Markdownで効率Up

常三島技術部門
情報システムグループ　　片岡　由樹 (Yoshiki Kataoka)

Keywords：Markdown,Pdf,mermaid,ヘッドレスブラウザ

１．はじめに

文書を作成するために使うツールとしてMarkdown（マークダウン）記法を使ってテキストベースで作業するのは以前から有名であり自身も試したりしていたが，気に入ったツールが見当たらなくテキストエディタ等で処理してしまっていたが，少しだけ気に入りそうな要素があったので試したらハマってしまった。今回はその内容について紹介していく。

内容は平成30年前期に確認したものである。最新の環境等には変更等があることに注意いただきたい。また、技術報告の原稿様式に合わせた為に多少見にくい箇所もあるが，ご了承いただきたい。

２．Markdownとは

Markdownとは文書を記述するための軽量マークアップ言語である。テキストベースで文書を作成する際にルールに従った記述の仕方で文書構造を定義し，書式も変えることができる。アウトライン（文章構造）をつけながら執筆するのに適している。書式はどちらかといえばスタイル・デザインの要素が大きいが頻度が高いので一緒に定義されている。同じようなマークアップ言語としてはHTMLなどが一番一般的に知られているのではないだろうか。ちなみにHTMLはHyperText Markup Languageの略でWebページとして使用されている。HTMLもSGMLというマークアップ言語をモデルに策定されたものである。似たようなマークアップ言語は非常に多く，単純に類似の機能があり記述方法（ルール）が違うと考えていいだろう。

３．ツールの選択について

Markdownを取り扱うツールは数多く存在する。その中でおススメのツールはAtom, VSCodeである。どちらのツールがいいのかは比較したWebページが存在するので検索して参考にしていただければと思う。今回はAtomというエディタにアドオンとしてmarkdownに関するパッケージを導入する。Atomにはパッケージ自体も多く開発され，提供されている。その中から「markdown-preview-enhanced」[1]というパッケージ（以下，MPE）を選択した。

同じパッケージがVSCode用に準備されている。少しだけAtom版の方がパッケージの更新が速かったでのAtomを選んだ。Atomというエディタ自体は大変良くできている。数年前にもテスト使用した事があるが，起動時に動作が遅いのが難点である。その欠点は変わらない状態でした。

MPEはMarkdownのプレビュー表示する以外に他のパッケージのいいところを集めたようなパッケージになっている。プレビュー，TOC作成，プレビューとスクロール同期に加えてLaTeX数式埋め込みからPDF，HTML，eBook出力など多機能である。また，図の作成としてmermaid[2]を埋め込むことができるのでテキストベースのみで，簡単な図が作成できる。多機能であるが，思ったようにならない，もしくは機能しない事も多いように感じた。開発者の好みにより状況が微妙な機能もあるのだが，開発者の対応に期待したい。開発者の好みが私の好みとのシンクロ率が高かったのでMarkdownにハマってしまった要因でもある。

PDF出力はpuppeteer, phantomJS, princexmlなどに対応している。eBook出力はebook-convertが必要で，ダウンロードしてインストールして，pathを通しておくと使用することができる。

phantomJSの場合も，ソフトウェアをインス

トールしておく必要があるが，pdfだけではなくjpegやpngとしてファイル出力可能になっている。

puppeteerの場合も，ソフトウェアをインストールしておく必要がある。puppeteerはヘッドレスブラウザであり，Chromeと同じと考えてよい。Nodeライブラリのため Node.jsは導入しておく必要がある。

princexmlの場合はPrinceというソフトウェアを導入する。これもHTMLからPDFに変換するソフトウェアである。

PandocというMarkdown文書をほかのファイルに変換するソフトも有名で，LaTeXに変換している方も多い。その後にLaTeXを扱うWindows用日本語TeXのインストーラなどで環境を整えておくとlataxに変換して出力でき便利である。ただし，今回使用した環境ではMarkdownは対応するがmermaidなどの取り扱いが不十分のため，お勧めできない。

