonts
fonts:
  sans: 'Roboto'
  serif: 'Roboto Slab'
  mono: 'Fira Code'

# Standard-Frontmatter, gilt für alle Folien
defaults:
  layout: 'default'
  # ...

# Zeichnungsoptionen
# Mehr Informationen: https://sli.dev/guide/drawing.html
drawings:
  enabled: true
  persist: false
  presenterOnly: false
  syncAll: true
---
```

Weitere Informationen sind in den [Typdefinitionen](#) zu finden.

Ordner Struktur

Slidev nutzt Ordner-Struktur-Konventionen, um die Konfigurationsoberfläche minimal zu halten und Erweiterungen flexibel und intuitiv zu gestalten.

Siehe Abschnitt [Ordner Struktur](#). ## Tools Konfigurieren

- [Highlighters](#)
- [Vue konfigurieren](#)
- [Vite konfigurieren](#)
- [Windi CSS konfigurieren](#)
- [Monaco konfigurieren](#)
- [KaTeX konfigurieren](#)
- [Mermaid konfigurieren](#)

Vue Globaler Kontext

Slidev injiziert einen [globalen Vue-Kontext](#) `$slidev` für erweiterte Bedingungen oder Navigationssteuerungen. ## Verwendung

Man kann überall im Markdown- und Vue-Template mit dem "[Mustache"-Syntax](#) darauf zugreifen.

```
<!-- slides.md -->
# Folie 1
Die aktuelle Folie ist: {{ $slidev.nav.currentPage }}

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

Eigenschaften

`$slidev.nav`

Ein reaktives Objekt, das die Eigenschaften und Steuerelemente der Foliennavigation enthält. Zum Beispiel:

```
$slidev.nav.next() // nächster Schritt
$slidev.nav.nextSlide() // nächste Folie (überspringe v-clicks)
$slidev.nav.go(10) // gehe zu Folie #10

$slidev.nav.currentPage // aktuelle Foliennummer
$slidev.nav.currentLayout // aktuelle Layout-ID
$slidev.nav.clicks // aktuelle Anzahl der Klicks
```

Weitere verfügbare Eigenschaften sind in den [nav.ts](#) Exporten zu finden.

`$slidev.configs`

Ein reaktives Objekt, das die geparssten [Konfigurationen im ersten Frontmatter](#) der `slides.md` Datei enthält. Zum Beispiel:

```
---
title: Meine erste Slidev!
---

{{ $slidev.configs.title }} // 'Meine erste Slidev!'
```

\$slidev.themeConfigs

Ein reaktives Objekt, das die geparsten Themenkonfigurationen enthält.

```
---  
title: Meine erste Slidev!  
themeConfig:  
  primary: #213435  
---
```

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

Animationen

Klick Animationen

v-click

Um "Klick-Animationen" zu Elementen hinzuzufügen, können die `v-click`-Direktive oder `<v-click>` Komponente verwendet werden.

```
# Hallo
<!-- Nutzung der Komponente: Der Text ist unsichtbar, bis "Weiter" gedrückt wird --
-->
<v-click>
    Hallo Welt
</v-click>

<!-- Nutzung der Direktive: Der Text ist unsichtbar, bis "Weiter" ein weiteres mal
gedrückt wird -->
<div v-click class="text-xl p-2">
    Hey!
</div>
```

v-after

Die Verwendung von `v-after` ist ähnlich wie `v-click`, aber diesmal wird das Element sichtbar, wenn der Vorherige `v-click` ausgelöst wird.

```
<div v-click>Hallo</div>
<div v-after>Welt</div>
```

Wenn "Weiter" gedrückt wird, werden `Hallo` und `Welt` zusammen sichtbar.

v-click-hide

Funktioniert genau wie `v-click`, nur dass es das Element verschwinden lässt.

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

v-clicks

`v-clicks` wird nur als Komponente bereit gestellt. Es ist eine Abkürzung, um die `v-click`-Direktive auf alle untergeordneten Elemente anzuwenden. Es ist besonders nützlich, wenn man mit Listen arbeitet.

```
<v-clicks>
  - Item 1
```

```
- Item 2
- Item 3
- Item 4

</v-clicks>
```

Jedes Mal, wenn man auf "Weiter" klickt, wird ein Element sichtbar.

Benutzerdefinierte Anzahl der Klicks

Standardmäßig zählt Slidev, wie viele Schritte erforderlich sind, bevor zur nächsten Folie gewechselt wird. Man kann diese Einstellung überschreiben, indem man die Frontmatter-Option `clicks` übergibt:

```
---  
# 10 clicks in der Folie, bevor es zur nächsten geht  
clicks: 10  
---
```

Reihenfolge

Wenn man einen Index der `v-click` Direktive überreicht, kann die Reihenfolge der Enthüllung angepasst werden.

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

```
<!-- Die Reihenfolge ist umgekehrt. -->
<div v-click="3">1</div>
<div v-click="2">2</div>
<div v-click="1">3</div>
```

```
---  
clicks: 3  
---  
  
<!-- Sichtbar nach 3 Klicks -->
<v-clicks at="3">
  <div>Hi</div>
</v-clicks>
```

Element Übergänge

Wenn die `v-click` Direktive bei Elementen angewendet wird, erhalten diese auch die `slidev-vclick-target` CSS Klasse. Wenn das Element auch noch versteckt ist, des weiteren auch die Klasse `slidev-vclick-hidden`. Zum Beispiel:

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

Nach einem Klick ist es:

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

Standardmäßig wird auf diese Klassen ein subtler Deckkraftübergang angewendet:

```
// Standardmäßig:  
  
.slidev-vclick-target {  
  transition: opacity 100ms ease;  
}  
  
.slidev-vclick-hidden {  
  opacity: 0;  
  pointer-events: none;  
}
```

Man kann diese Übergänge in den eigenen Stylesheets überschreiben und verändern.

Zum Beispiel wäre ein Hochskalierender Übergang:

```
// styles.css  
  
.slidev-vclick-target {  
  transition: all 500ms ease;  
}  
  
.slidev-vclick-hidden {  
  transform: scale(0);  
}
```

Animationen können auch nur für bestimmte Folien und Layouts festgelegt werden.

```
.slidev-page-7,  
.slidev-layout.my-custom-layout {  
  .slidev-vclick-target {  
    transition: all 500ms ease;  
  }  
  
  .slidev-vclick-hidden {  
    transform: scale(0);  
  }  
}
```

Weitere Informationen zum [Anpassen von Styles](#)

Bewegungen

Slidev hat [@vueuse/motion](#) integriert. Man kann die `v-motion` Direktive nutzen, um an Elementen Bewegungen anzuwenden. Beispiel:

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

Der Text `Slidev` bewegt sich von `-80px` zu seiner ursprünglichen Position.

Hinweis: Sliddev lädt die nächste Folien für eine bessere Leistung vorab. Dadurch können eventuell einige Animationen schon beginnen. Damit dies verhindert wird, kann das vorab laden abgeschaltet werden:

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

Oder man steuert den Elementlebenszyklus mit `v-if` für eine bessere Kontrolle

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

Mehr erfahren: [Demo](#) | [@vueuse/motion](#) | [v-motion](#) | [Presets](#)

Folienübergänge

Bisher gibt es KEINE integrierte Unterstützung für Folienübergänge. Wir planen Folienübergänge in der nächsten Hauptversion hinzuzufügen. Zuvor können dafür nur benutzerdefinierte Styles und Libraries genutzt werden.

Zeichnung & Anmerkungen

Verfügbar seit v0.23

Wir haben [drauu](#) für Zeichnungen und Anmerkungen eingebaut, damit deine Präsentation noch besser wird.

Um zu starten, muss nur das Icon in der Menüleiste geklickt werden und schon geht's los. Das Zeichnen ist auch im [Präsentatoren Modus](#) verfügbar. Zeichnung & Anmerkungen, die im Präsentationsmodus erstellt werden, werden automatisch in Echtzeit über alle Instanzen **synchronisiert**.

Verwendung mit Stylus Stift

Wenn ein Stylus Stift auf einem Tablet verwendet wird, erkennt Slidev das automatisch. Nun kann man direkt auf die Folien malen, ohne extra den Zeichenmodus zu aktivieren. Die Finger oder Maus können weiterhin zur Navigation genutzt werden.

Zeichnungen Speichern

Mit der folgenden Frontmatter Konfiguration werden Zeichnungen als SVGs im `.slidev/drawings` Ordner gespeichert. Dadurch können diese auch mit PDFs exportiert werden oder sind in der SPA zu sehen.

