AH Formatter を用いた Markdown-PDF 変換事例紹介

2019 年 5 月

フェリックス・スタイル

はじめに

この文書は、次のような方に向けた事例紹介です。

- Visual Studio Code で、ソフトウェア開発に付随する報告書などの作成も行いたい
- 文書は Markdown で執筆し、PDF で提出したい
- コマンドライン操作のみで PDF ビルドが完了するようにしたい
- GitHub における Markdown 表示のようなスタイリングにしたい
- PDF には表紙や目次、ノンブル（ページ番号）を付けたい
- 目次には、項目タイトルとページ番号、ドキュメント内リンクが自動的に生成されてほしい
- ノンブルは、表紙や目次を除いた本文ページのみに振りたい
- PDF には余計なプロパティを含めたくない

上記の要件を全て満たす PDF を得るために筆者が用いたフローを紹介します。（上記の要件の一部が不要であれば、また違ったフローがあり得るでしょう。）

ここで取り上げる内容は、筆者の環境で手間をかけずに実現するために用いたフローであって、ベストプラクティスではありません。汎用性に欠ける点もありますが、より良い実現方法が共有されていく叩き台になれば幸いです。

本稿の公開にあたり、鹿野桂一郎さんにレビューのご協力をいただきました。ありがとうございました。

目次

Markdown から CSS 組版により PDF 文書を生成する

本稿では、Markdown 形式の原稿から、表紙や目次、ノンブルなどを備えた PDF の文書へと変換するフローを紹介します。

Markdown を PDF に変換するフローはいくつか考えられますが、本稿で紹介するフローでは、AH CSS Formatter を利用した CSS 組版により以下の点を実現しています。

- コマンドライン操作のみで完結するので繰り返しビルドが容易
- TeX の知識がなくても PDF の見た目を調整できる

変換の方針

AH CSS Formatter による CSS 組版のために、まずは Markdown の原稿を HTML へと変換します。具体的には下記のような前処理を行います。

- 複数の Markdown ファイル（前書き、本文、付録など）を結合する
- 結合した Markdown の見出し要素から目次を生成する
- 目次を含む Markdown ファイルから HTML ファイルへと変換する
- HTML ファイルにメタデータなどを付与する
- 画像などを HTML ファイル中にインライン化して埋め込む

Markdown や HTML に対するこれらの操作では、JavaScript 製の便利なツールが Node.js パッケージとして利用できます。そこで今回は、AH CSS Formatter による CSS 組版を含めた処理全体を、Node.js のパッケージ管理ツールである npm により実行できるようにしました。

以降では、npm のタスク実行機能を用いて上記の前処理および CSS 組版を逐次実行し Markdown 原稿から PDF へとビルドする手順を、この文書そのものを例に紹介します。その後、ビルドプロセスの各ステップについて、簡単に内容を説明していきます。

実際に Markdown から PDF のビルドを試す

この文書の作成に用いたソースファイル一式「md2pdf」[1)]をダウンロードして、Markdown 原稿から PDF へのビルドを試すことができます。

動作環境

ここで紹介するフローは、以下の環境で動作します。

- Node.js 10+
- AH CSS Formatter V6.6 MR5+

また、必須ではありませんが、執筆作業のための環境として Visual Studio Code（VS Code）があると便利でしょう。以降では、VS Code で Markdown 原稿の編集やプレビューを設定する方法や、PDF へのビルドなどを一つのウィンドウでできるようにする方法も説明します。