これらのツールを使うには文書の先頭にfront matterとして決まった様式(YAML)で設定を記述してファイル出力する。基本的にヘッドレスブラウザなので画面を見る用途ではないのでイメージしづらいがHTMLで有効な表現（CSS）などでスタイルを定義できる。デフォルトでスタイルが用意されているので，カスタマイズしなくてもそれなりに出力結果は見える。それぞれのツールのデフォルトがかなり異なるので，そのあたりでどのツールを使ってPDF出力するのかを選択するといいだろう。

４．Markdown風記法について

Markdown記法についても実は様々な流派が存在する。GitHubで使われる記法がメジャーである。また，記法についても日々更新されている。Markdownを扱うエディタやツールでは使用できる記法等が，ある程度オプションなどで選択できる。紹介する記法が使えない場合もあるので注意が必要だ。テキストベースのコードだけでも文章構造がわかりやすく表現できるが，プレビューで綺麗に表示して確認ができる。

４．１　文書構造

見出しの文字列の行頭に# （シャープ）を入れておく。#の後には必ずスペースを入れておく（表１）。

表１　Headers

```
# This is an <h1> tag  見出し1
## This is an <h2> tag  見出し2
### This is an <h3> tag  見出し3
#### This is an <h4> tag  見出し4
##### This is an <h5> tag  見出し5
###### This is an <h6> tag  見出し6
```

改行を含んだヘッダにするにはこの書き方ではHTMLのbrタグで記入するしかないが，h1・h2のみ別の書き方が用意されている。見出し1の改行後にイコールを2つ以上でh1に，見出し2の改行後にハイフンでh2にもなる（表２）。

表２　Headers without br tag

```
This is an <h1> tag
over one-line without br tag
====

```

イコールの後に空行を入れ忘れると下に記述した文字列もヘッダになってしまうこともあるので，要素の境目は空行をいれると良い。もちろん見出しの前にも空行が入っている。

４．２　強調表示

書式であるが強調していることはプレビューしなくてもわかるので便利である。斜体と太字の組み合わせも使える（表３）。

表３　Emphasis

```
*italic* , _italic_ , *イタリック*
**bold** , __bold__ , **太字**
~~strikethrough~~ , ~~取り消し線~~
*You **can** combine them.*
```

４．３　リスト表示

リストアイテムの下の文章はインデントが

自動で付く（表４）。数字は1のままでも自動でナンバリングされる。似たような形式として後述のタスクリストがある。メモ書きとしての文書に便利な機能である。ちなみに，Wordでも似たようなことはオートコレクトで箇条書きにすることもできる。

表４　Lists

```
Unordered List
* Item 1
* Item 2
   * Item 2a
   * Item 2b
* Item 3

Ordered List
1. Item 1
1. Item 2 1. Item 3
    1. Item 3a
    1. Item 3b
```

４．４　画像

後述のLinksとほぼ同じ書式だが少し異なる（表５）。ファイルを指定せずにデータURIスキームを入れて画像をテキストファイル（マークダウンファイル）に埋め込むこともできる。

表５　Images

```
![GitHub Logo](/images/logo.png)
Format: ![Alt Text](url)
```

「data URI scheme」という書式で「data:」から始まるのが特長だ。data:[メディアタイプ];[エンコード方式],[データ]の順番で記述する。つまり![Alt Text](.....)のようになる。アイコンほどの画像でもテキストで埋め込むとデータ量は多く感じる。データURIスキームはjpgとかでもOKだ。JPEGを埋め込む場合は「data: image/png」の部分を「data:image/jpeg」とする。ただしデータが大きい場合はファイルにして読み込む方がいいだろう。今回使用しているMPEの場合は@importをつかってもいい。

Chrome，Firefoxなどのブラウザで開発ツールを使えばWeb上の画像ファイルをURIとしてデータ表示できるので，それをコピペするとよい（著作権の取り扱いは自己責任）。

- Chromeの場合

適当な画像を右クリックし新しいタブに表示し，検証を選択する。DevToolsのElementsタブが表示されるので，ソースタブに変更するとBase64文字列が表示されている。