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

Zeichnungen deaktivieren

Vollständig:

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

Nur während der Entwicklung:

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

Nur im Präsentationsmodus:

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

Zeichnung Synchronisierung

Standartweiße werden alle Zeichnungen aller Instanzen miteinander synchronisiert. Wenn man eventuell seinen Präsentation anderen vorträgt, kann dies stören. Mit folgender Konfiguration kann die Synchronisierung ausgeschaltet werden.

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

Wenn `syncAll` deaktiviert ist, werden trotzdem alle Zeichnungen von Präsentator der synchronisiert.

Editor Unterstützung

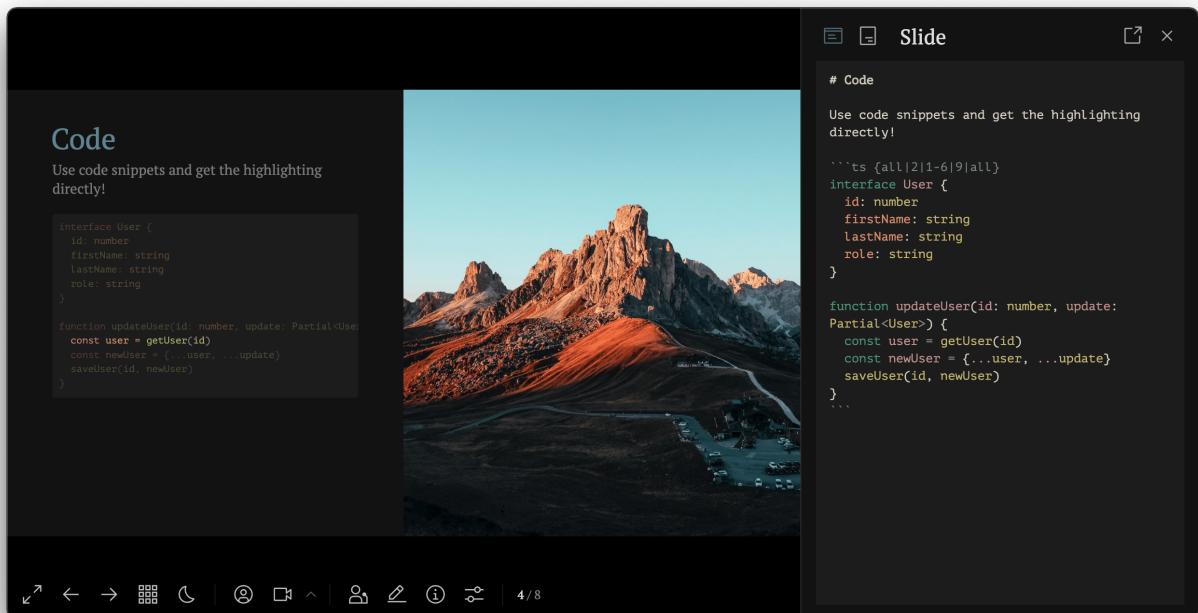
Da Slidev Markdown als Informationsquelle nutzt, kann JEDER Editor genutzt werden.

Für das beste Folienmanagement, bieten wir auch folgende Editor Integrationen:

Integrierter Editor

Slidev wird mit einem integrierten [CodeMirror](#) Editor geliefert, in dem Änderungen an der Präsentation sofort nach dem Speichern sichtbar sind.

Klicke den Button, um den CodeMirror Editor zu öffnen.



VS Code Erweiterung



Slidev for VS Code

VS Code Marketplace v0.4.1 downloads 26k

Die VS Code Erweiterung bietet ein paar Funktionen, die dabei helfen, die Folien besser zu organisieren und einen besseren Überblick zu behalten.

Funktionen

- Folien werden im Seitenbereich angezeigt
- Nächste / Vorherige Folien Buttons
- Folien umsortieren
- Code Folding für Folien Blöcke
- Wandle Markdown in HTML um

The screenshot shows a slide editor interface. On the left is a sidebar with a file tree titled "SLIDEV: SLIDES". The tree includes items like "Composable Vue", "Anthony Fu", "Sponsors", and several sections under "Composition API" such as "Ref", "Ref Auto Unwrapping", "Patterns & Tips", and "What's Composable Func...". On the right is a main editor window titled "slides.md — demo". The code in the editor is:

```
168 # Ref Auto Unwrapping <MarkerCore />
170 Get rid of `value` for most of the time.
<div class="grid grid-cols-2 gap-x-4">
<v-clicks :every='2'>
  - `watch` accepts ref as the watch target, and returns the
    unwrapped value in the callback
```
180 const counter = ref(0)

watch(counter, count => {
 console.log(count) // same as `counter.value`
})
```
  - Ref is auto unwrapped in the template
```
190 <template>
 <button @click="counter += 1">
```

---

[Go to TOC](#)

# Exportieren

## PDF

Exportieren in PDF oder PNGs benötigt [Playwright](#) zum rendern. Dafür muss `playwright-chromium` installiert sein. Falls das exportieren in einer CI Umgebung passiert, kann der [playright CI Guide](#) hilfreich sein.

Installiere `playwright-chromium`

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

Jetzt kann die Präsentation in PDF mit folgenden Befehl exportiert werden:

```
$ slidev export
```

Nach ein paar Sekunden wird die fertige Präsentation unter `./slides-exports.pdf` erscheinen.

Falls du deine Folien mit der dunklen Version des Themes exportieren willst, kannst du die `--dark` option wählen:

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

## Animationsschritte exportieren

Verfügbar seit v0.21

Standardmäßig exportiert Slidev Präsentationen mit einer Seite pro Folie und deaktivierten Klick-Animationen. Wenn die Präsentation mit mehreren Schritten pro Folien exportiert werden soll, muss die `--with-clicks` Option übergeben werden.

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

## PNGs

Wenn die Option `--format png` überreicht wird, exportiert Slidev PNG Bilder für jeden Folie der Präsentation anstatt einer PDF.

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

# Single-Page Application (SPA)

Slidev Präsentationen können auch in eine selbst-hostbare SPA exportiert werden:

```
$ slidev build
```

Die generierte Applikation befindet sich unter `dist/` und kann auf [GitHub Pages](#), [Netlify](#), [Vercel](#), oder wo auch immer man will gehostet werden. Jetzt kann man die Präsentation mit der ganzen Welt in nur einem Link teilen.

## Basispfad

Wenn die SPA auf Unteroutes gehostet werden soll, muss die Option `--base` mit einem Pfad überreicht werden:

```
$ slidev build --base /präsentationen/meine-coole-präsentation/
```

Mehr Details gibt es auf der [Vite Dokumentation](#).

## Herunterladbare PDF bereitstellen

Man kann eine herunterladbare PDF für die Zuschauer der SPA bereitstellen. Mit der folgenden Konfiguration wird diese Funktion aktiviert:

```

download: true

```

Jetzt generiert Slidev eine PDF mit der SPA und ein Herunterladen-Button erscheint in der SPA.

Man kann eine eigene URL zum herunterladen der PDF angeben. In diesem Fall wird das Generieren der PDF übersprungen.

```

download: "https://meineseite.com/meine-präsentation.pdf"

```

## Beispiele

Hier sind ein paar Beispiele einer exportierten SPA:

- [Starter Vorlage](#)
- [Composable Vue von Anthony Fu](#)

Mehr Infos: [Statisches Hosting](#).

---

[Go to TOC](#)

# FAQ

## Grids

Da Slidev Webbasiert ist, können Grid-Layouts genutzt werden. [CSS Grids](#), [Flexboxen](#), oder sogar [CSS Masonry Layouts](#), die volle Kontrolle liegt bei dir!

Da wir Windi CSS integriert haben, geht es ganz einfach, wie hier:

```
<div class="grid grid-cols-2 gap-4">
<div>
 Die erste Spalte
</div>
<div>
 Die zweite Spalte
</div>
</div>
```

Weiter, kann die Größe jeder einzelnen Spalte angepasst werden:

```
<div class="grid grid-cols-[200px,1fr,10%] gap-4">
<div>
 Die erste Spalte (200px)
</div>
<div>
 Die zweite Spalte (auto fit)
</div>
<div>
 Die dritte Spalte (10% Breite des übergeordneten Containers)
</div>
</div>
```

Erfahre mehr über [Windi CSS Grids](#).

## Positionierung

Folien werden mit festen Größen definiert (Standart: `980x552px`) und so skaliert, dass sie mit der Breite des Benutzbildschirms übereinstimmen. Somit können Elemente problemlos absolut positioniert werden, ohne sich Gedanken über das Skalieren zumachen. Da die gesamten Folien skaliert werden, werden auch die einzelnen Elemente mit skaliert.

Zum Beispiel:

```
<div class="absolute left-30px bottom-30px">
 Das ist eine nach unten links ausgerichtete Fußzeile
</div>
```

Um die tatsächliche Größe der Folie zu ändern, kann eine `canvasWidth` Option im ersten Formationsblock hinzugefügt werden:

```

 canvasWidth: 800

```

## Schriftgröße

Falls dir die Schriftgrößen auf den Folien zu klein sind, können diese in folgenden Wegen geändert werden:

### Lokale Styles überschreiben

Man kann die lokale Styles für jede Folie mit einem `<style>` Tag überschreiben.

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

Folie 2
Diese Folie wird nicht beeinflusst.
```

Erfahre mehr über [Embedded Styles](#)

### Globale Styles überschreiben

Man kann eigene globale Styles in der `./style.css` Datei bereitstellen, zum Beispiel:

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

Erfahre mehr über [Globale Styles](#)

### Den Canvas skalieren

Das Ändern der tatsächlichen Größe des Canvases skaliert alle Inhalte der Folien (Text, Bilder, Komponenten, ...) und Folien.

```

standart: 980
da der Canvas kleiner wird, wird die visuelle Größe größer
canvasWidth: 800

```

## Transform verwenden

Wir bieten eine integrierte Komponente `<Transform />`, welcher ein Wrapper der CSS `transform` Eigenschaft ist.

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

# Statisches Hosting

## Erstellen von Single Page Applications (SPA)

Man kann Slidev Präsentation als selbst-hostbare SPA exportieren:

```
$ slidev build
```

Die erstellte SPA ist im `dist/` Ordner verfügbar und kann mit [GitHub Pages](#), [Netlify](#), [Vercel](#) oder mit was auch immer man will gehostet werden. Nun kann man seine Präsentation mit nur einem link mit der ganzen Welt teilen.

### Basispfad

Wenn Präsentationen auf sub-Routen bereit gestellt werden soll, kann eine `--base` Option übergeben werden. Zum Beispiel:

```
$ slidev build --base /slides/mein-toller-vortrag/
```

Weitere Einzelheiten sind in [Vite's Dokumentation](#) zu finden.

### Herunterladbare PDF

Man kann herunterladbare PDFs für Zuschauer der SPA mit der folgenden Konfiguration bereitstellen:

```

download: true

```

Slidev generiert automatisch mit der SPA eine PDF und in der SPA wird ein 'Download' Button angezeigt.

Man kann auch eine eigene PDF zum herunterladen anbieten, in diesem Fall wird das generieren der PDF übersprungen.

```

download: "https://meine-seite.de/vortrag.pdf"

```

## Beispiele

Hier sind einige Beispiele für die exportierte SPA:

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

Mehr ist in den [Beispielprojekten](#) zu finden.

# Hosting

Wir empfehlen die Verwendung von `npm init slidev@lastest`, um das Projekt zu erstellen, somit werden die notwendigen Konfigurationsdateien für die Hosting-Dienste direkt mitgeliefert.

## Netlify

- [Netlify](#)

Erstelle eine `netlify.toml` im Projektstamm mit folgendem Inhalt:

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

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

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

Gehe dann zu deinem Netlify Dashboard und erstelle eine neue Website mit dem Repository.

## Vercel

- [Vercel](#)

Erstelle eine `vercel.json` im Projektstamm mit folgendem Inhalt:

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

Gehe dann zu deinem Vercel Dashboard und erstelle eine neue Seite mit dem Repository.

## GitHub Pages

- [GitHub Pages](#)

Erstelle eine `.github/workflows/deploy.yml` im Projektstamm mit folgendem Inhalt um deine Präsentation via Github Actions auf Github Pages bereitzustellen:

```
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: Build
 run: npm run build
- name: Deploy pages
 uses: crazy-max/ghaction-github-pages@v2
 with:
 build_dir: dist
 env:
 GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

# Erste Schritte

## Überblick

Slidev (slide + dev, `/slaidiv/`) ist ein Web-basiertes Tool zum Erstellen und Präsentieren von Präsentationen. Slidev wurde für Entwickler entwickelt und setzt den Fokus auf das Verfassen von Inhalten in Markdown, während HTML und leistungsfähige Vue Komponenten pixelgenaue Designs und Layouts mit eingebetteten interaktiven Demos für deine Präsentation liefern.

Slidev verwendet eine funktionsreiche Markdown Datei um schöne Folien mit sofortigem Wiederladeerlebnis und vielen anderen Integrationen, wie Live-Coding, PDF-Export oder Präsentationsaufzeichnung zu generieren. Da Slidev über Web-Technologien läuft ist so gut wie alles möglich - die Möglichkeiten sind endlos.

Weitere Informationen über den Hintergrund von Slidev sind im [Warum Slidev](#) Abschnitt zu finden.

## Funktionen

-  **Markdown-basiert** - benutze deinen Lieblingseditor und Arbeitsablauf
-  **Entwickler freundlich** - integrierte Syntax Hervorhebung, live Coding, etc.
-  **Anpassbarer Stil** - Themen können geteilt und via npm packages genutzt werden
-  **Stylish** - Windi CSS on-demand utilities und einfach zu benutzende, eingebettete Stylesheets
-  **Interaktiv** - eigene Vue Komponenten
-  **Moderatoren Modus** - nutze ein anderes Fenster oder deine Handy um deine Präsentation zu steuern
-  **Drawing** - draw and annotate on your slides
-  **LaTeX** - integrierte Unterstützung für mathematische LaTeX Gleichungen
-  **Diagramme** - erstelle Diagramme mit Beschriftungen
-  **Icons** - direkter Zugriff auf Icons von jedem Iconset
-  **Editors** - integrierter Editor, oder [extension für VS Code](#)
-  **Aufzeichnungen** - integrierte Aufnahme und Kameraansicht
-  **Portabel** - exportiere in PDF, PNG oder hostbare SPA
-  **Blitzschnell** - sofortiges Nachladen durch [Vite](#)
-  **Hackbar** - nutze Vite plugins, Vue Komponenten, oder ein beliebiges npm package

## Tech-Stack

Slidev wird durch die Kombination dieser Tools und Technologien ermöglicht:

- [Vite](#) - Ein extrem schnelles Frontend-Tool
- [Vue 3](#) powered [Markdown](#) - Konzentriere dich auf den Inhalt, während die Power von HTML und Vue Komponenten immer an deiner Seite ist
- [Windi CSS](#) - On-Demand Utility-First CSS Framework, style deine Folien mit einer Lichtigkeit
- [Prism](#), [Shiki](#), [Monaco Editor](#) - Erstklassige Code-Snippet-Unterstützung mit Live-Coding Fähigkeit
- [RecordRTC](#) - Integrierte Aufnahmeunterstützung mit Kameraansicht
- [VueUse](#) Familie - `@vueuse/core`, `@vueuse/head`, `@vueuse/motion`, etc.
- [Iconify](#) - Iconset Sammlung.

- [Drauu](#) - Drawing and annotations support
- [KaTeX](#) - LaTeX Formeln rendern.
- [Mermaid](#) - Textbasierte Diagramme.

## Die 1. Präsentation

### Online ausprobieren

[sli.dev/new](https://sli.dev/new)



[Open in StackBlitz](#)

### Lokal erstellen

Mit NPM:

```
$ npm init slidev
```

Mit Yarn:

```
$ yarn create slidev
```

Folge den Anweisungen und beginne mit dem Erstellen der Folien. Lies dir die [Syntaxanleitung](#) für mehr Informationen über den Syntax durch.

## Command Line Interface

In einem Projekt, wo Slidev installiert ist, kann Slidev über den `slidev` Befehl in den npm Skripten verwendet werden.

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

Andernfalls kann Slidev mit `npx` verwendet werden.

