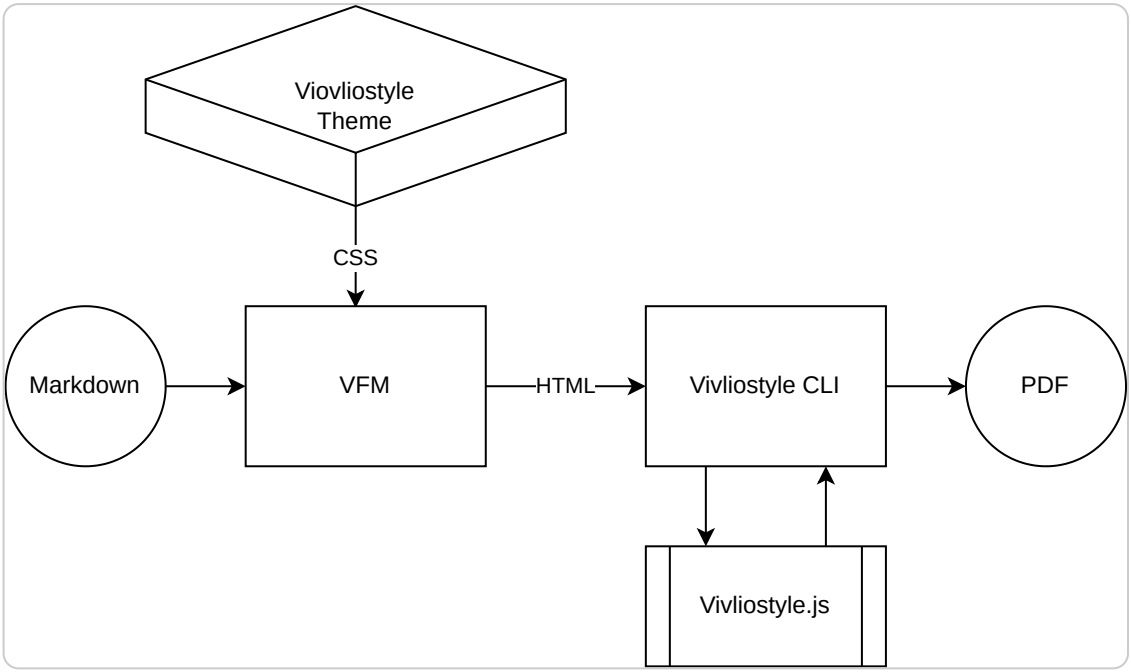
VFM (Vivliostyle Flavored Markdown) ドキュメント

Table of Contents

Vivliostyle Flavored Markdown


Viovliostyle
Theme
CSS
Markdown
VFM
HTML
Vivliostyle CLI
PDF
Vivliostyle.js

Principles

1. Rule of least surprise
    - Should be lined and matched to another Markdown syntax.
2. (Mostly) backward-compatible syntax. should not be incorrectly rendered in Markdown editor like Typora.

Vivliostyle Flavored Markdown

Vivliostyle Flavored Markdown (VFM)は本の執筆のために最適化されたMarkdown記法です。Vivliostyleとその関連プロジェクトのために標準化され、公開されています。VFMはCommonMark (https://commonmark.org/)およびGitHub Flavored Markdown (GFM) (https://github.github.com/gfm/)をベースにして実装されています。

コード(Code)

VFM

````
```javascript
function main() {}
```
````

HTML

```
<pre class="language-javascript">
<code class="language-javascript"><span class="token keyword">function</span> <span class=
</code></pre>
```

CSS

```
pre {
}
pre code {
}
```

VFMは構文強調にPrism (https://prismjs.com/)を利用しています。

キャプションをつける(with caption)

VFM

````
```javascript:app.js
function main() {}
```
````

あるいは

````
```javascript title=app.js
function main() {}
```
````

HTML

```
<figure class="language-javascript">
  <figcaption>app.js</figcaption>
  <pre>
    <code class="language-javascript">
      function main() {}
    </code>
  </pre>
</figure>
```