- Firefoxの場合

「開発ツール」で画像を選択して右クリック
「コピー」で「画像のデータURI」

- Windowsの場合

コマンドプロンプトで変換する。

`certutil -encode arrow_up.png base64.txt`

画像を入力してテキストファイルを出力する。出力されたファイルで最初行と最終行は無視して，改行もなくすと問題ない。

HTMLはそのままプレビューされるのでSVGはファイルの中身を記述することもできる（表６）。もちろんファイルも読み込める。画像サイズを変更するには以下のようにする。細かい設定はHTMLタグにすると自由度が高く指定できる。

表６　Images(SVG)

```
![](test_logo.svg| width=100)
![WIDTHxHEIGHT](test_logo.svg =100x100)
![Skip height](test_logo.svg   =100x)
<img  style="float:left;"  height="100"  alt=""
src="test_logo.svg"><br clear=both>
```

MPEの拡張である@importを使うなら他の属性も指定できる。imgタグのbase64の埋め込みを@importに使っても機能しないので，注意が必要だ。必要に応じてHTMLの素のコードでも構わない。

４．５　リンク等

Web上のページのURLをそのまま記述するとは自動的にリンク付きになる（表７）。

同一文書の見出しを[こちら](#Links)のように表記できる（Linksという見出しが前提）。HTMLとして見出しには自動でname属性が付けられている。ただし見出しはアルファベットのみで構成されたときにのみ機能する。

表７　Links

```
http://github.com - automatic!
[GitHub](http://github.com)
<http://qiita.com>
```

引用文もわかりやすい表現である（表８）。

表８　Blockquote

```
アインシュタインいわく
> 天才とは努力する
> 凡才のことである。
```

3つ以上のHyphens,Asterisks,Underscoresで水平線を引くことができる（表９）。もちろんHTMLのhrタグでも問題ない。記号の間にスペースがあっても問題ない

表９　Horizontal Rule

```
---
***

___
* * *
```

４．６　コード等

インラインコードの場合はbackticks（バッククォート）で囲む（表１0）。複数行のブロックコードを表示するにbackticksを3つで囲む。バッククォートはシングルクォートによく似ているが別のものだ。Windows用のキーボードの場合は，Shiftキーを押しながら「@」の書いてあるキー（「P」のキーの1つ右のキー）を押すと「`」が入力することができる。

行番号を付けて表示するにはクラスを付加して表現できる（表１１）。言語により定義されたワード・予約語があるので3つのbackticksの後に言語を指定しておくとコードが見やすくなる。

箇条書き（リスト）とほぼ同じ記法だがチェックマークが付けられるタスクリストは非常に役立つ（表１２）。

表１０　Code

````
Inline `インラインコード`
(空行)
```
 code block.
```
(空行)
````

表１１　Code（line-numbers）

````
```javascript {.line-numbers}
function add(x, y) {
  return x + y
}
```
````

表１２　Task lists

```
- [x]@mentions,#refs,[links](),formatting,
      and <del>tags</del> supported
- [x] list syntax required  (any unordered or
ordered list supported)
- [x] this is a complete item
- [ ] this is an incomplete item
```

表１３　Tables

```
1st Header | 2nd Header
---- | ---
Content 1 | Content 2
Content 3 | Content 4
```

表１４　Tables (aligned)

```
| Left align<br>:---  | Right align<br>---: |
Center align<br>:---: |
|:-----------|------------:|:------------:|
| This       |        This |     This     |
| column     |      column |    column    |
| will       |        will |     will     |
| be         |          be |      be      |
| left       |       right |    center    |
| aligned    |     aligned |    aligned   |
```

表形式について様々な流派があるが，以下のようになっている（表１３）。先頭と末尾のパイプを省略可能である。左寄せや中央寄せ・右寄せなども指定できる（表１４）。

５．Markdown Preview Enhancedによる拡張

MPEによる拡張を列挙していこう。

５．１　MPE拡張

数式の表示としてLaTeX形式(Katex)とMathjax形式かを設定で選ぶことができる（表１５）。

表１５　Katex

```
$ y=¥sqrt{x+¥frac{1}{2}} $, ¥(y=¥sqrt{x}¥) will be rendered inline.

$$y=¥sqrt{x}$$

¥[y=¥sqrt{x}¥]