```
$ npx slidev
```

Führe `slidev --help` aus, um alle verfügbaren Optionen zu erhalten.

## Markdown Syntax

Slidev liest die `slides.md` Datei in deinem Projektstamm und wandelt sie in eine Präsentation um. Bei Änderungen wird deine Präsentation sofort aktualisiert. Ein Beispiel:

```
Slidev

Hallo Welt

Seite 2

Codeblöcke zum direkten hervorheben verwenden

//`ts console.log('Hallo, Welt!') //`

Seite 3
```

Erfahre mehr über [Slidev's Markdown Syntax](#).

# Installation

## Starter Vorlage

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

Der beste Weg, um loszulegen, ist mit unserer offiziellen Starter Vorlage.

Mit NPM:

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

Mit Yarn:

```
$ yarn create slidev
```

Folge den Anweisungen und die Präsentation öffnet sich auf <http://localhost:3030/> automatisch.

Die Vorlage beinhaltet eine grundlegende Einrichtung und eine kleine Demo mit einer Anleitung für die ersten Schritte mit Slidev.

## Manuell installieren

Wenn man Slidev trotzdem lieber manuell installieren oder mit bereits existierenden Projekten integrieren möchte, kann man auch folgendes tun:

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

```
$ touch slides.md
```

```
$ npx slidev
```

Bitte beachte, dass wir `pnpm` nutzen. `Shamefully-hoist` muss aktiviert sein, damit alles problemfrei funktioniert:

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

## Global installieren

Verfügbar seit v0.14

Slidev kann mit folgendem Befehl auch global installiert werden:

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

Danach kann `slidev` überall genutzt werden ohne, dass jedes mal erst ein Projekt erstellt werden muss.

