Welcome to Slidev

Presentation slides for developers

Press Space for next page →

What is Slidev?

Slidev is a slides maker and presenter designed for developers, consist of the following features

- 📝 **Text-based** - focus on the content with Markdown, and then style them later
- 🎨 **Themable** - themes can be shared and re-used as npm packages
- 🧑‍💻 **Developer Friendly** - code highlighting, live coding with autocompletion
- 🤹 **Interactive** - embed Vue components to enhance your expressions
- 🎥 **Recording** - built-in recording and camera view
- 📤 **Portable** - export to PDF, PPTX, PNGs, or even a hostable SPA
- 🛠️ **Hackable** - virtually anything that's possible on a webpage is possible in Slidev

Read more about Why Slidev?

Navigation

Hover on the bottom-left corner to see the navigation's controls panel, learn more

Keyboard Shortcuts

`right` / `space`	next animation or slide
`left` / `shiftspace`	previous animation or slide
`up`	previous slide
`down`	next slide

Table of contents

You can use the `Toc` component to generate a table of contents for your slides:

```
<Toc minDepth="1" maxDepth="1"></Toc>
```

The title will be inferred from your slide content, or you can override it with `title` and `level` in your frontmatter.

Code

Use code snippets and get the highlighting directly, and even types hover!

```ts
// TwoSlash enables TypeScript hover information
// and errors in markdown code blocks
// More at https://shiki.style/packages/twoslash

import { computed, ref } from 'vue'

const count = ref(0)
const doubled = computed(() => count.value * 2)

doubled.value = 2
```

```
Cannot assign to 'value' because it is a read-only property.
```

```ts
// Inside ./snippets/external.ts
export function emptyArray<T>(length: number) {
  return Array.from<T>({ length })
}
```

Learn more

Shiki Magic Move

Powered by shiki-magic-move, Slidev supports animations across multiple code snippets.

Add multiple code blocks and wrap them with ````md magic-move```` (four backticks) to enable the magic move. For example:

```
// step 1
const author = reactive({
  name: 'John Doe',
  books: [
    'Vue 2 - Advanced Guide',
    'Vue 3 - Basic Guide',
    'Vue 4 - The Mystery'
  ]
})
```

Components

You can use Vue components directly inside your slides.

We have provided a few built-in components like `` `<Tweet/>` `` and `` `<Youtube/>` `` that you can use directly. And adding your custom components is also super easy.

```
<Counter :count="10" />
```

Check out the guides for more.

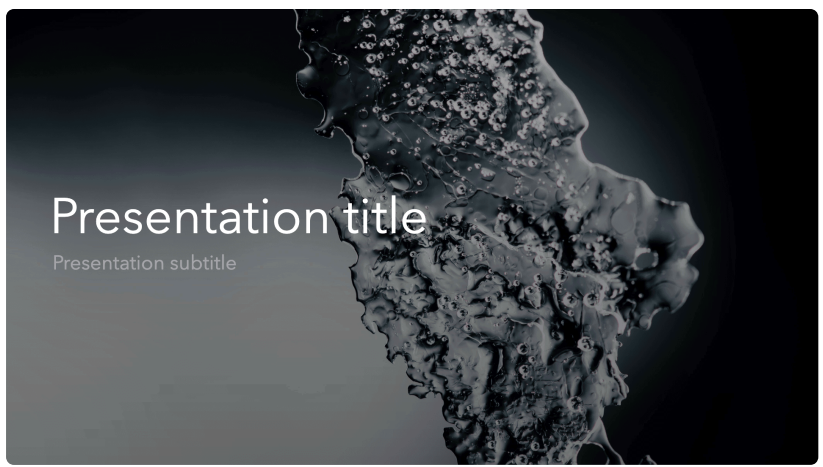
```
<Tweet id="1390115482657726468" />
```

Themes

Slidev comes with powerful theming support. Themes can provide styles, layouts, components, or even configurations for tools. Switching between themes by just **one edit** in your frontmatter:

```yaml
---
theme: default
---
```

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



Read more about How to use a theme and check out the Awesome Themes Gallery.

Clicks Animations

You can add `v-click` to elements to add a click animation.

This shows up when you click the slide:

```
<div v-click>This shows up when you click the slide.</div>
```

The `v-mark` directive also allows you to add inline marks, powered by Rough Notation:

```
<span v-mark.underline.orange>inline markers</span>
```

Learn more

Motions

Motion animations are powered by @vueuse/motion, triggered by `v-motion` directive.

```
<div
  v-motion
  :initial="{ x: -80 }"
  :enter="{ x: 0 }"
  :click-3="{ x: 80 }"
  :leave="{ x: 1000 }"
>
  Slidev
</div>
```

Slidev

Learn more

LaTeX

LaTeX is supported out-of-box. Powered by KaTeX.

