Marpでスライドを作成しよう ～ カスタムテーマとGitHub Actionsによる自動化まで ～

はじめに

この記事では、Marpの使い方や、実際にGitHub Pagesで公開する運用の流れまでを説明します。

アイスタイルのアプリ開発グループでは、実際に中途採用向けのスライドをMarpを使って作成・公開しています。
手軽に更新できるなど、メリットが多くありますので、ぜひ試していただければと思います。
https://istyle-inc.github.io/recruitment-docs/app-group/introduction

Marpとは

https://marp.app/

> Marp (also known as the Markdown Presentation Ecosystem) provides an intuitive experience for creating beautiful slide decks. You only have to focus on writing your story in a Markdown document.

シンプルなMarkdown形式のテキストファイルからスライドを作成できます。

- このスライドのソース： https://github.com/75py/slide/blob/main/src/md/marp.md
- GitHub Pagesで公開したもの： https://75py.github.io/slide/marp.html
- PDFをアップロードしたもの： Speaker Deck

スライドをMarpで作成するメリット

資料の内容に注力しやすい

パワーポイント等では、レイアウトの調整にどうしても時間を取られがちです。
一方、MarpはMarkdownファイルから自動でスライドが出力できるため、資料の中身に注力できます。

レビューしやすい

MarkdownファイルをGit管理することで、差分確認が容易になります。
パワーポイントでも差分は確認できますが、やはりテキストファイルのdiffほど分かりやすくはありません。
GitHub Pagesと組み合わせれば、PRを作成→社内レビュー→マージ後即公開が可能です。

marp-cli

https://github.com/marp-team/marp-cli

インストール方法はREADMEの通りです。
このプロジェクトではローカルインストールとしています。

```
npm install --save-dev @marp-team/marp-cli
```

```
{
  "name": "slide",
  "version": "1.0.0",
  "scripts": {
  },
  "devDependencies": {
+     "@marp-team/marp-cli": "^3.0.0"
  }
}
```

marp-cliの使い方

`marp` でヘルプが表示できます。今回使ったのは以下の通りです。

プレビュー表示

```
marp src/md/marp.md --theme src/theme/slide.css --preview
```

PDF出力

```
marp src/md/marp.md --theme src/theme/slide.css --pdf
```

npm runでテーマ指定を省略

```
{
  "name": "slide",
  "version": "1.0.0",
  "scripts": {
+     "preview": "marp $npm_config_src --theme src/theme/slide.css --preview",
+     "pdf": "marp $npm_config_src --theme src/theme/slide.css --pdf"
  },
  "devDependencies": {
    "@marp-team/marp-cli": "^3.0.0"
  }
}
```

使い方

- `npm run preview --src src/md/marp.md`
- `npm run pdf --src src/md/marp.md`

marp-cli オプション設定

> Marp CLI can be configured options with file, such as marp.config.js, marp.config.cjs, .marprc (JSON / YAML), and marp section of package.json. It is useful to configure settings for the whole of project.

例えば、テーマを固定したいだけなら `.marprc.yml` に以下を書くだけで実現可能です。

```
theme: src/theme/slide.css
```

地味なはまりポイント：順序なし箇条書き

Markdownではアスタリスク、ハイフンで順序なし箇条書きを表現できますが、どちらを使うかでMarpの出力が変わります。

- ハイフン：一度に描画される
- アスタリスク：アニメーション描画される（次へ進むと表示される）

なお、当然ながらPDF出力では差分はありません。

カスタムテーマ

built-in themes

default、gaia、uncoverの3種類があります。
どれもかなり見やすいテーマなので、色指定だけで済むのであればカスタマイズなしで利用するのも選択肢に入るのではないでしょうか。

https://github.com/marp-team/marp-core/tree/main/themes

defaultを拡張したテーマを作成する

拡張する場合はこれだけで済みます。

```
@import 'default';
```

私が愛用しているWebStorm（JetBrainsのIDE）だと、defaultが解決できずに赤線が引かれますが、無視して構いません。
どうしても気になるようなら、警告を消すことも可能です。

```
+ /*noinspection CssUnknownTarget*/
@import 'default';
```

カスタマイズ

公式ドキュメント

https://marpit.marp.app/theme-css

例えば、背景色を変えたいならsectionに指定すればOKです。

```
section {
    background-color: lightblue;
}
```

企業なら会社ロゴを入れたりするといい感じになります。

特定のページのみカスタマイズ

Markdownに以下の記述を追加します。

```
<!-- _class: title -->
```

すると、出力されるHTMLは `<section class="title">` に変わるので、section.titleにCSS定義を追加すればOKです。

```
section {
    background-color: red;
}
```

GitHub Actionsによる自動化

今回想定する使い方

- mainブランチにマージされたら、MarkdownファイルからPDFファイルを作成する
- 作成されたPDFファイルをGoogleドライブに置いて共有する（artifactsで保存し、手動でアップロードする）
- GitHub Pagesで公開する

`marp-cli-action`を利用するのが一番簡単そうでした。
https://github.com/KoharaKazuya/marp-cli-action/blob/main/README.ja.md

ワークフロー（全文）

```yaml
name: Convert Markdown into PDF
on:
  push:
    branches:
      - main
  workflow_dispatch:

jobs:
  publish:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Convert Markdown into PDF
        uses: KoharaKazuya/marp-cli-action@v2
        with:
          config-file: ./.marprc-ci.yml
          generate-html: true
          generate-pdf: true

      - name: Save outputs
        uses: actions/upload-artifact@v2
        with:
          name: output
          path: ./output

      - name: Deploy to GitHub Pages
        uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./output
```

トリガー

```yaml
name: Convert Markdown into PDF
on:
  push:
    branches:
      - main
  workflow_dispatch:
```

- mainブランチがプッシュされたとき
- 手動実行

marp-cli-action

```yaml
jobs:
  publish:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Convert Markdown into PDF
        uses: KoharaKazuya/marp-cli-action@v2
        with:
          config-file: ./.marprc-ci.yml
          generate-html: true
          generate-pdf: true
```

HTML, PDFファイルを作成します。

設定ファイルは `.marprc-ci.yml` を使用します。

成果物を保存

```
      - name: Save outputs
        uses: actions/upload-artifact@v2
        with:
          name: output
          path: ./output
```

output.zipをダウンロードできるようになります。
出力されたファイルをGoogleドライブ等で共有したい場合はこちらを使ってください。

GitHub Pagesで公開する

```
      - name: Deploy to GitHub Pages
        uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./output
```

outputディレクトリのファイルを公開します。

GitHubの `Settings > Actions > General > Workflow permissions` を `Read and write permissions` にする必要があります。

成功すると、outputディレクトリ配下のファイルが `gh-pages` ブランチにプッシュされます。

https://github.com/75py/slide/tree/gh-pages

良い感じに設定すると、以下のようにアクセスできます。

https://75py.github.io/slide/marp.html

まとめ

Marpを使うことで、勉強会の登壇資料や、採用資料などを効率よく作成できます。
資料作成に疲弊している方はぜひ試してみてください。

https://open.talentio.com/r/1/c/isytyle_career/pages/43022
https://open.talentio.com/r/1/c/isytyle_career/pages/43019