

How to manage Slidev assets

Press Space for next page →

はじめに

- ここ数年、仕事以外の趣味の発表資料作成は **Slidev** を使っている
- 使い続ける中で原稿の管理方法とか試行錯誤してるので、それらの棚卸し
 -  Slidev ?
 -  Slidev と  ウサギ と  カメ と
 -  Slidev と  GitHub と
 -  Slidev と  AI と

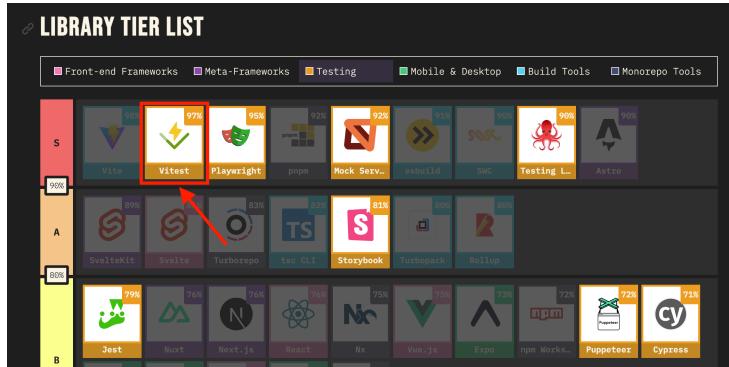


Slidev?



Slidevとは

- Markdownからスライド資料を作成できる
Dev向けプレゼンテーションフレームワーク
 - 2021年ぐらい?に日本でも話題に
 - Slidevを導入してMarkdownで美しいスライドを書こう - Qiita
 - Hackableなスライド作成ツールSlidevで遊ぶ
 - 作者はNuxtLabs社のAnthony Fu氏
 - 最近人気のVitestの作者でもある



ref: State of JavaScript 2023: Libraries



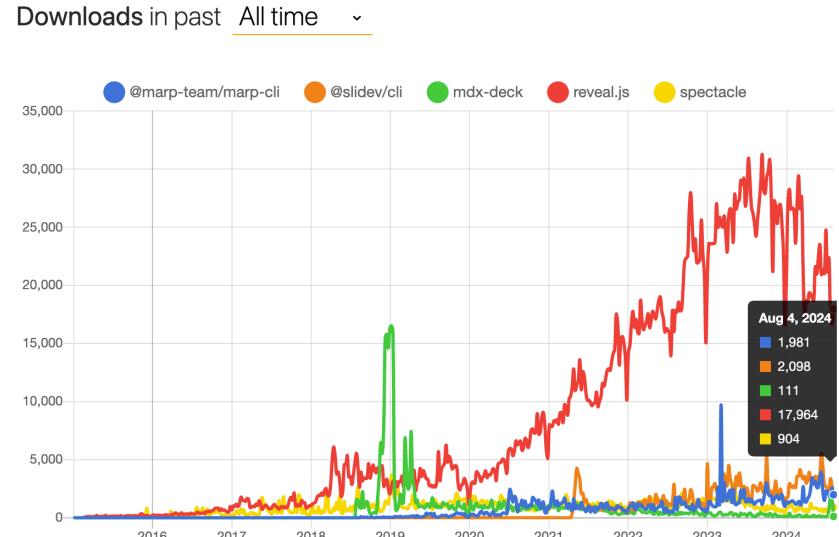
Slidev

Presentation Slides for Developers

NPM Trends

Marp | Sliderv | Reveal.js | MDX Deck | Spectacle

- DL数はReveal.jsが圧倒的
- MDX Deckは2019年ごろ盛り上がったが開発停止
- SlidervとMarpは(だいたい)同じような傾向
- Spectacleは古くから細く長く使われ続けている



- Start数もReveal.jsが多いが、Slidervも約半数
- Slidervが一番若いプロジェクト
- Marpは各サブプロジェクトに分散

灰字は marp-team/marp のもの

Stats

	Stars	Issues	Version	Updated	Created
@marp-team/marp-cli	1,822 →7,501	28	3.4.0	9 months ago →5 days ago	6 years ago
✓ @sliderv/cli	32,310	72	0.49.24	7 days ago	3 years ago
✓ mdx-deck	11,307	139	4.1.1	4 years ago	6 years ago
✓ reveal.js	67,415	818	5.1.0	4 months ago	11 years ago
✓ spectacle	9,727	75	10.1.8	5 months ago	9 years ago

