marp-cli を Next.js から利用する

背景

- Next.js で生成しているサイトで Marp を利用したい
  - プレゼンテーション用のスライド表示
  - 画像や PDF も作成
- marp-cli はモジュールとしても利用可能
  - ただしモジュールでの利用は「ソースと出力がファイル経由」
  - テーマも基本的には `.css` ファイルに書き出しておく必要がある

ファイル経由の問題点

- CMS ソースの場合は一時ファイルの作成が必要(or FIFO？)
- 出力ファイルの競合管理
  - プレビューモード等の動的生成時
  - 複数ページからの同一ファイル参照等

できればファイルを経由しないで利用したい。

ソースをストリームとして渡す

marp-cli はコマンドとして実行した場合はパイプで入力を渡せる。今回はそれを利用する。

```
const marpP = spawn(marpPath, formatOpts);
if (marpP && marpP.stdout) {
  // marpP.stdout.setEncoding('base64');
  marpP.stdout.setEncoding(options.encoding || 'binary');
  marpP.stdout.pipe(w);
}
if (marpP && marpP.stdin) {
  marpP.stdin.write(markdown);
  marpP.stdin.end();
}
```

出力はファイルへ書き出し

当初はできる限りストリームで扱い「プレゼンテーション(.html)」等は `page/*` 以下に配置していたが、変換処理が重いのでキャッシュを作成しながらファイルとして書き出すことにした。

ただし、厳密な排他制御をしていないので今回は静的生成のみ対応としている。

- `pages/deck[id].tsx` 用のスライド作成時に hash を生成
- `.mardock/cache/` 以下に hash をキーとしてキャッシュを作成
- hash を元に最新版のファイルを `public/` 以下にコピーする

テーマの利用(`marp.config.js`)

(テーマ専用の機能ではないが) `marp.config.js` の利用で「marp-cli から利用されるインスタンス」の生成に介入できる。ここでテーマをコード(文字列)として渡すことが可能。

```
const { Marp } = require('@marp-team/marp-core');
const themes = require('./src/marp-theme');

module.exports = {
  engine: (opts) => {
    const marp = new Marp(opts);
    themes.forEach((theme) => marp.themeSet.add(theme));
    return marp;
  }
};
```

外部スクリプトの利用

`public/` 等に静的なファイルを書き出す場合は `next build` とは別に外部スクリプトを用意するのが定石のもよう。

今回はそれなりに進めてしまった後なので対応していないが、縛りがないのならば外部スクリプトは検討したほうが良い。

参考: Create a Next.js RSS feed for your static website - DEV Community

その他

- 複数画像へ変換は marp-cli 側からファイルへ出力されることになる(たぶん)
- テーマは毎回コンパイルされてしまうのでファイル書き出しも検討
- Auto Scaling 対応は marp-core でも必要(コードブロックなどが表示されない)
- 動的生成することはあるので厳密な競合管理はいずれ必要

付録: Auto-Scaling(helper script) 対応

現在の構成では使っていないが、ログとしてメモしておく。

- 前述のスクリプトとは別に helper script が必要(テーマによる)
- marp-core で export されている関数があるのでそちらを利用
- 詳細は「The core of Marp converter # script: boolean | object」

```
import { browser as marpCoreBrowserScript } from '@marp-team/marp-core/browser';

const Layout = ({ notification, head, body }: Props) => {
  // snip...
  useEffect(() => {
    const cleanup = marpCoreBrowserScript();
    return () => cleanup();
  }, []);
```