```math
y=¥sqrt{x}
```
```

Katexはブラウザ上でも動いて，Texの数式を描画するものでMathJaxより速い。しかし日本語対応が不十分である。日本語は¥text{}で囲むなどをする。また，数式表示しようとするがHTMLタグなどで囲ったときに機能しないときがあり，注意が必要である。

見出しの拡張としてidやclassをつけることができる（表１６）。これにより，より細かなスタイル設定が実現できる。使用するかどうかはAtomのパッケージの設定を変更する必要ある。さらにPandocパーサだと表示されない項目がある（表１７）。

略語は可能だが，エディタのmarkdownハイライターが機能しないのでわかりにくくなるので注意が必要である（表１８）。

Links拡張としてWiki風のリンクも使うことができる（表１９）。Wikiと同じような使い方としてリンクをクリックしてリンク先がなかったら新規作成される。ファイル名に拡張子(md)が自動的に補完される。Atomではファイルが開かれる。通常のリンクの使い方では拡張子は補完されない。

表１６　Headers extension

```
# This heading has 1 id           {#my_id}
# This heading has 2 classes {.class1 .class2}
```

表１７　Emphasis extension

```
Emoji :smile: :fa-car:
==marked==
脚注元 [^1]
Superscript　30^th^
Subscript　H~2~O
[^1]: 脚注でしめす内容
```

表１８　Abbreviation

```
*[HTML]: Hyper Text Markup Language
*[W3C]:　World Wide Web Consortium
The HTML specification is maintained by the W3C.
```

表１９　Links(Wiki)

```
[[WikiLink]]
[[Link Text|WikiLink]]
[Link Text](URL_to_wiki)
```

Tables拡張はMPEパッケージの設定で有効にしなければいけないがセルの連結ができる。（デフォルトでOFFになっている。Markdown extraというPHP系のtable表現の拡張である。表２０）

表２０　Tables extension

```
| Header | Header2 |
| ---: | --- |
| Content1 | Content2 |
| ^ | Content3 |
|　|　|

| Header | Header2 | Header3 |
| :--- | ---: | :---: |
| > | 1 | 3 |
| 2 || 4 |
```

目次を挿入することができる（表２１）。プレビューにおいてはESCで自動表示されるが，データが反映されないようにするにはheadersの所に{ignore=true}を付け加える（タブを入れてタイトル文章とは区別しておくのがポイントである）。

表２１　Toc

```
[toc]
```

Ctrl+Shift+pでコマンド呼び出しをして「Markdown Preview Enhanced: Create Toc」を選択すると表２２の文字が挿入される。

表２２　Create Toc

```
<!--@import "[TOC]" {cmd="toc" depthFrom
    =1 depthTo=6 orderedList=false} -->
```

これによりmarkdown-tocというパッケージをインストールする必要がなくなる。Importの仕様が変更された加減なのか，出力がされない。

目次に関してはfront-matterで細かな設定は変えられる（表２３）。

表２３　Toc Setting

```
---
toc:
  depth_from: 1
  depth_to: 6
  ordered: false
---
```

CriticMarkupを埋め込むこともできる。MPEパッケージの設定で有効にしなければいけない。（デフォルトでOFFになっている。）CriticMarkupとはバージョン管理ではなくソースに履歴を残していく手法で，画面には表れない。コメントみたいな使い方もできる。

・Addition {++ ++}
・Deletion {-- --}
・Substitution {~~ ~> ~~}
・Comment {>> <<}
・Highlight {== ==},{>> <<}

５．２　MPE拡張（インポート）

ファイルをインポートできる（表２４）。サポートするファイルは以下の通りである。

* 画像　(jpeg ,gif, png, apng, svg, bmp)
* テーブルデータ　(csv)
* 図として
  * mermaid(.mermaid)
  * graphviz (.dot)
  * PlantUML(.plantuml, .puml)
* 組み込みHTML
  * .html
  * .js（<script src = "your_js" >< /script >)
  * .less　（ローカルのみ対応）
* .css(<link rel="stylesheet" href="your_css" >)
* .pdf（ただしpdf2svg が必要）
* markdown file