セットアップ

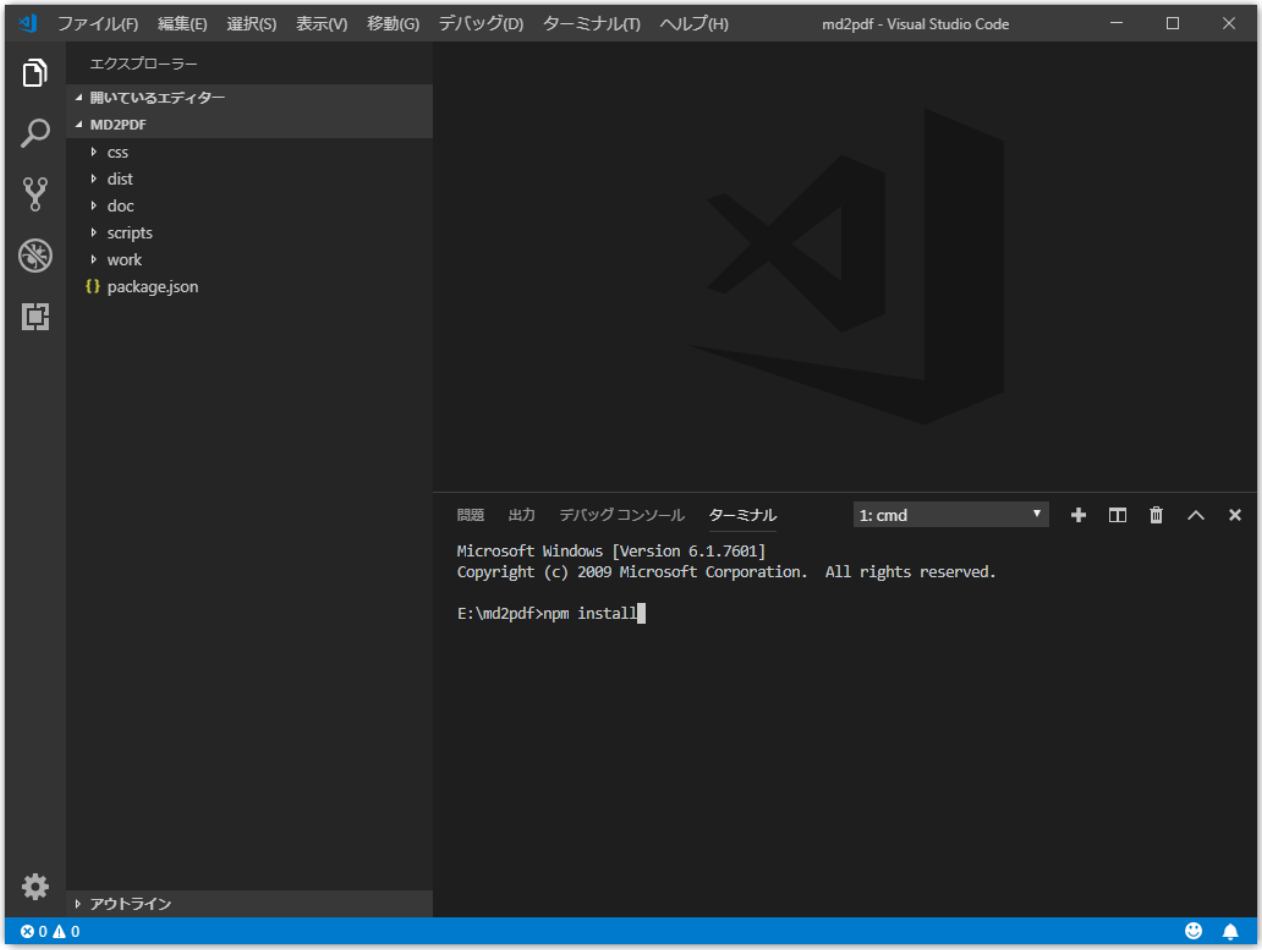
ソースファイル一式をダウンロードしたら、適当な場所に解凍してください。

次に、VS Code を用いる場合には、VS Code の「ファイル」メニューから「フォルダーを開く」を選択し、解凍したフォルダを開いてください。続いて、「ターミナル」メニューから「新しいターミナル」を選択します。VS Code を用いない場合は、ターミナル（Windows であればコマンドプロンプトなど）を開き、解凍したフォルダに移動してください。