CSS

```
figure[class^='language-'] {
}
figure[class^='language-'] figcaption {
}
figure[class^='language-'] pre {
}
figure[class^='language-'] pre code {
}
```

脚注(Footnotes)

脚注の定義はPandoc (https://pandoc.org/MANUAL.html#footnotes)のようになります。

VFM

```
VFM は GitHub リポジトリで開発しています[^1].
イシューは GitHub で管理します[^イシュー].
脚注は行の中に記述することもできます^[この部分が脚注です。].

[^1]: [VFM](https://github.com/vivliostyle/vfm)

[^イシュー]: [イシュー](https://github.com/vivliostyle/vfm/issues)
```

HTML

```
    <p>
      VFM は GitHub リポジトリで開発しています<a id="fnref1" href="#fn1" class="footnote-ref" r
      イシューは GitHub で管理します<a id="fnref2" href="#fn2" class="footnote-ref" role="doc-
      脚注は行の中に記述することもできます<a id="fnref3" href="#fn3" class="footnote-ref" role="
    </p>
    <section class="footnotes" role="doc-endnotes">
      <hr>
      <ol>
        <li id="fn1" role="doc-endnote"><a href="https://github.com/vivliostyle/vfm">VFM</
        <li id="fn2" role="doc-endnote"><a href="https://github.com/vivliostyle/vfm/issues
        <li id="fn3" role="doc-endnote">この部分が脚注です。<a href="#fnref3" class="footnote-
      </ol>
    </section>
```

CSS

```
.footnotes {
}
```

フロントマター／前付け(Frontmatter)

フロントマター／前付けはMarkdownファイル単位でメタデータを定義するための方法です。ファイルの冒頭へYAMLで記述します。

YAMLの解析にはjs-yaml (https://www.npmjs.com/package/js-yaml)を使用しています。スキーマはJSON_SCHEMA (https://yaml.org/spec/1.2/spec.html#id2803231)です。

js-yamlの解析は `key:` を `key: null` にします。しかしVFMはこれを空の文字列として扱います。属性値のプロパティとして `key:` または `key:""` が指定された場合は `key=""` を出力します。

VFM

```
---
id: 'my-page'
lang: 'ja'
dir: 'ltr'
class: 'my-class'
title: 'Title'
html:
  data-color-mode: 'dark'
  data-light-theme: 'light'
  data-dark-theme: 'dark'
body:
  id: 'body'
  class: 'foo bar'
base:
  target: '_top'
  href: 'https://www.example.com/'
meta:
  - name: 'theme-color'
    media: '(prefers-color-scheme: light)'
    content: 'red'
  - name: 'theme-color'
    media: '(prefers-color-scheme: dark)'
    content: 'darkred'
link:
  - rel: 'stylesheet'
    href: 'sample1.css'
  - rel: 'stylesheet'
    href: 'sample2.css'
script:
  - type: 'text/javascript'
    src: 'sample1.js'
  - type: 'text/javascript'
    src: 'sample2.js'
vfm:
  math: false
  theme: 'theme.css'
  partial: false
  hardLineBreaks: false
  disableFormatHtml: false
author: 'Author'
---

テキスト
```

HTML

```
<!doctype html>
<html data-color-mode="dark" data-light-theme="light" data-dark-theme="dark" id="my-page"
  <head>
    <meta charset="utf-8">
    <title>Title</title>
    <base target="_top" href="https://www.example.com/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="theme-color" media="(prefers-color-scheme: light)" content="red">
    <meta name="theme-color" media="(prefers-color-scheme: dark)" content="darkred">
    <meta name="author" content="Author">
    <link rel="stylesheet" href="sample1.css">
    <link rel="stylesheet" href="sample2.css">
    <script type="text/javascript" src="sample1.js"></script>
    <script type="text/javascript" src="sample2.js"></script>
  </head>
  <body id="body" class="my-class foo bar">
    <p>テキスト</p>
  </body>
</html>
```