Slidev: Getting Started

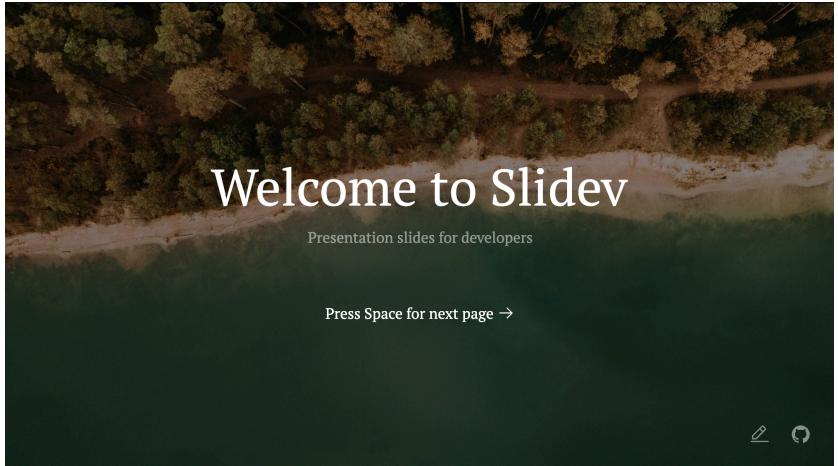
```
$ npm init slidev@latest
```

Demoスライドプロジェクトが生成される。

```
$ npm run dev  
--> 開発用にスライドアプリを起動(ホットリロード付)
```

```
$ npm run export  
--> スライドをPDFファイルとしてエクスポート (要 playwright-chromium)
```

```
$ npm run build  
--> スライドをSPAとしてビルド
```



Markdown(+α)で書ける

いろいろと

- こんな感じに
- 書けて

HTMLも書けて

UnoCSSでスタイルも指定できる

💡 Syntax Highlightも高機能

```
console.log(`Writable/runnable highlight by Monaco`) ▶
```

Writable/runnable highlight by Monaco

```
const str = JSON.stringify(json)
console.log("Shiki Magic Move: ", str)
```

layout: two-cols

Markdown(+α)で書ける

いろいろと

* こんな感じに

* 書けて

<h3>HTMLも書けて</h3>

<div class="c-orange animate-bounce">UnoCSSでスタイルも指定で

<mdi-lightbulb class="c-yellow"/> Syntax Highlightも高機能

```
```js {monaco-run} {autorun:false}
console.log(`Writable/runnable highlight by Monaco`)
```
```

<!-- since v0.48.0 -->

```
```md magic-move
```

```
```js
```

```
const str = JSON.stringify(json)
console.log("Shiki Magic Move: ", str)
```
```

```
```js
```

```
console.log(`Shiki Magic Move: ${JSON.stringify(json)}`)
```
```

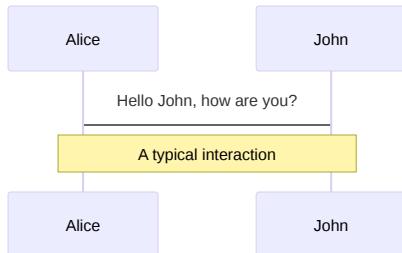
# Markdown(+α)で書ける

いろいろと

LaTeX

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

Diagram (mermaid)



Draggable Object (since v0.49.0)



```

layout: two-cols
dragPos:
 logo: 84,447,70,69,0
: Left,Top,Width,Height,Rotate

Markdown(+α)で書ける
いろいろと

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

Diagram (Mermaid)
``mermaid {scale:0.5}
sequenceDiagram
 participant Alice
 participant John
 Alice->>John: Hello John, how are you?
 activate John
 John-->>Alice: A typical interaction
 deactivate John
```

#### Draggable Object (since v0.49.0)

```



&



Rabbit

<https://rabbit-shocker.org/ja/>

- Rubyist向けのプレゼンテーションツール
- RD(Ruby Document), Wiki記法, Markdownでスライド作成できる
- Rubyのパパ・Matzがよく使っている

The screenshot shows the Rabbit website's homepage. At the top, there is a navigation bar with links for "English", "いいね!", "シェアする", and "gem install rabbit". Below the navigation bar is a banner featuring a red gradient background with a white rabbit icon and the word "Rabbit". The main content area has a pink header with the text "はじめに" (Getting Started). Below this, there is a section titled "Rabbitとは" (What is Rabbit?) containing text about the tool's purpose and how it works. To the right, there is a sidebar titled "最新リリース" (Latest Releases) listing "Rabbit 3.0.3" (2023-07-02) and "Rabbiter 2.0.3" (2016-08-21). The bottom of the page features a decorative footer with a repeating pattern of rabbits and turtles.