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

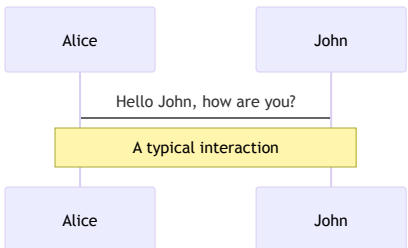
Block

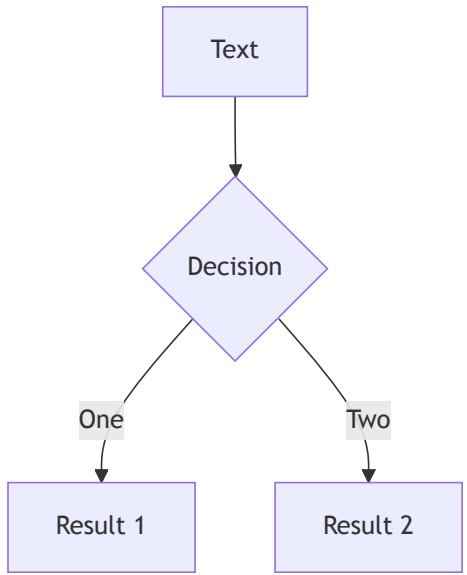
$$\nabla \cdot \vec{E} = \frac{\rho}{\varepsilon_0}$$
$$\nabla \cdot \vec{B} = 0$$
$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}$$
$$\nabla \times \vec{B} = \mu_0\vec{J} + \mu_0\varepsilon_0\frac{\partial \vec{E}}{\partial t}$$

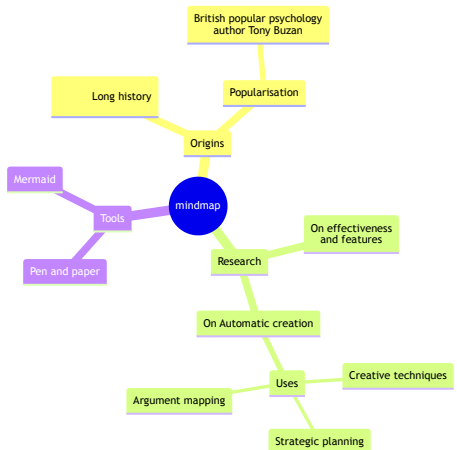
Learn more

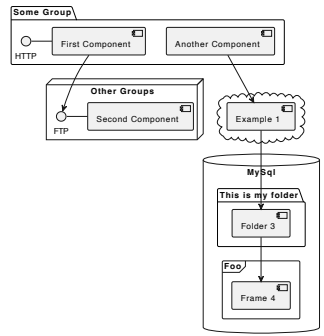
Diagrams

You can create diagrams / graphs from textual descriptions, directly in your Markdown.









Learn more: Mermaid Diagrams and PlantUML Diagrams

Draggable Elements

Double-click on the draggable elements to edit their positions.

Directive Usage

```
<img v-drag="'square'" src="https://sli.dev/logo.png">
```

Component Usage

```
<v-drag text-3xl>
  <carbon:arrow-up />
  Use the `v-drag` component to have a draggable container!
</v-drag>
```

Draggable Arrow

```
<v-drag-arrow two-way />
```

Imported Slides

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

`slides.md`

```
# Page 1

Page 2 from main entry.

---

## src: ./subpage.md
```

`subpage.md`

```
# Page 2

Page 2 from another file.
```

Monaco Editor

Slidev provides built-in Monaco Editor support.

Add `` `{monaco}` `` to the code block to turn it into an editor:

```ts
import { ref } from 'vue'
import { emptyArray } from './external'

const arr = ref(emptyArray(10))
```

Use `` `{monaco-run}` `` to create an editor that can execute the code directly in the slide:

```ts
import { version } from 'vue'
import { emptyArray, sayHello } from './external'

sayHello()
console.log(`vue ${version}`)
console.log(emptyArray<number>(10).reduce(fib => [...fib, fib.at(-1)! + fib.at(-2)!], [1, 1]))
```

```
vue 3.5.13
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
```