CSS

```
.my-class {
}

.foo.bar {
}
```

定義済みのプロパティ(Defined properties)

プロパティ	データ型	説明
`id`	`String`	`<html id="...">`
`lang`	`String`	`<html lang="...">`
`dir`	`String`	`<html dir="...">`、指定可能な値は `ltr`、`rtl` または `auto`。
`class`	`String`	`<html class="...">` と `<body class="...">` へ反映される。
`title`	`String`	`<title>...</title>` がない場合、コンテンツの最初の見出しがタイトルになる。
`html`	`Object`	`<html key="value">`、キーと値のペアが `<html>` の属性になる。
`body`	`Object`	`<body key="value">`、キーと値のペアが `<body>` の属性になる。
`base`	`Object`	`<base key="value">`、キーと値のペアが `<base>` の属性になる。
`meta`	`Object[]`	`<meta key="value">`、キーと値のペアが `<meta>` の属性となる。
`link`	`Object[]`	`<meta key="value">`、キーと値のペアが `<link>` の属性となる。
`script`	`Object[]`	`<script key="value">`、キーと値のペアが `<script>` の属性となる。
`vfm`	`Object`	VFMの設定。
`head`	-	将来の利用に予約済み。
`style`	-	将来の利用に予約済み。
その他	`String`	`<meta name="key" content="value">`、キーと値のペアは単独の `<meta>` になる。

vfm

プロパティ	データ型	初期値	説明
`math`	`Boolean`	`true`	数式を有効にする。
`partial`	`Boolean`	`false`	Markdown部分だけをHTML化する。 `<body>` 以上は出力されない。
`hardLineBreaks`	`Boolean`	`false`	空白を必要とせず強制改行の位置へ ` ` を追加。
`disableFormatHtml`	`Boolean`	`false`	HTMLの自動整形を無効にする。
`theme`	`String`	-	Vivliostyleのthemeパッケージか、CSSファイルをそのまま指定する。

プロパティのオプション(Priority with options)

同じ目的の仕様が複数ある場合、優先順位は以下の通りになります。

1. フロントマター／前付け
2. VFMオプション

フロントマター／前付けにおいて `html` プロパティでルートの `id` と重複した `id` が指定されている場合、ルートで定義された方が優先されます。

```
---
id: 'sample1'
html:
  id: 'sample2'
---
```

この例では `sample1` が採用されました。

```
<html id="sample1">
</html>
```

classプロパティの結合(Merge class properties)

最上層と `html`、`body` の `class` プロパティはスペース区切りで結合されます。

```
---
class: 'root'
html:
  class: 'html'
body:
  class: 'body sample'
---
```

以下は結合された例です。

```
<html class="root html">
  <body class="root body sample">
  </body>
</html>
```

強制改行（オプション）(Hard new line (optional))

- 改行すると行末へ `<br/>` が付きます
- 2行連続の改行は新しいブロックを生成します

この機能はオプションです。Node.js APIはオプションとして `hardLineBreaks: true`、CLIでは `--hard-line-breaks` を指定することで有効化されます。

VFM

```
はじめまして。

Vivliostyle Flavored Markdown（略して VFM）の世界へようこそ。
VFM は出版物の執筆に適した Markdown 方言であり、Vivliostyle プロジェクトのために策定・実装されました。
```

HTML

```
<!-- hardLineBreaks: true -->
<p>はじめまして。</p>
<p>
  Vivliostyle Flavored Markdown（略して VFM）の世界へようこそ。<br />
  VFM は出版物の執筆に適した Markdown 方言であり、Vivliostyle
  プロジェクトのために策定・実装されました。
</p>

<!-- hardLineBreaks: false (Default) -->
<p>はじめまして。</p>
<p>
  Vivliostyle Flavored Markdown（略して VFM）の世界へようこそ。 VFM
  は出版物の執筆に適した Markdown 方言であり、Vivliostyle
  プロジェクトのために策定・実装されました。
</p>
```