The screenshot shows a presentation slide from RubyKaigi 2022. The slide has a black header with the text "Hiring Developers". The main content area is mostly blank, with a few small decorative icons at the bottom. In the bottom right corner, there is a logo for "RubyKaigi 2022 Sep 8th - 10th" and a "Powered by Rabbit 3.0.0" badge. The footer of the slide includes the text "RubyKaigi 2022 #RubyKaigi" and "Sponsored by NEXWAY".

Matz Keynote / Yukihiro "Matz" Matsumoto @yukihiro_matz

特徴

- スライドの下部に ウサギ と カメ が表示される
 - ウサギ: ページが進むごとに右に移動する
 - カメ: 時間経過とともに右に進む
- ウサギ が カメ より遅れている
→ プレゼンが時間内に終わらない



🐱 kaakaa/slides-addon-rabbit | 🏛️ slides-addon-rabbit

RabbitをSlidev Addonとして実装

How to use

1. Install the addon

```
$ npm i slides-addon-rabbit
```

2. Declare the addon in frontmatter

```
---
addon:
  - slides-addon-rabbit
rabbit:
  slideNum: true
---
```

3. Access a slide with `?time=${MINUTE}` query

e.g.: <https://example.com/slides?time=25>

🐢 components/turtle.vue

```
<template>
  <div class="rabble-container" :style="{left: left + 'px'}>
    <emojione-monotone-turtle class="icon" />
  </div>
</template>
<script>
  ...
  data() {
    const maxWidth = this.$slidev.configs.canvasWidth - 20
    const time = this.$route?.query?.time || 10;
    return {
      left: 0,
      speed: maxWidth / (time * 60 * 10) // 100ms毎の変化量
    }
  },
  beforeUpdate() {
    ...
    this.intervalId = setInterval(() => {
      if (this.left < this.maxWidth) {
        this.left += this.speed;
      }
    }, 100); // update per 100ms
  }
</script>
```

TODO

Page Weight対応

セクションタイトルスライドも内容盛りだくさんのスライドも、同じ1ページとして換算されるためページごとにWeightを設定できた方が良い? (Weightの大きなスライドを消化した時に一気にウサギが移動することになるので、見た目的にスマートではない気がする)

長い時間を指定すると亀の動きが実際の時間とズレている気がする

30分などの長い時間を指定すると、30分経っても亀がゴールしていない気がする(未検証)

setIntervalで素朴に100msごとに動かしているだけなので、ブラウザのクロックがズレていっているのかもしれない

<https://github.com/kaakaa/slides-addon-rabbit/blob/master/components/Turtle.vue#L36>



やりたい事

- Slidevの原稿はMarkdown(テキスト)ベースなので、GitHubで管理したい
 - 資料ごとにリポジトリ作るのは面倒なので、1リポジトリで管理したい
- GitHub Actions使ってビルドできるようにしたい

現在のリポジトリ構成

ディレクトリごとに資産 (*.md, *.pngなど) を格納

Directory Structure

- `slides.md`: メインのスライド (※ファイル名固定)
- `pages/`: 分割したスライドページ
- `public/`: 画像などのアセットファイル

```
.  
|   └── 20230202_supply-chain/  
|   └── 20230406_sidejob/  
|       └── 20230601_slidev/  
|           ├── slides.md  
|           └── pages/  
└── public/
```

Build

```
# devコマンドはエントリとなるMarkdownファイルを指定  
$ npm run dev 20230601_slidev/slides.md  
  
# ビルド処理はディレクトリ名を指定して実行 (`slides.md`を探す)  
$ npm run build --slide=20230601_slidev  
$ npm run export --slide=20230601_slidev
```

`public/`

```
|   └── README.md  
|   └── package-lock.json  
└── package.json
```