Learn More

Documentation · GitHub · Showcases

Powered by Slidev

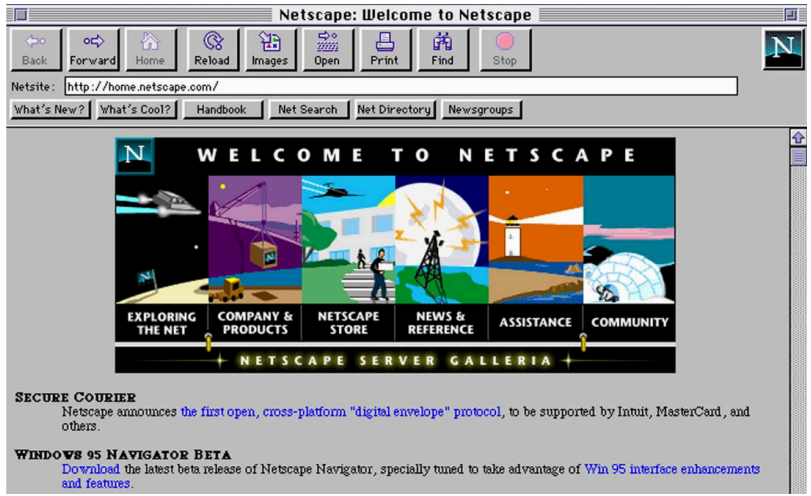
Javascript history

Back to 90's, à la création d'internet

Javascript history

Marc Andressen, créateur de Netscape.

Javascript history



Netscape, un navigateur assez complet (texte, vidéo, son, plugin, images, ...)

Novateur ! A l'époque, les navigateurs sont majoritairement texte.

Javascript history

Netscape gagne en popularité, gagne en part de marché et devient le navigateur le plus utilisé.

Javascript history

Bill Gates sent le filon et commence également à mettre de l'internet partout...

"Pourquoi ne pas moi aussi créer mon navigateur ?"

Javascript history

Microsoft souhaite rachteter Netscape… Qui ne se laisse pas avoir !

Ils décident alors de développeur leur navigateur, qu'ils appeleront…

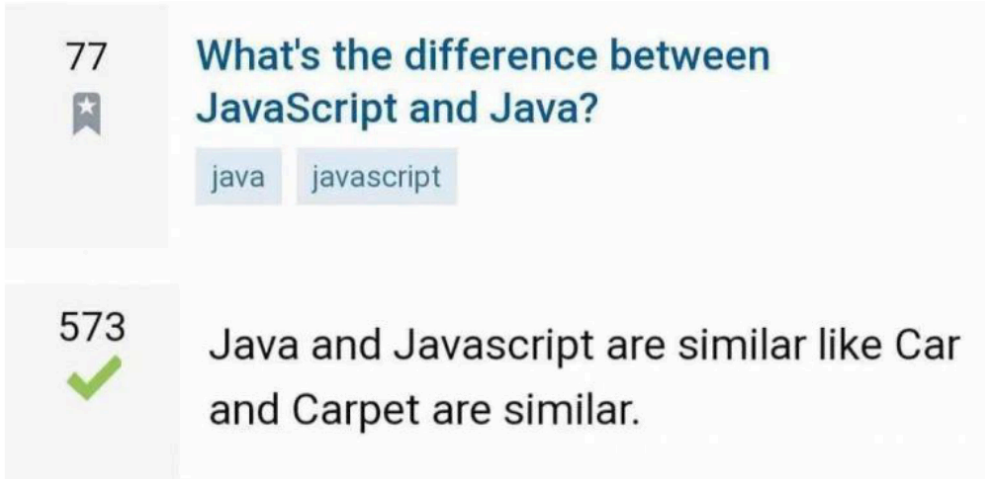
Javascript history

Javascript history

Alerte : Microsoft a énormément de part de marché avec ses PC...

Netscape accélère, les idées fusent : "rendont notre navigateur programmable !" \ Brendan Eich est embauché, un crack de l'IT.