```
$ slidev
```

Dieser Befehl versucht auch die lokale Version `@slidev/cli` zu nutzen, wenn sie im `node_modules` Ordner gefunden werden kann.

## Auf Docker installieren

Wenn ein schneller Weg benötigt wird, um Präsentationen auch auf Containern zu nutzen, kann das bereits fertige [Docker Image](#) verwendet werden, welches von [tangramor](#) gemaintained wird. Oder man baut sich sein eigenes Image.

Man muss nur folgenden Befehl im Projektordner ausführen:

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

Wenn dein Projektordner leer ist, wird eine `slides.md` Vorlage und andere benötigte Dateien generiert und ein Server auf dem Port `3030` gestartet.

Jetzt kannst du deine Präsentation hier finden: <http://localhost:3030/>

## Baue ein deploy-bares Image

Oder du kreierst dein eigenes Projekt in einem Docker Image mit einer Dockerfile:

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

Baue das Image: `docker build -t meinePraesentation .`

Und Starte den Container: `docker run --name praesentation --rm --user node -p 3030:3030 meinePraesentation`

Nun findest du deine Präsentation unter <http://localhost:3030/>

## Baue eine hostbare SPA (Single Page Application)

Führe den Befehl `docker exec -i slidev npx slidev build` an einem Container aus, in dem `slidev` läuft. Er wird die statischen HTML Dateien und den `dist` Ordner generieren,

## Hosten auf Github Pages

Du kannst den `dist` Ordner als eine statische Webseite mit [Github Pages](#) or Gitlab Pages hosten.

Weil der Github Pages URL eventuell einen Unterordner enthält, muss die generierte `index.html` angepasst werden. Entweder änderst du `href="/assets/xxx"` zu `href="./assets/xxx"` oder du übergibst eine `--base=/<subfolder>/` Option dem Build-Befehl, zum Beispiel: `docker exec -i slidev npx slidev build --base=/slidev_docker/`.

Damit der Jekyll Bauprozess übersprungen wird, musst du eine leere `.nojekyll` Datei erstellen.

## Mit Docker Hosten

Man kann die Webseite auch selber über Docker hosten:

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

Oder baue ein statische Image mit der folgenden Dockerfile:

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

Baue das Docker Image: `docker build -t mystaticppt .`

Und starte de Container: `docker run --name myslides --rm -p 80:80 mystaticppt`

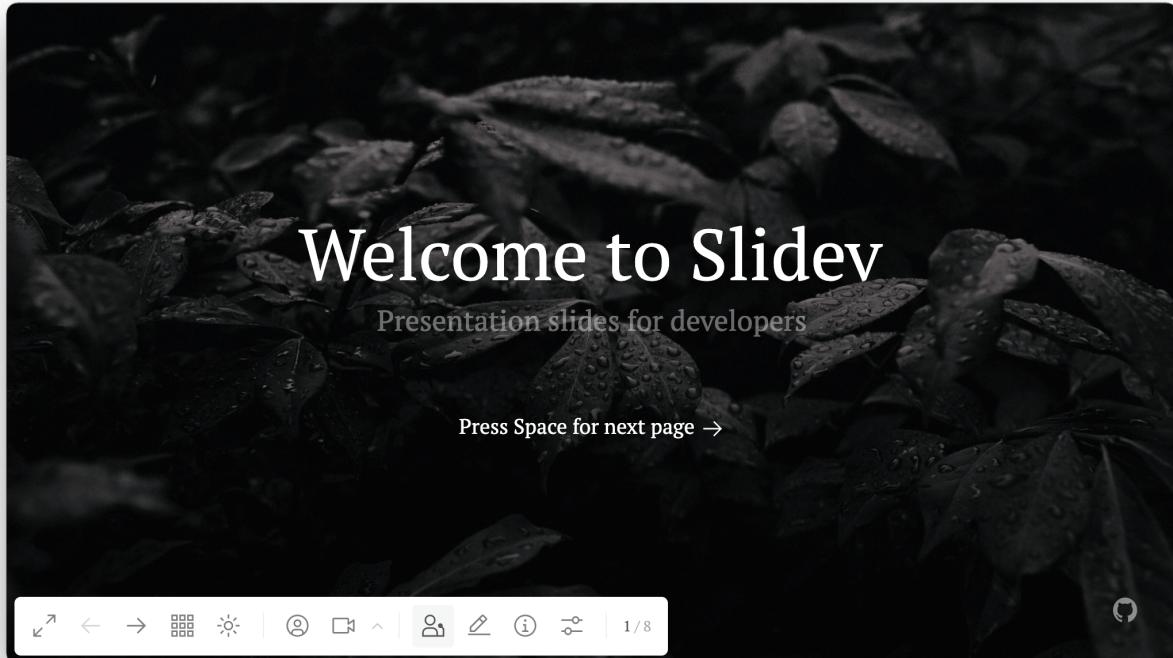
Deine Webseite findest du hier: <http://localhost/>

Bei [tangramor/slidev\\_docker](#) findest du mehr Informationen.

# Navigation

## Navigationsleiste

Bewege deine Maus zur linken unteren Ecke der Slidev Seite, damit die Navigationsleiste erscheint.

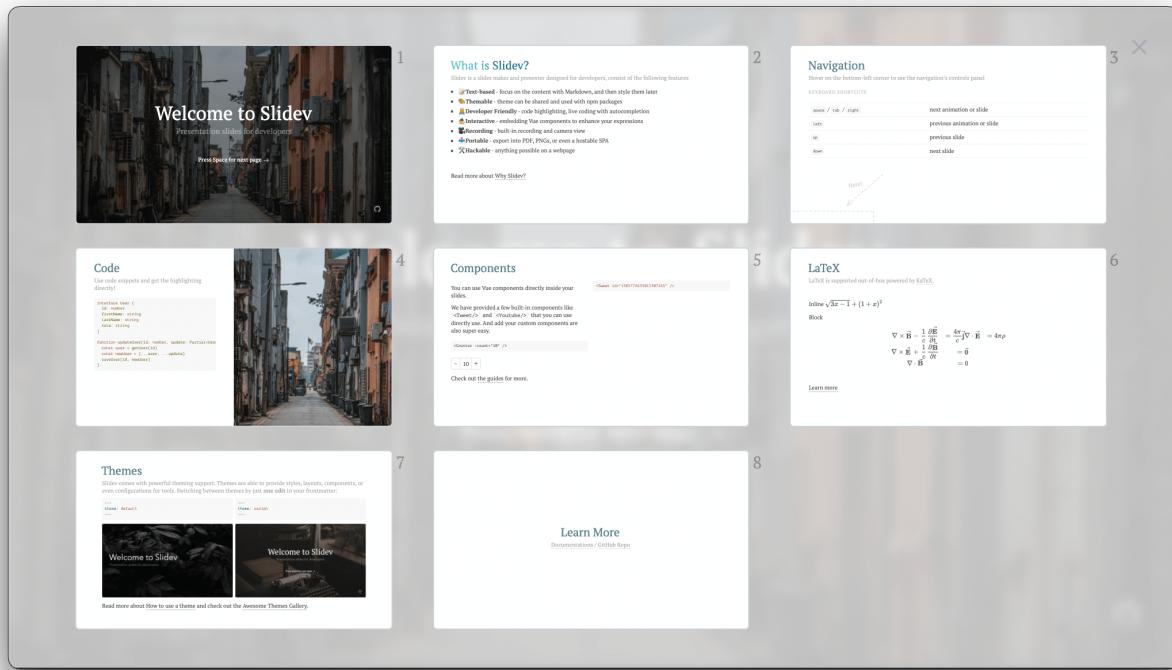


Tastenkombination	Schaltfläche	Beschreibung
f		Vollbild umschalten
rechts / Leertaste		nächste Animation oder Folie
links		vorherige Animation oder Folie
hoch	-	vorherige Folie
runter	-	nächste Folie
o		Folienübersicht umschalten
d		Darkmode umschalten
-		Kamerabild umschalten
-		Präsentation aufnehmen
-		in den Moderatoren Modus wechseln

Tastenkombination	Schaltfläche	Beschreibung
-		Integrierten Editor umschalten
-		Präsentation herunterladen (nur in SPA Version sichtbar)
-		Informationen über die Präsentation anzeigen
-		Einstellungen öffnen
g	-	zeige "Gehe zu ..."

## Folienübersicht

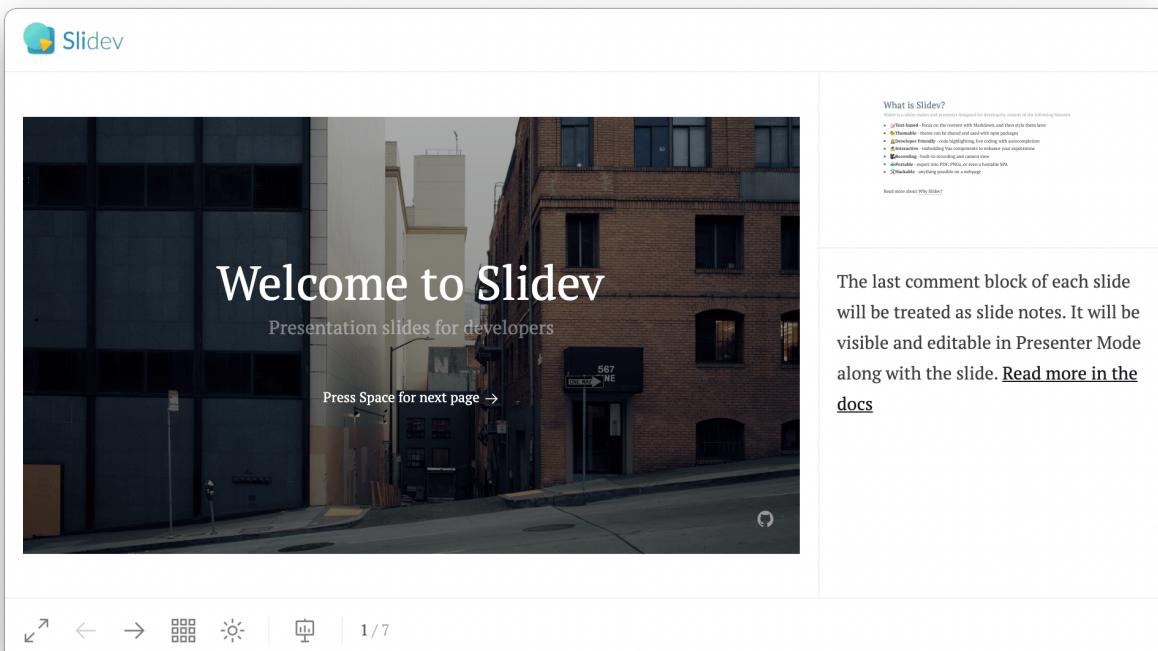
Durch drücken der o Taste oder des Buttons in der Navigationsleiste, kann die Folienübersicht geöffnet werden. Dort kann man problemlos zwischen Folien hin und her wechseln.



[Go to TOC](#)

# Moderatoren Modus

Durch klicken des Buttons in der Navigationsleiste wird der Moderatoren Modus gestartet. Alternativ kann man auch direkt <http://localhost:3030/presenter> manuell besuchen. Wenn der Moderatoren Modus geöffnet ist, bleiben andere Seiteninstanzen automatisch synchronisiert.



---

[Go to TOC](#)

# Präsentation aufzeichnen

Slidev hat eine integrierte Aufnahme- und Kameraansicht. Diese kann genutzt werden, um die eigenen Präsentationen aufzuzeichnen. Alles ganz einfach an einem Ort.

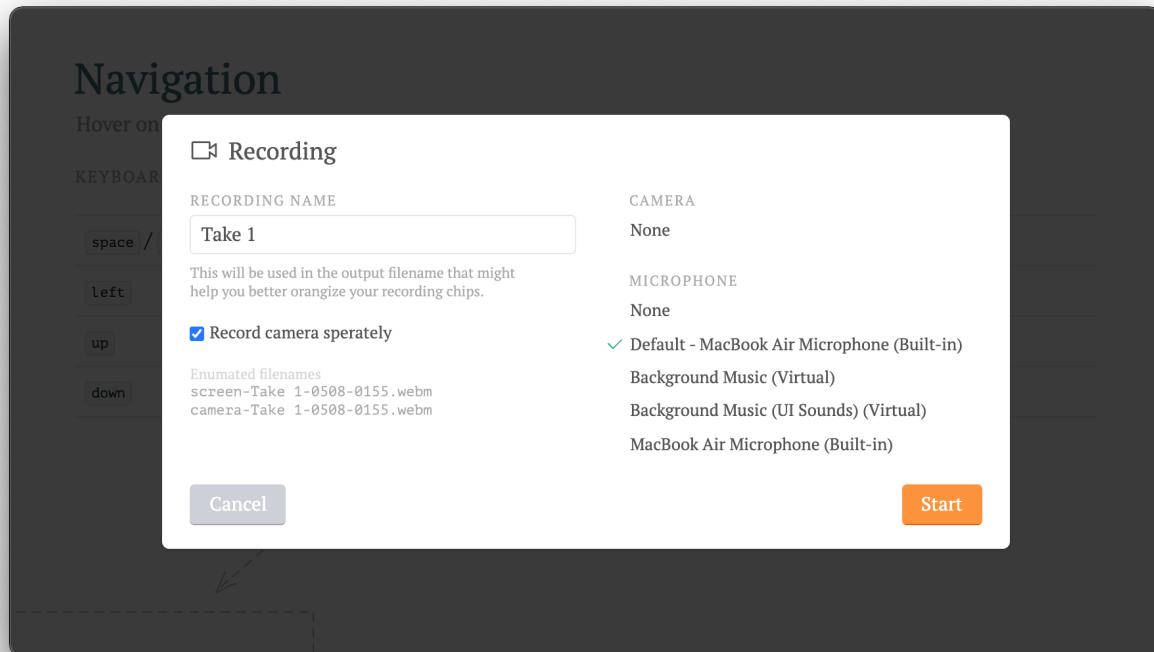
## Kameraansicht

Klicke den Button in der Navigationsleiste um die Kameraansicht in der Präsentation zu öffnen. Die Kameraansicht kann auf dem Bildschirm umher geschoben werden und mittels des Handlers in der rechten unteren Ecke in der Größe verändert werden. Die Größe und Position wird im `localStorage` gespeichert und bei den nächsten Präsentation erneut angewendet.

## Aufzeichnung

Klicke den Button in der Navigationsleiste und es öffnet sich ein Dialog. Hier kann man zwischen der Aufnahme der Präsentation oder des Kamerabilds oder beidem entscheiden. Außerdem kann man angeben, ob beide Videos in eine Datei oder in 2 getrennte Dateien gespeichert werden soll.

Diese Funktion läuft über [RecordRTC](#) und nutzt die [WebRTC API](#).




---

[Go to TOC](#)

# Markdown Syntax

Alle Folien der Präsentation sind in einer **einzelnen Markdown Datei** untergebracht (Standartweiße `./slides.md`).

Es können alle [Markdown Funktionen](#) wie normal genutzt werden. Zusätzlich können auch noch HTML und Vue Komponenten oder Styles mit der Hilfe von [Windi CSS](#) genutzt werden. Folien sind durch `---` zusammen mit einer neuen Zeile getrennt.

```
Slidev

Hallo, Welt!

Folie 2

Codeblöcke zum direkten hervorheben verwenden

//```ts
console.log('Hallo, Welt!')
//```

Folie 3

Nutze Windi CSS und Vue Komponenten um deine Folien zu stylen.