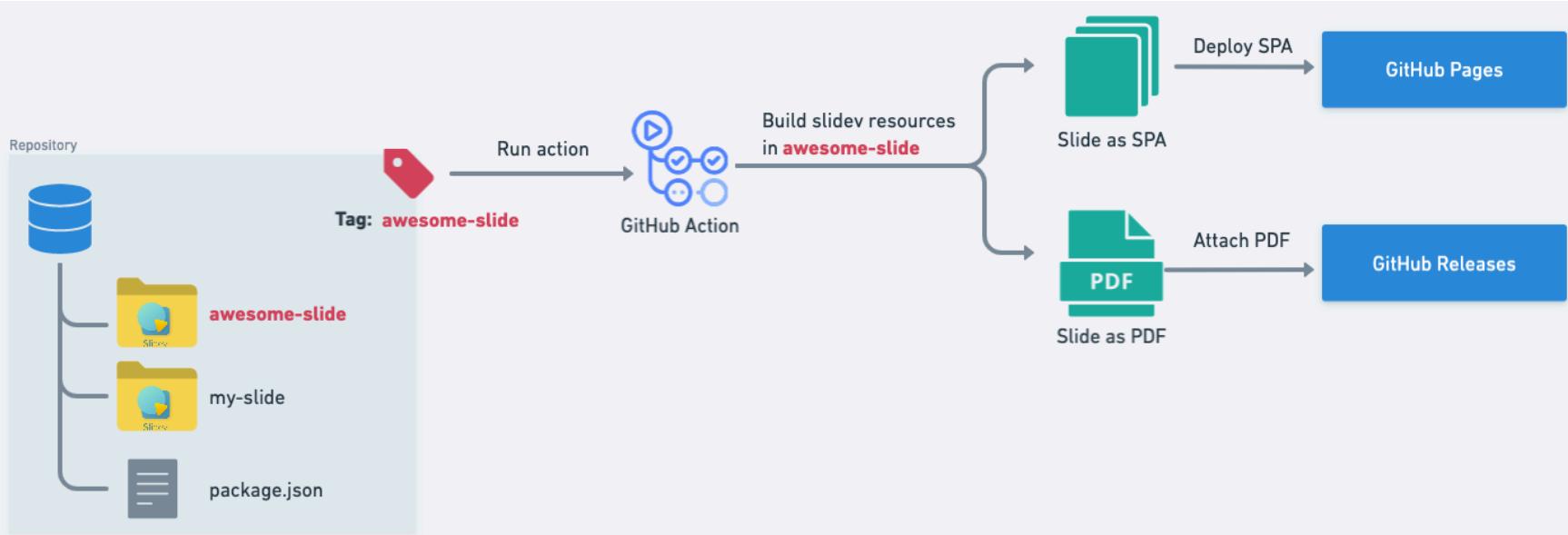
各スクリプトで辻褄あわせてゴリゴリやってるだけ...



GitHub Actions

Build & Deploy

Tagを打つことでTagと同名のディレクトリ内のSliderv資料をビルドしGitHub Pages / Releasesにデプロイする



OGP対応

yemoto
@kaakaa_hoe_prog · [Follow](#)

GitHub Pagesに公開したSPA形式のslidev
プレゼンテーションにOGPを付与できる
ようになった(かもしれない)

 kaakaa.github.io
Advanced Slidev

2:05 PM · May 28, 2023

[Read 1 reply](#)

GitHub Action内で以下の2ファイルを生成

 `index.html`

```
<head>
  <!-- Slidev原稿のfront matterの情報を利用 (title, info) -->
  <meta property="og:title" content="${title}" />
  <meta property="og:type" content="website" />
  <meta property="og:url" content="${url}" />
  <meta property="og:image" content="${imageUrl}" />
  <meta property="og:description" content="${info}" />
</head>
```

 `preview.png`

OGPのプレビュー画像は、生成されたPDFファイルの
先頭ページをPNG形式で切り取り



&

(AI)

A pink icon depicting a brain with a magnifying glass positioned over it, symbolizing artificial intelligence.