Javascript history

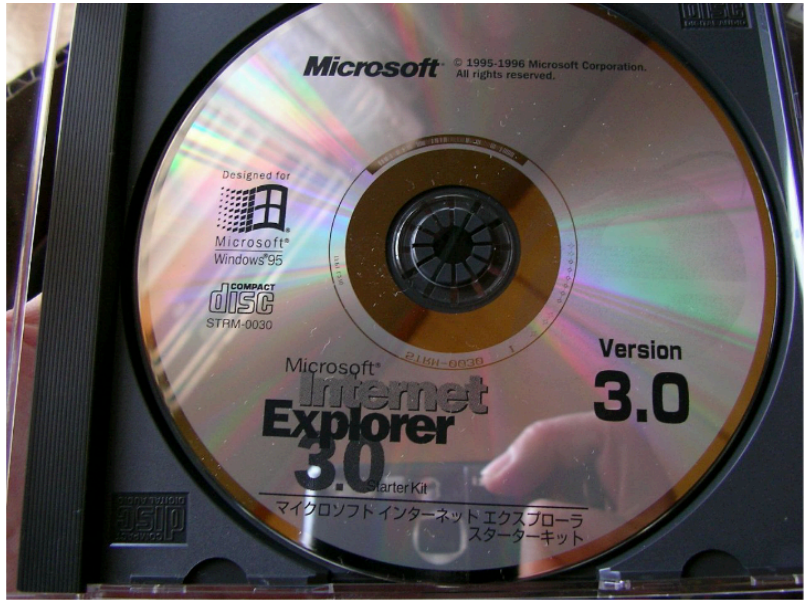


Mocha sera renommé JavaScript car censé être un compagnon de Java côté navigateur…

C’est un succès ! Il faut délivrer des features, les feedbacks sont intégrés rapidement…

Ce qui donne lieu aux bizarreries JavaScript que l’on connaît aujourd’hui.

Javascript history



JavaScript 1.0 sort en 1995... Et Microsoft sort IE 3.0 en 1996.

Ce navigateur ne contient pas JavaScript, mais JScript... Une copie, avec ses différences.

Javascript history

C'était à prévoir, Microsoft a bombardé le marché d'IE grâce à ses PC.

Ils ont perdus quelques procès, mais ont quand même réussi à tuer Netscape.

Alors racheté par AOL et laissé pour mort.

Standard JS : ECMAScript

Définition

En **1996**, **Netscape** (avec Javascript), **Microsoft** (avec JScript, qui est une copie de javascript avec quelques modifications) et **Sun** (qui possédait le nom “Java”) ont créé le **TC39** (Technical Comittee 39). Son but est de **superviser le développement de la norme ECMAScript** afin de faire évoluer les **standards** de Javascript.

En 2015 sort **ES 6, une révolution** le standard est **adopté par tous les navigateurs !**

Depuis ES6 (2015), le TC39 sort une nouvelle version de Ecmascript tous les ans, en suivant le TC39 process

Standard JS : ECMAScript

TC39 process

Le **TC39 process** est composé de **5 étapes**.

Toutes les propositions d'evolution de la norme EcmaScript doivent passer par ces 5 étapes.

Le comité TC39 doit approuver ces évolutions étape après étape.

Standard JS : ECMAScript

TC39 stages

Stage 0 (Strawperson) : **idée d'évolution non formalisée** venant d'un membre du comité ou un contributeur reconnu.
Stage 1 (Proposal) : l'**idée est formalisée**, avec le problème soulevé, ses potentielles solutions et/ou ses challenges pour résoudre ce problème.
Stage 2 (Draft) : la proposition est **écrite dans une première version à l'aide d'Ecmascript**. Cette version doit décrire le plus possible, la syntaxe ou l'API qui sera développée. A noter qu'à partir de cette étape, si le comité valide la proposition, elle sera développée et éventuellement intégrée dans la spécification officielle.
Stage 3 (Candidate) : la proposition est **disponible pour être relue** et le comité est invité à émettre des feedbacks. Elle doit également passée un certain nombre de tests pour être validée.
Stage 4 (Finished) : la **proposition est prête à être intégrer** dans la prochaine version draft de la specification.

Stranger Things - JavaScript Edition

```
console.log(0 == [])
```

```
true
```

```
[] == ![];
true == [];
true == ![];
false == [];
false == ![];
!!"false" == !!"true";
!!"false" === !!"true";
true == "true";
```

```
null == 0;
null > 0;
null >= 0;
1 < 2 < 3;
3 > 2 > 1;
“10” + “2” = ?
“10” - “2” = ?
[] + [] = ?
{} + [] = ?
```

Awesome Javascript

Arrays

Les Arrays sont très utiles en Javascript, les données peuvent être hétérogènes et ils peuvent être redimensionner.
Il y a également beaucoup de méthodes utilitaires associées aux arrays, qui permettent de ne pas devoir parcourir les arrays avec un for ou un while

```
const dumbArray = [1, 2, 3, 4, 5];

// Map
const doubleDumbArray = dumbArray.map(element => element * 2);


// Reduce
const sumOfDumbArray = dumbArray.reduce((acc, current) => {
  return acc + current;
}, 0);

console.log("🚀 ~ doubleDumbArray:", doubleDumbArray)
console.log("🚀 ~ sumOfDumbArray ~ sumOfDumbArray:", sumOfDumbArray)
```

```
🚀 ~ doubleDumbArray:, [2, 4, 6, 8, 10]
🚀 ~ sumOfDumbArray ~ sumOfDumbArray:, 15
```