表２４　Import

```
@import "test_logo.svg"
@import "test.csv"
```

属性を追加してインポートすることができる（表２５）。挿入するPDFのページの指定などである。

表２５　Import plus alpha

```
@import "test.png"{width="300px" height=
    "200px" title="my title" alt="my alt"}
@import "test.pdf"{page_no=1}
@import "test.pdf"{page_begin=2
                   page_end=4}
@import"test.puml"{code_block=true
                   class="line-numbers"}
@import "test.py" {class="line-numbers"}
@import "test.json" {as="vega-lite"}
@import "test.py" {cmd="python3"}
```

５．３　MPE拡張（図，コードチャンク）

コードの記述方法（3つのハイフンで囲む）で種類をして図を表示することができる。プロパティを加えることもできる。flow charts, sequence diagrams, mermaid（表２６）, PlantUML, WaveDrom, GraphViz, Vega & Vega-lite, Ditaa diagramsなどが対応している。コードチャンクを使ってコマンドを実行した結果を表示することもできる（表２７）。出力がSVGなら図が, テキストなら文字情報を挿入できる（TikZ, Python Matplotlib, Plotly など）。設定にて有効にする必要がある。実行もショートカット等でする。mdファイル自体を日本

語のないディレクトリに配置してそれぞれのアプリケーションへのパスが設定済みの場合に機能することに注意が必要である。Windowsの日本語のディレクトリパスは不具合を生み出すのでCドライブ直下に作業用ディレクトリを作成しておくことを推奨する。

表２６　Mermaid

```
```mermaid
graph LR
   A --> B;
   B --> C;
   C --> A;
```
```

表２７　Code Chank

```
```latex {cmd=true}
¥documentclass{standalone}
¥begin{document}
    Hello world!
¥end{document}
```
```

５．４　MPE拡張（スライド）

スライド形式でページを付けて表示することができる（表２８）。スライドとして表示するには

1. Front Matter に設定を追加する
2. スライドの仕切りごとに<!-- slide -->のコメントを追加する。

HTMLのコメントに意味のある文字列を挿入して，認識させている。基本的に中央揃えになってしまうのでスタイルを作っておいた方が良さそうだ。jsのスライドライブラリは他にもあるので，そちらを使うという手もある。

表２８　Slide Setting

```
---
presentation:
width: 800
height: 600
---
```

５．５　MPE拡張（絵文字）

Reveal.jsのemojiがMPEパッケージでは使用できる。パッケージに入っているReveal.jsのバージョンの問題だと思うが，すべての絵文字が正確には表示されなかった。

６．HTMLの活用

HTMLを直接記述してもうまく機能する。SVGファイルならHTML内に記述できる（HTML5）ので直接記述することもできる（表２９）。

表２９　SVG include

```
<svgid="test" xmlns="http://www.w3.org/2000
/ svg">
<circle class="icon" cx="50" cy="50" r="50"/>
</svg>
<style>
.icon{
   fill: red;
}
</style>
```

プレビューやPDF出力にはスタイルシートが機能する。普通にHTML内に記述するように宣言するとよい。スタイルシートもimportによるlessファイル読込(@import "my-style.less")も可能だ。

コメントとしてHTMLのコメント(<!---と-->)を使ってプレビューには表示されないコメントを記述できる（表３０）。ただしPandocなどで処理する際にはパースの際に違う解釈を与えてしまうので,ハイフンを3つにして使用しておくことが推奨される。

表３０　Comment

```
<!---  ここはコメント　--->
```

PDF出力の際の改ページの指定方法だが,Pandocではpagebreakとしたらいいというネット情報があるが,それはTexへの変換をするので¥newpageや¥pagebreakがTexの改ページに対応するからだ。Pandocを使わないAtom+MPEの環境でのPDF出力を考えた場合に一番

確実なのはCSSをつかった方法だ（表３１）。

表３１　Pagebreak

```
<div style="page-break-before:always">
</div>
```

段落とは別に表中で改行する場合は
を入れる。そのほかにもHTMLでつかわれる空白等の特殊文字記号は使える。空白( )，小なり(<)，大なり(>)，クォーテーション(")，アンパサンド(&)やコピーライト(©)である。