準備ができたら、ターミナルから以下のコマンドを実行してください。Markdown から PDF へビルドするプロセスに必要な Node.js パッケージがインストールされます。

```
npm install
```

1) https://github.com/2SC1815J/md2pdf



VS Code のターミナル（右下）

Markdown Preview Github Styling の導入

VS Code で Markdown 原稿を執筆する場合、VS Code の拡張機能「Markdown Preview Github Styling」[2]をインストールすると、Markdown が GitHub 風のスタイリングでプレビュー表示されるようになり、変換結果をイメージしやすくなります。

2) https://marketplace.visualstudio.com/itemdetails?itemName=bierner.markdown-preview-github-styles

「Markdown Preview Github Styling」によるプレビュー表示

複雑な表の記述

Markdown では表現力が十分でない部分（セルが結合された表など）は、表 1 のように、Markdown 中に HTML で直接記述しています（もちろん、生成された PDF 上では、HTML ではなく表として描画されているはずです）。

表 1

見出し 1	見出し 2	
	見出し 2-1	見出し 2-2

なお、AH Formatter V6.6 MR4 では、結合されたセルの背景色が正しく設定されないことがありました。この問題は AH Formatter V6.6 MR5 で修正されています。[3)]

PDF のビルド

ターミナルから次のコマンドを実行することで、doc フォルダ内の Markdown 原稿から HTML と PDF が生成され、dist フォルダに出力されます。[4)]

```
npm run build
```

このビルドプロセスは、package.json ファイルの scripts プロパティに記載された、`build:doc-1` から `build:doc-6` のステップに従って実行されます。以降では、この各ステップに沿って、Markdown 原稿を PDF へと変換するまでの処理内容を見ていきます。

Markdown 原稿から目次を生成

`build:doc-1` Markdown ファイルの連結

目次のひな形（doc/toc.md）を、複数の Markdown 原稿（doc/preface.md, doc/main.md, doc/appendix.md）と合わせて、単一の Markdown ファイルにします。

```
npx minicat doc/preface.md doc/toc.md doc/main.md doc/appendix.md > work/all.md
```

このステップによって、仮 Markdown ファイル（work/all.md）が生成されます。

`build:doc-2` DocToc による目次生成

Node.js パッケージ「DocToc」[5)]を用いて、目次を生成します。Markdown に記述された見出し要素が抽出され、目次が自動的に作成されます。

```
npx doctoc --notitle --maxlevel 3 work/all.md
```

3) AH Formatter V6.6 MR4 までの場合、明示的にクラス指定を行うことで対処します。
4) VS Code を利用している場合は、VS Code の `npm.enableScriptExplorer` 設定を有効にすることで、コマンドを入力しなくてもマウスクリックでビルドを実行できるようになります。
5) https://github.com/thlorenz/doctoc

このステップによって、仮 Markdown ファイル（work/all.md）の目次が更新されます。

Markdown から HTML へ変換

`build:doc-3` markdown-it による変換

Node.js パッケージ「markdown-it」[6]を用いて、Markdown から HTML へ変換します。

```
node scripts/mdit.js work/all.md work/all_md.html
```

Markdown の見出し要素を日本語で記述しており、目次部分の href に URI エンコードされた日本語文字列が設定されている場合、AH Formatter V6.6 MR4 では、CSS 組版によるページ番号表示（target-counter(attr(href))の処理）が正しく動作しない問題がありましたが、AH Formatter V6.6 MR5 で修正されました。[7]

このステップによって、部分的な HTML ファイル（work/all_md.html）が生成されます。

`build:doc-4` Markdown 原稿に含まれない内容を付与

markdown-it が生成する HTML には、<html>や<head>、<body>、CSS ファイルへのリンクなどは含まれていません。これらを記述したテンプレート HTML ファイル（doc/template.html）を別途用意し、markdown-it によって生成された HTML がその中に含まれるようにします。今回は表紙の内容もここに記述しました。

```
node scripts/ejs.js doc/template.html work/all.html
```