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

## Titelseite & Layouts

Layouts und andere Metadaten können für Folien mit dem Umwandeln der Trennzeichen in [Frontmatter-Blöcke](#) angegeben werden. Jeder Frontmatter-Block startet mit einem Dreifachstrich (`---`) und ended mit einem weiterem. Texte dazwischen sind Datenobjekte im [YAML](#) Format. Zum Beispiel:

```

layout: cover

Slidev

Das ist die Titelseite.

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

Folie 2
```

Diese Folie hat ein layout `center` und ein Hintergrundbild.

---

# Folie 3

Das ist eine Standartfolie ohne weiteren Metadaten.

Schau unter [Individualisierung](#) für mehr Informationen nach.

## Codeblöcke

Ein großer Grund, warum ich Slidev entwickle, ist damit mein Code auf den Folien gut aussieht. So, wie erwartet, können Codeblöcke mit Markdown-Flavor genutzt werden, damit der Syntax ordentlich hervorgehoben wird.

```
//``ts
console.log('Hallo, Welt!')
//``
```

Wir unterstützen [Prism](#) und [Shiki](#) zu Syntaxhervorhebung. Mehr Informationen gibt es im [Abschnitt Highlighters](#).

## Zeilen Hervorhebung

Um bestimmte Zeilen im Code hervorzuheben, muss man nur die Zeilen, die hervorgehoben werden sollen, in geschwungenen Klammern (`{}`) notieren. Zeilen werden von 1 aufwärts gezählt.

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

Um die Hervorhebung in mehreren Schritten zu ändern, können mehrere Angaben mit `|` getrennt angegeben werden. Zum Beispiel:

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

Hier wird zuerst `a: Ref<number> | number` und `b: Ref<number> | number`, einen Klick später `return computed(() => unref(a) + unref(b))` und nach dem letzten Klick wird der ganze Block hervorgehoben. Erfahre mehr über [Klicks und Animationen](#).

## Monaco Editor

Wenn man Änderung am Code während der Präsentation machen möchte, kann der Monaco Editor genutzt werden. Ein einfaches `{monaco}` nach der Programmiersprache und der Codeblock wandelt sich in einen komplett ausgestatteten Monaco Editor um!

```
//``ts {monaco}
console.log('Hallo, Welt!')
//``
```

Wie man [Monaco konfiguriert](#).

## Embedded Styles

Man kann den `<style>` Tag nutzen, um direkt in der Markdowndatei Styles für die **aktuelle Folie** zu überschreiben.

```
Das ist Rot.

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

Die nächste Folie wird nicht beeinflusst
```

Der `<style>` Tag ist immer [bereichsbezogen](#). Mehr über das Überschreiben von globalen Styles gibt es im Abschnitt [Individualisierung](#)

Durch [Windi CSS](#), kann direkt verschachteltes CSS oder [Direktiven](#) (z.B. `@apply`) genutzt werden.

```
Slides

> Hallo, `Welt!`

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

## Statische Assets

Wie man es auch in Markdown machen würde, kann man Bilder mit der Hilfe von remoten oder lokalen Urls definieren.

Für remote Assets kann man das integrierte `vite-plugin-remote-assets` Plugin nutzen. Dieses Plugin speichert Bilder direkt auf der Festplatte, sodass selbst große Bilder sofort laden.

! [Remotes Bild](https://sli.dev/favicon.png)

Lokale Assets können direkt im `public` Ordner abgelegt werden und mit **führendem Schrägstrich** genutzt werden.

! [Lokales Bild](pic.png)

Falls man eigene Styles auf Bilder anwenden möchte, kann man den Markdown in einen `<img>` Tag umwandeln.

``

## Notizen

Für jede Folie kann man Notizen anlegen. Diese erscheinen dann im [Moderatoren Modus](#), damit man sie in den Präsentationen nutzen kann.

In Markdown wird der letzte Kommentar in einer Folie in eine Notiz umgewandelt.

```

layout: cover

Folie 1

Das ist die Titelseite.

<!-- Das ist eine Notiz -->

Folie 2

<!-- Das ist keine Notiz, weil es vor dem Inhalt der Folie steht -->

Die 2. Folie

<!--
Das ist eine weitere Notiz
-->
```

## Icons

In Slidev können fast alle Open-Source Iconsets dank `vite-plugin-icons` und [Iconify direkt](#) in der Markdown Datei genutzt werden.

Die Benennung folgt den [Iconify](#) Namenskonventionen `{iconset-name}-{icon-name}`. Zum Beispiel:

- `<mdi-account-circle />` - von [Material Design Icons](#)
- `<carbon-badge />` - von [Carbon](#)
- `<uim-rocket />` - von [Unicons Monochrome](#)
- `<twemoji-cat-with-tears-of-joy />` - von [Twemoji](#)
- `<logos-vue />` - von [SVG Logos](#)

- Und viele mehr...

Alle verfügbaren Icons können mit [Icônes](#) durchsucht werden.

## Icons stylen

Icons können genau, wie alle anderen HTML Elemente gestylt werden:

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

## Slots

Verfügbar seit v0.18

Einige Layouts können mithilfe von [Vue's benannten Slots](#) mehrere Basispunkte bereitstellen.

Zum Beispiel, im `two-cols` Layout hat man zwei Spalten, links (`default` Slot) und rechts (`right` slot), nebeneinander.

```

layout: two-cols

<template v-slot:default>
 # Links
 Dieser Text steht links

 </template>
 <template v-slot:right>
 # Rechts
 Dieser Text steht rechts

 </template>
```

## Links

Dieser Text steht links

## Rechts

Dieser Text steht rechts

Wir bieten außerdem abgekürzten Syntax Zucker `::name::` für Slot Namen. Das folgende Beispiel funktioniert genau das wie das letzte.

```

layout: two-cols

Links
Dieser Text steht links

::right::

Rechts
Dieser Text steht rechts
```

Der Standart-Slot kann explizit und in eigener Reihenfolge angegeben werden

```

layout: two-cols

::right::

Rechts
Dieser Text steht rechts

::default::

Links
Dieser Text steht links
```

## Konfigurationen

Alle benötigten Konfigurationen können in der Markdown Datei definiert werden. Zum Beispiel:

```

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

Sliderv
Das ist die Titelfolie.
```

Erfahre mehr über [Frontmatter-Konfigurationen](#)

## LaTeX

Slidev kommt mit out-of-box LaTeX Unterstützung von [KaTeX](#).

### Inline

Umgebe den LaTeX Syntax mit einem einzelnen `$` auf jeder Seite für das rendern in der Zeile.

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

## Block

Nutze zwei ( `$$` ) für das Rendern im Block. Dieser Modus nutzt größere Symbole und zentriert den Text.

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

Erfahre mehr: [Demo](#) | [KaTeX](#) | [markdown-it-katex](#)

## Diagramme

Man kann außerdem Diagramme / Graphen aus Textbeschreibungen in der Markdowndatei mit der Hilfe von [Mermaid](#) erstellen.

Codeblöcke, welche mit `mermaid` markiert sind, werden in Diagramme umgewandelt:

```
//```mermaid
sequenceDiagram
 Alice->John: Hallo John, wie geht es dir?
 Note over Alice,John: Eine typische Konversation.
//````
```

Des Weiteren kann ein Objekt übergeben werden, dass Optionen wie Skalierung oder Themierung definiert. Der Syntax dieses Objekts ist ein Javascript-Objektliteral. Für String müssen Anführungszeichen ( `'` ) genutzt werden und Kommas ( `,` ) zwischen Keys.