VS Code Extension: AIが生成するSlidev原稿



- AIを使ってSlidevの原稿を生成するVS Code Extension を公開！

- スライドアイで検索

ChatGPT等とは異なり、もっとインタラクティブにAIと原稿を作りたい！という想いから誕生しました。

- リアルタイム生成

SlidevのVS Code Extensionと組み合わせることで、AIが作った原稿を即座にプレビュー可能です。

デモ

[デモ動画やGIFを入れる] AIによる原稿作成のデモンストレーションを見せる

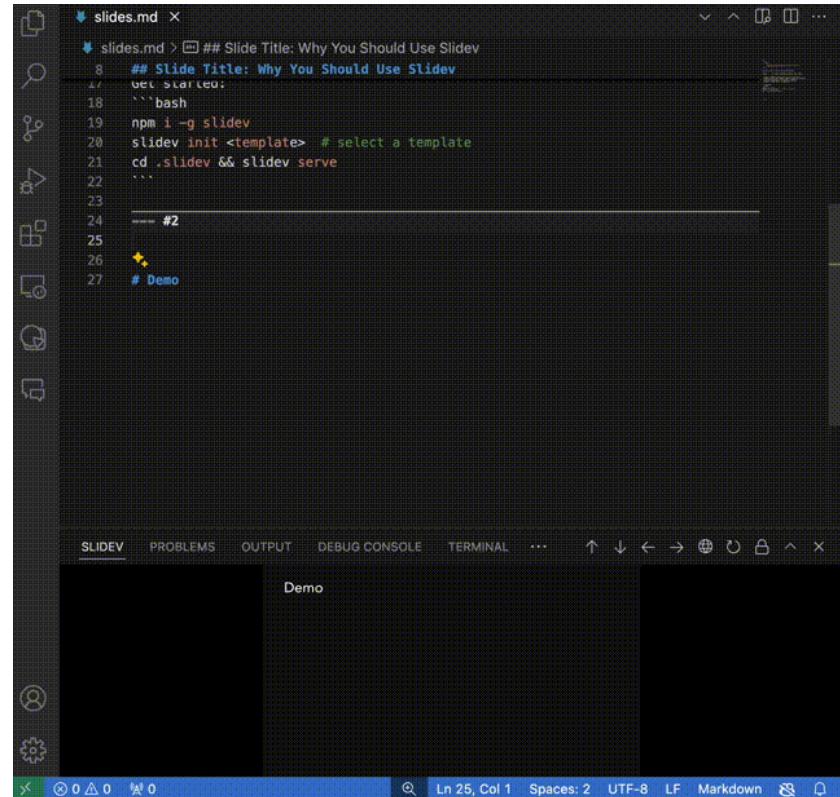
実際に使ってみましょう！

このスライドは、その場でAIに書き起こさせることで説明できます！

Slidaiv

 kaakaa/slidaiv |  Slidaiv

- VS Code上でSlidevの原稿を生成できるVS Code拡張
- プロンプトはFrontmatterで指定
(@slidev/parserで原稿を解析してモデル化して取得)
- SlidevのVS Code拡張を併用すれば、生成した原稿の表示内容をすぐに確認できる
- YAMLに複数ページ分のPromptを書いて一気に原稿を出力するCLIツール slidaiv - npm
- 調査資料を元に生成できるようにしたい。RAGりたい。
- 原稿の改善もAIでやれるようにしたい
- 画像生成AIの組み込みやUnoCSSを使ったデザイン等もしたい
- あんま良い感じの原稿が生成されないのでなんとかしたい...



The screenshot shows the VS Code interface with the Slidaiv extension installed. The left sidebar has icons for file, search, and other extensions. The main editor area displays a Markdown file named 'slides.md' with the following content:

```
slides.md x
  * slides.md > ## Slide Title: Why You Should Use Slidev
  8  ## Slide Title: Why You Should Use Slidev
  9  Get Started:
 10  ````bash
 11  npm i -g slidev
 12  slidev init <template> # select a template
 13  cd .slidev && slidev serve
 14  ...
 15  ...
 16  ...
 17  ...
 18  ...
 19  ...
 20  ...
 21  ...
 22  ...
 23  ...
 24  ...
 25  ...
 26  ...
 27  # Demo
```

The bottom status bar shows the file is at Line 25, Column 1, with 2 spaces, using UTF-8 encoding, and is a Markdown file. The bottom right corner shows the Slidaiv extension's status bar with icons for file operations.

Slidaiv

 kaakaa/slidaiv |  Slidaiv

2ページ前のスライドは以下のプロンプト (+ System Prompt) qを元に **Ollama x Command-R Plus** で生成

```
---
```

```
slidaiv:
  prompt:
    - AIを使ってSlidevの原稿を生成するVS Code Extensionを公開したことを紹介したい
    - >-
      公開したVS Code Extension
      https://marketplace.visualstudio.com/items?itemName=kaakaa.slidaiv
    - 元々、ChatAIでスライドの原稿を生成する方法は提案されていたが、もっとインタラクティブにやりたいという野望があった
    - >-
      Slidev自体のVS Code
      Extension(https://marketplace.visualstudio.com/items?itemName=antfu.slidev)と組み合わせることで、リアルタイムにAIが生成した/
      Code Extensionとして作ろうと思った
    - unocssに関する記述は含めないでください
  ---
```

(Command-R Plus (104B) on Ollama)