ブロック要素（divなど）におけるテキストの右寄せなどはHTMLの表現を使う（表３２）。

インデントする時も同様である（表３３）。インデントする箇所に何らかのブロックオブジェクトがある場合はtext-indentでは対応していないのでmargin-leftなどのブロックのスタイルシートを使うこともある（表３４）。

表３２　Text-align

```
<div style="text-align:right">右寄せ</div>
```

表３３　Text-indent

```
<div style="text-indent:1em">
インデントする文章
</div>
```

表３４　Margin-left

```
<div style="margin-left:10em">
インデントするブロック
</div>
```

児童向け文書作成に必要になってくるフリガナもHTMLならば作成できる（表３５）。

表３５　Ruby

```
<ruby>漢字<rp>(</rp><rt>かんじ</rt><rp>)
</rp></ruby>
```

HTML5から追加された<details>についてはアコーディオンで表示できる（表３６）。Summaryの中身により改行を挿入しないと動作がおかしかった。

表３６　Details

```
<details style="margin:10pt;margin-left:4em">
<summary>関連する数式</summary>

$r_1=a/2$ ,$r_2=b/2$ ,
$¥theta = ¥frac{2¥pi}{360}¥times f$
</details>
```

７．MPE使用上のTips

円記号を表示する際に少し困ってしまったので記述しておく。HTML記法を使った方法など様々な表現方法があるが，思い通りに円記号を表示するのは難しい。以下のようなパターンが候補として挙げることができる。一番良かったのは<pre></pre>タグの中で実体参照¥を書くことであった。

1. そのまま「￥」
2. 文字の前後を「backtick 3つ」ではさむ
3. 先頭行と最終行を「backtick3つ」
4. 空行1行と先頭に4文字スペース
5. <code>タグではさむ
6. <pre>タグではさむ
7. <pre><code>タグではさむ
8. 実体参照　¥

インラインなら¥で表示する。ブロック表示するときに<pre>タグではさむのは<code>タグが入ると背景が灰色になるからだ。また，<pre>タグは前には改行が入ってしまうのでスタイルで処理するなどの工夫が必要である。

８．Front Matter使用上のTips

文書に様々な情報や挙動を記述できる。ファイルの先頭に項目と値をYAML形式で記述する。パッケージの設定で表示をするかどうかの選択項目があるのでプレビューでも確認できる。Pandocの設定が他の設定より優先されているのはPandocを使う方が多いからだろう（表３７）。

表３７　PandocのWord出力(FrontMatter)

```
---
title: パソコン講習会向け資料
author: yoshiki kataoka
date: 2018/06/26
output: word_document
---
```

便利な機能としてexport_on_saveの設定をしておくと，ファイル保存するたびにPDFが作成され，既存のファイルも更新される（表３８）。

PDF文書のヘッダやフッタも表現できる。ただしテキスト位置をHTML風に記述するのがスマートではない。もしかするとphantomjsの設定としてテキスト位置のプロパティがあるかもしれない。

表３８　PhantomjsのPDF出力(FrontMatter)

```
---
# front matter (YAML)
export_on_save:
  phantomjs: "pdf"
phantomjs:
  orientation: portrait
  border: 10mm
  header:
    height: "10mm"
    contents: '<div style="text-align:
                    center ;">page.title</div>'
  footer:
    height: 10mm
    contents:
      first: 'Cover page'
      #2: 'Second page'
      last: 'Last Page'
      default: '<div style="text-align:
                    center ;">{{page}}</div>'
title:  イライラ棒解説シート
author: Yoshiki Kataoka
date:    2018-07-09
lang: ja-jp
papersize:   A4
---
```

９．mermaid 使用上のTips

mermaidはFontAwesome[3]のWeb iconが使用できる（表３９）。fa-spinをclassにつけると回転するが回転中心がずれている。fa-2xをclassにつけると倍率が2倍になりアイコンが大きくなるが，ノードまで反映されない。

表３９　mermaid (Web icon)

```
```mermaid{align=center}
graph LR
  A(fa:fa-twitter)
  A --> B(fa:fa-apple)
  B --> C[fa:fa-ban]
  B --> D(fa:fa-spinner);
  B --> E(fa:fa-camera-retro)
  %%class D fa-spin
  %%class E fa-2x
  %% fa-3x,fa-4x,fa-5x
```
```