```
//```mermaid {theme: 'neutral', scale: 0.8}
graph TD
 B[Text] --> C{Entscheidung}
 C -->|Eins| D[Ergebnis 1]
 C -->|Zwei| E[Ergebnis 2]
//````
```

Erfahre mehr: [Demo](#) | [Mermaid](#)

# Mehrere Dateien

Verfügbar seit v0.15

Man kann die Hauptdatei (`slides.md`) in mehrere Dateien aufteilen.

`slides.md` :

```
Folie 1
Das ist eine normale Folie.

src: ./subpage2.md

<!-- Diese Folie wird von './subpage2.md' geladen. -->
geschriebene Inhalte werden ignoriert
```

`subpage2.md` :

```
Folie 2
Diese Folie ist von einer anderen Datei
```

## Frontmatter Zusammenführung

Man kann Formatter von der Hauptdatei oder anderen Markdownseiten nutzen. Falls mehrere gleiche Attribute enthalten sind, werden die Attribute der **Einstiegsdatei** genutzt, da diese immer die **höhere Priorität** hat. Zum Beispiel:

`slides.md` :

```

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

```

`cover.md` :

```

layout: cover
background: https://sli.dev/foo.png

Titelseite
Titelseite
```

Diesen Folien werden genau, wie die folgenden Aussehen:

```

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

Titelseite

Titelseite
```

## Folienwiederverwendung

Mit Mehrfach-Eintrag Unterstützung ist das Wiederverwenden von Folien super einfach:

```

src: ./cover.md

src: ./intro.md

src: ./content.md

Wiederverwenden
src: ./content.md

```

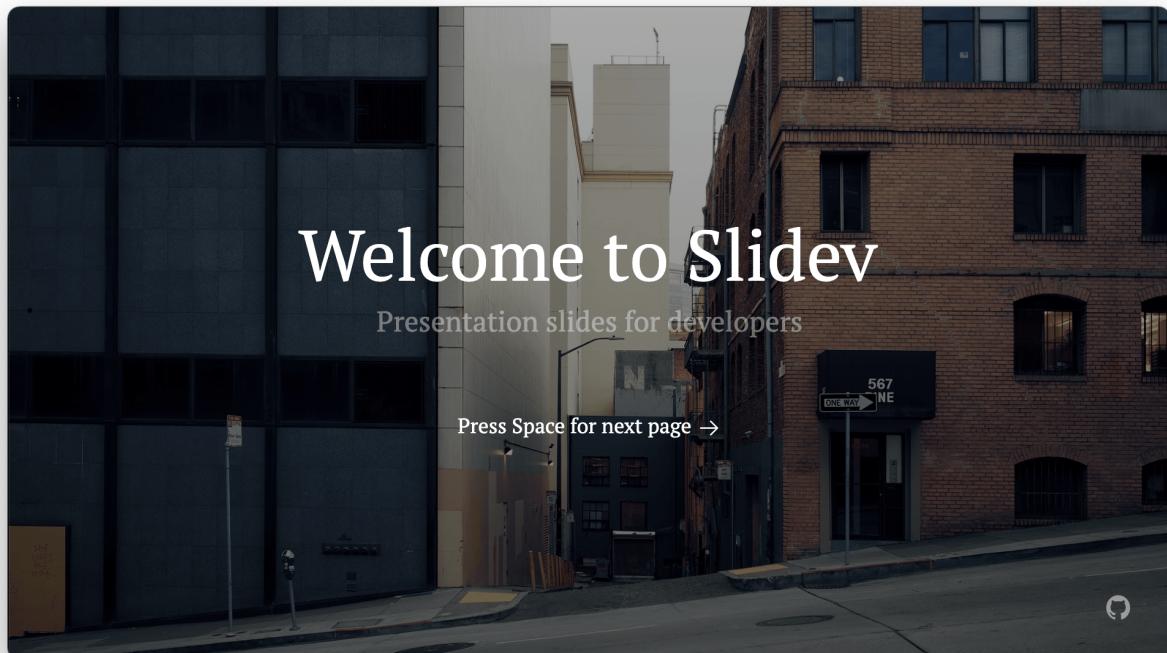
# Warum Slidev

Es gibt eine Menge funktionsreiche, universelle, WYSIWYG (man kriegt was man sieht) Programme zum erstellen von Präsentationen, wie zum Beispiel [Microsoft PowerPoint](#) und [Apple Keynote](#). Mit ihnen kann man super Präsentationen mit Animationen, Diagrammen und vielen anderen tollen Funktionen erstellen. Dazu sind sie noch intuitiv und einfach zu lernen. Warum sollte man sich da die Mühe machen und Slidev entwickeln?

Slidev versucht Entwicklern Flexibilität und Interaktivität zu bieten und hilft ihnen dabei ihre Präsentationen noch interessanter, ausdrucks voller und attraktiver zu gestalten, indem man die Technologien und Tools zur Verfügung stellt, die sie bereits kennen.

Bei der Arbeit mit WYSIWYG Programmen kann man sich schnell in den ganzen Styling-Optionen verlieren. Slidev eliminiert solche Ablenkungen, indem Inhalt und Styling strikt getrennt sind. Dadurch kann man sich besser auf nur eine Sache konzentrieren und gleichzeitig Themen aus der Community wiederverwenden. Slidev versucht nicht alle anderen Programme zu ersetzen, sondern konzentriert sich vielmehr darauf, die Entwickler Community zu bedienen.

## Slidev



Hier sind ein paar Slidev's bester Funktionen:

## Markdown-basiert

Slidev nutzt ein erweitertes Markdown Format, um deine Folien in einer einzelnen Textdatei zu organisieren und speichern. Dadurch kann man sich allein auf den Inhalt der Folien konzentrieren. Da Inhalt und Styles voneinander getrennt sind, kann man auch super einfach zwischen verschiedenen Themen wechseln.

Erfahre mehr über [Slidev's Markdown Syntax](#).

## Anpassbarer Stil

Themen für Slidev können geteilt und via npm packages installiert werden. Installierte Themen können mit nur einer Konfigurationszeile verwendet werden.

Besuche die [Themengalerie](#) oder erstelle [dein eigenes Thema](#).

## Entwickler freundlich

Slidev liefert erstklassige Unterstützung für Code Blöcke. Slidev unterstützt [Prism](#) und [Shiki](#) für perfekte Syntax Hervorhebung, während der Code die ganze Zeit editierbar bleibt. Der [Monaco Editor](#) mit Autovervollständigung, Typprüfung und sogar Typescript Unterstützung bildet eine einwandfreie Basis für live Coding oder Demonstrationen in deiner Präsentation.

Lerne mehr über [Highlighters](#) und die Konfiguration des [Monaco Editors](#).

## Blitzschnell

Slidev läuft über [Vite](#), [Vue 3](#) und [Windi CSS](#). Diese Tools sorgen für eine wundervollste Autorenerfahrung. Jede Änderung wird **sofort** auf den Folien sichtbar.

Entdecke unseren [Tech-Stack](#)

## Interaktiv & Ausdrucksvoll

Erstelle deine eigenen Vue Komponenten und nutze sie direkt in deiner Markdown Datei. Außerdem kann mit den Komponenten in der Präsentation interagiert werden. Damit kann man seine Ideen in der Präsentation auf eine noch interessantere und intuitivere Art und Weise rüber bringen.

## Aufnahmeunterstützung

Slidev bietet eine integrierte Aufnahme und Kameraansicht. Teile deine Präsentation mit deinem Kamerabild oder zeichne deine Präsentation auf und speichere deinen Bildschirm und Kameraaufzeichnung separat. Alles ist mit einem Handgriff erledigt.

Erfahre mehr über [Aufzeichnungen](#).

## Portabel

Exportiere deine Präsentation mit einem einzigen Befehl in PDF, PNG oder sogar hostbare Single-Page-Anwendungen (SPA) und teile sie überall.

Mehr über das [Exportieren von Präsentationen](#)

## Hackbar

Angetrieben von Webtechnologien ist alles, was im Web möglich ist, auch mit Slidev möglich. Zum Beispiel WebGL, API-Anfragen, iframes, oder sogar Live-Sharing. Der Fantasie sind keine Grenzen gesetzt.

## Versuche es doch mal

Slidev selbst auszuprobieren und etwas herumzuspielen erzählt mehr als tausend Wörter. Alles nur ein Befehl entfernt:

```
$ npm init slidev
```

Oder sieh dir eine kurze Vorschau an:

Slidev First Preview Demo

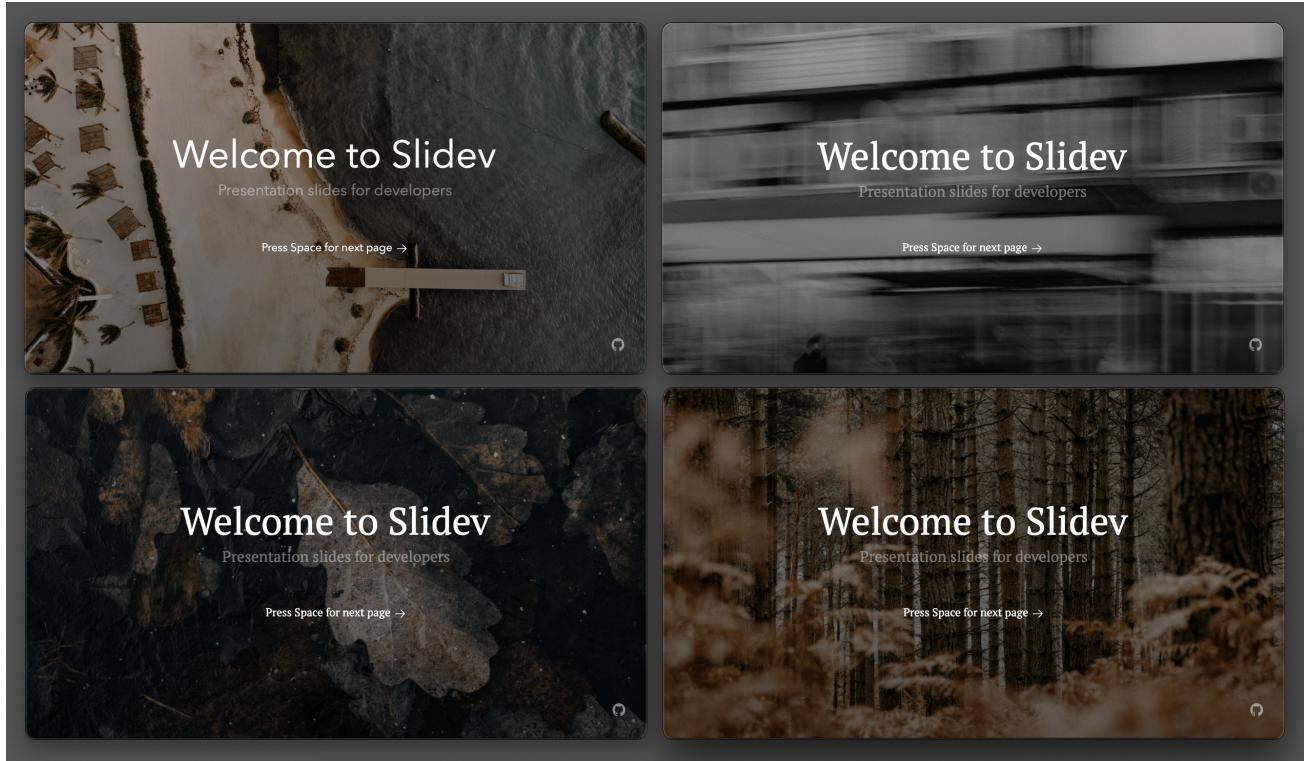


---

[Go to TOC](#)

# Gesammelte Titelbilder

Wir haben ein paar Titelbilder zusammengesammelt, um unsere Starter-Vorlage zum demonstrieren.



```

zufälliges Bilder aus der Sammlung
background: https://source.unsplash.com/collection/94734566/1920x1080

```

Hier ist unsere [Unsplash Sammlung](#) mit Bildern und deren Autoren, falls dir welche von den Bildern gefallen.

---

[Go to TOC](#)

# Lernressourcen

## English

### Videos

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



### Artikel

- [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](#)

# Beispielprojekte

Vorträge und Präsentationen mit Sliderv.

---

[Go to TOC](#)

# Themengallerie

Hier kann man sich verschiedene Themen (Designvarianten) aussuchen.

Hier kan man auch darüber lesen, [wie man eine Thema benutzt](#), oder [wie man sein eigenes Thema entwirft](#) und dann mit Anderen teilt.

## Offizielle Themen

## Community Themen

Hier sind ein paar auserwählte Themen, die von der Community entworfen wurden.

## Mehr Themen

Man kann alle restlichen Themen direkt auf [NPM](#) finden.

# Ein Thema Benutzen

Das Thema auswählen ist in Slidev echt einfach. Es genügt das Hinzufügen von `theme:` zum Beginn des Dokumentes:

```

theme: serif

```

Wenn man den Server startet, fragt Slidev einen automatisch, ob man das ausgewählte Thema installieren möchte:

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

Man kann aber auch ein Thema manuell installieren:

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

Das wars! Viel spaß beim Benutzen deines neuen Themas!

Für mehr Informationen bezüglich des Benutzens eines Themas ist es am Besten, sich an das README des jeweiligen Themas zu richten.

Wollen Sie ihr eigenes Thema entwerfen? Dann können Sie [lernen, wie man ein Thema entwirft](#).

## Thema "Eject"-en

Sollten man die volle Kontrolle über das aktuelle Thema haben wollen, dann kann man das Thema mithilfe des folgendem Befehles in das lokale Dateisystem kopieren:

```
$ slidev theme eject
```

Jetzt befindet sich das Thema in `./theme`.

Die Referenz zum Thema, die sich in der Hauptdatei des Projektes befindet, wird automatisch aktualisiert.

```

theme: ./theme

```

Dieses Feature kann besonders hilfreich sein, wenn man ein Thema entwerfen möchte, das auf einem Anderen basiert. In dem Fall aber nicht vergessen, dem Originalautor seine Bemühungen zuzuschreiben!

## Locales Thema

Wie der vorherige Abschnitt es verraten hat, kann man ein Thema im lokalen Dateisystem benutzen, indem man ein relativer Pfad im Referenzfeld benutzt:

```

theme: ./pfad/zum/thema

```

Weitere details dazu befinden sich unter [wie man ein Thema entwirft](#).

# Erstelle ein Thema

Wir empfehlen unseren Generator zu nutzen, um ein Gerüst für das Thema zu erstellen:

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

Jetzt kann man alles verändern und mit dem code spielen. Man kann sich auch auf Beispiele von den [Offiziellen Themen](#) beziehen.

## Möglichkeiten

Ein Thema kann folgende Punkte anpassen:

- Globale Styles
- Standardkonfigurationen bereitstellen (Schriften, Farbschema, Highlighters, etc.)
- benutzerdefinierte Layouts bereitstellen oder vorhandene überschreiben
- benutzerdefinierte Komponenten bereitstellen oder vorhandene überschreiben
- Windi CSS-Konfigurationen erweitern
- Tools wie Monaco und Prism konfigurieren

## Konventionen

Themen werden im npm Registry veröffentlicht und sollten folgenden Konventionen folgen:

- Der Paketname sollte mit `slidev-theme-` beginnen, zum Beispiel: `slidev-theme-awesome`.
- `slidev-theme` und `slidev` sollte in den `keywords` der `package.json` Datei enthalten sein.

## Einrichtung

Um den Testspielplatz für das Thema einzurichten, kann man eine `example.md` Datei mit folgendem Inhalt erstellen, damit Slidev weiß, dass dieser Ordner als Thema genutzt werden soll:

```

theme: ./

```

Optional, kann man Skripte zur `package.json` Datei hinzufügen.

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

Um das Thema zu veröffentlichen, muss nur der `npm publish` Befehl ausgeführt werden. Es wird kein Build-Prozess benötigt (man kann die `.vue` und `.ts` Datei direkt veröffentlichen, Slidev ist clever genug, um das Thema zu verstehen).

Thema Kontribution folgt den selben Punkten, wie die Konventionen bei lokalen Anpassungen, in der [Dokumentation](#) steht mehr über Namenskonventionen.

## Standardkonfigurationen

Verfügbar seit v0.19

Ein Thema kann [Standardkonfigurationen](#) in der `package.json` Datei bereitstellen.

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

Schriftarten werden automatisch von [Google Fonts](#) importiert.

Erfahre mehr über [Schriftarten](#) und [Frontmatter konfigurationen](#)

## Thema-Metadaten

### Farbschema

Standartmäßig geht Slidev davon aus, dass das Thema Dark- und Lightmode unterstützt. Wenn man möchte, dass das Thema in einem bestimmten Farbschema präsentiert wird, kann dies in der `package.json` Datei angegeben werden.

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

Alle Styles, die Darkmode spezifisch sind, können in einer `dark` CSS Klasse untergebracht werden:

```
/* Allgemeines css hier */

html:not(.dark) {
 /* Lightmode CSS hier */
}

html.dark {
 /* Darkmode CSS hier */
}
```

Slidev schaltet die `dark` CSS Klasse am `html` Element der Seite um, um das Farbschema zu wechseln.

## Highlighter

Syntaxhervorhebungs-Farben können auch im Thema bereit gestellt werden. Wir unterstützen sowohl [Prism](#) als auch [Shiki](#). Weitere Informationen sind in den [Syntax-Highlighting Dokumentationen](#).

Dein Thema kann entweder einen der beide oder beide unterstützen. Konfigurationsbeispiele sind im Standard-Thema `./styles/code.css` / `./setup/shiki.ts` zu finden.

Denke auch daran, unterstützte Highlighters in der `package.json` Datei anzugeben.

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

## Slidev Version

Falls das Thema auf einer bestimmten Slidev Funktion basiert, kann die minimale Slidev Version angegeben werden:

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

Falls Benutzer eine ältere Version von Slidev nutzen, wird ein Fehler ausgeworfen.

# Colophon

This book is created by using the following sources:

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