AH Formatter V6.6 では、HTML ファイルの meta タグに次の記載を含めることで、この HTML から変換された PDF をビューワで開いたときにページ全体がウィンドウに収まっている状態にズームさせることができます。

```
<meta name="openaction" content="#view=fit">
```

6) https://github.com/markdown-it/markdown-it
7) AH Formatter V6.6 MR4 までの場合、Markdown の見出し要素からアンカーを作成する Node.js パッケージ「anchor-markdown-header」に変更を加えて対処する方法があります。

また、AH Formatter V6.6 MR5 からは、次の記載を含めることで、生成される PDF の文書情報に含まれる作成日（/CreationDate）と更新日（/ModDate）を指定の値に設定することができます。[8)]

```
<meta name="creationdate" content="20190501T090000+09">
<meta name="modifydate" content="20190501T090000+09">
```

GitHub での Markdown 表示のようなスタイリングを実現するには、CSS ファイル「github-markdown-css」[9)]が利用できます（css/github-markdown.css）。

CSS 組版用の CSS ファイル（css/custom.css）を、「CSS ページ組版入門」[10)]などを参考にして記述し、テンプレート HTML ファイルから読み込まれるようにしておきます。

このステップによって、仮 HTML ファイル（work/all.html）が生成されます。

CSS 組版用の CSS ファイルを調整する場合は、この HTML ファイルを AH Formatter のグラフィカルユーザインターフェイスで 読み込み、組版結果を確認しながら行うと良いでしょう。

`build:doc-5` html-inline によるインライン化

Node.js パッケージ「html-inline」[11)]を用いて、HTML ファイルから参照されている CSS ファイルや画像ファイルを HTML ファイル内にインライン化（埋め込み）します。[12)]

```
npx html-inline work/all.html -b doc -o dist/all.html
```

このステップによって、必要な情報が全て含まれた HTML ファイル（dist/all.html）が生成されます。

---

8) AH Formatter V6.6 MR4 までの場合、「Coherent PDF Command Line Tools (cpdf)」を利用して設定する方法があります。
9) https://github.com/sindresorhus/github-markdown-css
10) https://www.antenna.co.jp/AHF/ahf_publication/index.html#CSSPrint
11) https://github.com/substack/html-inline
12) AH Formatter V6.6 MR5 までの場合、生成される PDF のプロパティ「ベース URL」に生成元 HTML ファイルの場所が表示され、同表示を空にするには、HTML ファイルから参照されている外部ファイルをインライン化した上で、PDF 出力時にコマンドラインパラメータ指定（-base " "）を行う必要がありました。AH Formatter V6.6 MR6 からは、これらの処理は不要となっています。

HTML から PDF へ変換

`build:doc-6` AHFCmd による変換

AH Formatter のコマンドラインインターフェイス AHFCmd を用いて、必要な情報が全て含まれた HTML ファイルを CSS 組版し、PDF ファイルとして出力します。

```
AHFCmd -d dist/all.html -p @PDF -pdfver 1.5 -base " " -x 4 -pgbar -o dist/
all.pdf
```

このステップによって、最終的な PDF ファイル（dist/all.pdf）が生成されます。

付録 ソースファイル抜粋

この文書の作成に用いたソースファイル一式のファイル構成と、主要なファイルの内容を挙げます。

ファイル構成

```
md2pdf
│  package.json
│
├─ css
│      custom.css
│      github-markdown.css
│
├─ dist
│      all.html
│      all.pdf
│
├─ doc
│      appendix.md
│      fig001.png
│      fig002.png
│      main.md
│      preface.md
│      template.html
│      toc.md
│
├─ node_modules
│
├─ scripts
│      ejs.js
│      mdit.js
│
└─ work
```