VS Code Extension: AIが生成するSlidev原稿



- AIを使ってSlidevの原稿を生成するVS Code Extension を公開！

- スライドアイで検索 

ChatGPT等とは異なり、もっとインタラクティブにAIと原稿を作りたい！という想いから誕生しました。

- リアルタイム生成 

SlidevのVS Code Extensionと組み合わせることで、AIが作った原稿を即座にプレビュー可能です。

デモ

[デモ動画やGIFを入れる] AIによる原稿作成のデモンストレーションを見せる

実際に使ってみましょう！

[このスライドは、その場でAIに書き起こさせることで説明できます！]

(llama3.1:70b on Ollama)

Slidev形式のVS Code Extensionの公開について

AIによるスライド原稿生成ツールの紹介

- Slidaiv
- ChatAIを利用したスライド原稿生成のアイデアをインタラクティブに進化

背景

- SlidevのVS Code Extensionとは?<https://marketplace.visualstudio.com/items?itemName=antfu.slidev>
- リアルタイムな原稿生成確認が可能になり、生産性向上を目指す
このツールにより効率的なスライド制作が実現します。

(OpenAI gpt-4o)

新しいVS Code Extensionの紹介

- **Slidevの原稿生成にAIを活用**
- 公開されたVS Code Extension: [Slidaiv](#)
- **背景**
 - ChatAIでのスライド原稿生成は既に提案されていた
 - もっとインタラクティブな方法を追求
- **目的**
 - SlidevのVS Code Extension([リンク](#))と組み合わせ
 - リアルタイムにAI生成原稿を確認

Develop / Publish VS Code Extension

- 公式ドキュメントとサンプル集で大体のことはできる (未だにYeomanを使ってるのが謎)
 - [Your First Extension | Visual Studio Code Extension API](#)
 - [microsoft/vscode-extension-samples: Sample code illustrating the VS Code extension API.](#)
- Marketplaceへのリリースも公式ドキュメント通りにやれば良い ([Publishing Extensions](#))
 - Azure DevOpsのPATを元にVisual Studio MarketplaceのPublisherを作成する必要がある
 - 公式CLIツール `vsce` を使うことでGitHub Actionsなどからリリースを実行することもできる
 - 管理画面からExtensionの閲覧数やインストール数が見える



TODO

Model/Promptの改善

GitHub Models (AIアプリ開発用途で商用AIサービスをお試しできるサービス。実行回数制限あり。) 使って、適切なModel/Prompt等を調査していきたい。

RAG等の導入

調査資料をどこかのディレクトリに放り込んでおくと、それを読み込んで良い感じの生成ができるようになるとか。

VS Code Extensionからも呼べるローカルで動くRAGとは...。どこかのサービスを呼び出す感じにしなきやだめかなあ...。

The screenshot shows the GitHub Models interface. At the top, there are filters: 'By: All providers', 'Capability: Chat/completion', and 'Tag: All'. Below the filters, there are five model cards:

- Llama-3.2-11B-Vision-Instruct**: Described as having image reasoning capabilities on high-res images for visual understanding apps.
- Llama-3.2-90B-Vision-Instruct**: Described as having advanced image reasoning capabilities for visual understanding agentic apps.
- AI21 Jamba 1.5 Mini**: Described as a 52B parameters (12B active) multilingual model, offering a 256K long context window, function calling, structured...
- AI21-Jamba-Instruct**: Described as a production-grade Mamba-based LLM model to achieve best-in-class performance, quality, and cost efficiency.
- Cohere Command R+**: Described as a state-of-the-art RAG-optimized model designed to tackle enterprise-grade workloads.
- Cohere Command R+ 08-2024**: Described as a state-of-the-art RAG-optimized model designed to tackle enterprise-grade workloads.

まとめ

Warp Up

- Slidev。Markdownでスライド資料作れて、2段組のページも簡単に作れるのいいぞ
- ウサカメで発表時間管理できるようになったのもっといいぞ
- ディレクトリ構成を工夫することで、GitHub上でソース、SPA、PDFを管理できるのもいいぞ
- AIを使った原稿生成はまだ先が長いぞ

感想

- 画像などのアセットもリポジトリに入れると1スライド10MBぐらいのサイズを取るので、アセット管理を外に逃がせるともうちょっとスマートにはなると思う(あまりやる気はないけど)
- Markdownで書けると言ってもレイアウト確認しながら作っていくので、楽という気はありません
 - WYSIWYGなパワポとかの方が楽なんだろうなと感じることが多い
- とはいえGitHub Actionsを使ってPages/Releasesへのアップロードを自動化できたのは達成感ある
- 100BぐらいのLocal LLMだとgpt-4oにまだまだ及ばない感じ