表４０　FontAwesome

```
<span class="fa-stack fa-lg">
 <i class="fa fa-heart fa-stack-2x"></i>
 <i class="fa fa-apple fa-stack-1x fa-inverse">
 </i>
</span>
```

mermaid 内のFontAwesomeのバージョンによると思われるが最新のFontAwesomeにあるアイコンが表示されるとは限らない。FontAwesome自体は<i>でアイコンを使う形である。これはmermaid以外でもMarkdown中ならHTMLと同じだから使用できる（表４０）。

MPEパッケージならfa-で始まるクラスが使用できるようになっている。FontAwesomeではそうではないfab fa-address-bookのような表記もあり注意が必要だ。リストで利用する場合は ulのclass にfa-ul をつけ，liのclassにfa-liをつける。

fa-border,fa-rotate-90,fa-rotate-180,fa-flip-horizontal，fa-flip-verticalなどのクラスも使うと簡単にアイコンを表示・操作できる。fa-stackクラスを使えば重ね合わせもできる。

１０．業務での活用

実際にこれらのツールを業務に活用した。平成30年8月4日（土）及び5日（日）に開催された「第22回科学体験フェスティバル in 徳島」（徳島大学理工学部主催）において出展したブース（イライラ棒で遊ぼう）において技術的な仕組みのわかるデモ機を作成した際に解説する文書（ラミネート加工）を用意した。その文書は文章とともに図を配置させた文書であった。これらのすべてをAtom+MPEにてPDF文書として作成した。図はmermaidを使いルビ付きの文書である（図１～図３）。つまり，テキストベースのファイル一つで完結している。

図１　Mermaidによる作図例(１)

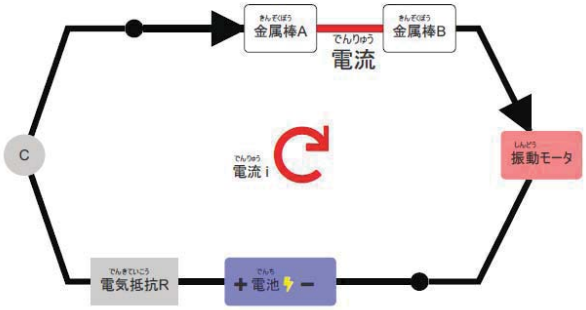

図２　Mermaidによる作図例(２)

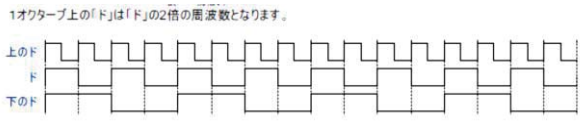

図３　Mermaidによる作図例(３)

もう一つ業務での活用例を紹介しよう。平成30年度徳島大学パソコン講習会の講師をした際の配布資料を作成した。アウトラインや箇条書きの作成が非常に楽にでき，テキストファイルなのでバージョン管理が非常に楽にできた。簡単な図はデータURIスキームによる埋め込みであった。

１１．これからの課題

これらのツールを業務にも活用できたが，不十分に感じた箇所もあった。Pythonのグラフ描画ライブラリMatplotlibを使った図形描画，プレビュー表示のスタイルシートのカスタマイズ，textlintパッケージを使った校正について今後テストを出来ればと考えている。また，Atomの起動時間の遅さにイライラとすることが多くなったので，VSCodeに乗り換えも検討している。

１２．最後に

ツールは非常に高機能にできているが，それを使いこなすには少し時間がかかる。

HTMLやCSS,SVGの知識が多少あったので今回はすぐに業務に活用できたが，日本語情報が少ないのが残念である。Atomに導入したパッケージも紹介しておくので，参考になれば幸いである（表４１）。

表４１　Recommand Package (Atom)

- Japanese-menu
- Activate-power-mode
- Docblockr
- File-icons
- Platformio-ide-terminal
- Show-ideographic-space

参考文献

[1] https://shd101wyy.github.io/markdown-review-enhanced/#/
[2] https://mermaidjs.github.io/
[3] https://fontawesome.com/