package.json

```
{
  "name": "md2pdf",
  "version": "1.0.0",
  "description": "Convert Markdown documents to PDF",
  "scripts": {
    "build:doc-1": "npx minicat doc/preface.md doc/toc.md doc/main.md doc/
appendix.md > work/all.md",
    "build:doc-2": "npx doctoc --notitle --maxlevel 3 work/all.md",
    "build:doc-3": "node scripts/mdit.js work/all.md work/all_md.html",
    "build:doc-4": "node scripts/ejs.js doc/template.html work/all.html",
    "build:doc-5": "npx html-inline work/all.html -b doc -o dist/all.html",
    "build:doc-6": "AHFCmd -d dist/all.html -p @PDF -pdfver 1.5 -base \" \" -x
4 -pgbar -o dist/all.pdf",
    "build": "npm run build:doc-1 && npm run build:doc-2 && npm run build:doc-3
&& npm run build:doc-4 && npm run build:doc-5 && npm run build:doc-6"
  },
  "keywords": [
    "Markdown",
    "PDF"
  ],
  "author": "2SC1815J",
  "license": "MIT",
  "devDependencies": {
    "anchor-markdown-header": "^0.5.7",
    "doctoc": "^1.4.0",
    "ejs": "^2.6.1",
    "eslint": "^5.16.0",
    "html-inline": "^1.2.0",
    "htmltidy2": "^0.3.0",
    "markdown-it": "^8.4.2",
    "markdown-it-implicit-figures": "^0.9.0",
    "markdown-it-named-headers": "0.0.4",
    "minicat": "^1.0.0"
  }
}
```

scripts/mdit.js

```
/*
 * md2html
 * Copyright 2019 2SC1815J, MIT license
 */
'use strict';
if (process.argv.length < 4) {
    console.error('Usage: node mdit.js input.md output.html');
    process.exit(1);
}

const header_instances = {};
const anchor = require('anchor-markdown-header');
const mdit = require('markdown-it')(
    {
        html: true
    })
    .use(require('markdown-it-named-headers'), {
        slugify: function(header) {
            if (header_instances[header] !== void 0) {
                header_instances[header]++;
            } else {
                header_instances[header] = 0;
            }
            const match = anchor(header, 'github.com',
header_instances[header]).match(/]\(#(.+?)\)$/);
            return match ? decodeURI(match[1]) : header;
        }
    })
    .use(require('markdown-it-implicit-figures'), {
        figcaption: true
    });

const { promisify } = require('util');
const fs = require('fs');

(async () => {
    const md = await promisify(fs.readFile)(process.argv[2], 'utf8');
    const html = mdit.render(md);
    await promisify(fs.writeFile)(process.argv[3], html, 'utf8');
})()
    .then(() => {
```

```
        console.log('Done.');
    })
    .catch((err) => {
        console.error(err);
        process.exit(1);
    });
```

scripts/ejs.js

```
/*
 * md2html
 * Copyright 2019 2SC1815J, MIT license
 */
'use strict';
if (process.argv.length < 4) {
    console.error('Usage: node ejs.js template.html output.html');
    process.exit(1);
}

const { promisify } = require('util');
const ejs = require('ejs');
const tidy = require('htmltidy2');
const fs = require('fs');

(async () => {
    const text = await promisify(ejs.renderFile)(process.argv[2]);
    const options = {
        doctype: 'html5',
        indent: 'auto',
        wrap: 0,
        tidyMark: false,
        quoteAmpersand: false,
        hideComments: true,
        dropEmptyElements: false,
        newline: 'LF'
    };
    const tidied = await promisify(tidy.tidy)(text, options);
    await promisify(fs.writeFile)(process.argv[3], tidied, 'utf8');
})()
    .then(() => {
        console.log('Done.');
    })
```

```
    .catch((err) => {
        console.error(err);
        process.exit(1);
    });
```

AH Formatter を用いた Markdown-PDF 変換事例紹介
2019 年 5 月 1 日