CSS

```
p {
}
```

画像(Image)

VFM

```
![](./fig1.png)
```

HTML

```
<img src="./fig1.png" />
```

CSS

```
img {
}
```

単一行キャプション(with caption and single line)

単一行で書かれた画像はキャプション付きで `<figure>` 内へ包み込みます。

VFM

```
![
Figure 1
](./fig1.png)

![
Figure 2
](./fig2.png "Figure 2"){id="image" data-sample="sample"}

text ![
Figure 3
](./fig3.png)
```

HTML

```
<figure>
  <img src="./fig1.png" alt="Figure 1">
  <figcaption aria-hidden="true">Figure 1</figcaption>
</figure>
<figure>
  <img src="./fig2.png" alt="Figure 2" title="Figure 2" id="image" data-sample="sample">
  <figcaption aria-hidden="true">Figure 2</figcaption>
</figure>
<p>text
  <img src="./fig3.png" alt="Figure 3">
</p>
```

CSS

```
figure img {
}
figure figcaption {
}
```

数式(Math equation)

MathJax (https://www.mathjax.org/)により処理したHTMLを出力します。

数式は標準で有効化されています。無効にする場合は以下を指定してください。

- `stringify` APIのオプション: `math: false`
- `VFM` APIのオプション: `math: false`
- CLIオプション: `--disable-math`
- フロントマター: `vfm:` プロパティへ `math: false`
  - 参照: フロントマター／前付け(Frontmatter)
  - これは `stringify` よりも優先されますが `VFM` ではそうなりません

VFMにおけるMathJaxのインライン記法は `$...$`、ディスプレイ記法は `$$...$$` となります。

また `$x = y\n1 + 1 = 2$` や `$$\nx = y\n$$` のような複数行もサポートします。ただし `$x = y\n\n1 + 1 = 2$` のように空行 `\n\n` がある場合は段落として分かれるため数式にはなりません。

OK:

- `$...$`, `$$...$$` ...範囲指定が一致
- `$...\n...$`, `$$\n...\n$$` ...同じ段落内
- `$...\$...$`, `$...\$...\\\$..$`, `$$...\$...\\\$...$$` ...奇数個の `\` により `$` をエスケープ（無効化）する

NG:

- `$...$$`, `$$...$` ...範囲指定が不一致
- `$...\n\n...$`, `$$...\n\n...$$` ...改行によって段落へ分離されてしまう
- `$ ...$` ...スペース(スペース、タブ文字、改行など)、インライン開始の `$` 直後に ` ` がある
- `$... $` ...スペース(スペース、タブ文字、改行など)、インライン終了の `$` 直前に ` ` がある
- `$...$5` ...インライン終了の `$` 直後に数字 `0...N` がある

VFM

```
inline:$x = y$

display: $$1 + 1 = 2$$
```

HTML

`math` が有効で数式記法か `<math>` タグが存在する場合はMathJax処理用の `<script>` も出力します。

```
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script async src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.9/MathJax.js?confi
  </head>
  <body>
    <p>inline: <span class="math inline" data-math-typeset="true">"\(x = y\)</span></p>
    <p>display: <span class="math display" data-math-typeset="true">$$1 + 1 = 2$$</span></
  </body>
</html>
```

CSS

```
.math.inline {
}

.math.display {
}
```

そのままのHTML (Raw HTML)

VFM

```
<div class="custom">
  <p>Hey</p>
</div>
```

HTML

```
<div class="custom">
  <p>Hey</p>
</div>
```

Markdownをつける(with Markdown)

VFM

```
<div class="custom">

# Heading

</div>
```

HTML

```
<div class="custom">
  <section class="level1">
    <h1 id="heading">Heading</h1>
  </section>
</div>
```

ルビ(Ruby)

VFM

```
This is {Ruby|ルビ}
```

HTML

```
This is <ruby>Ruby<rt>ルビ</rt></ruby>
```

CSS

```
ruby {
}
ruby rt {
}
```

ルビにおけるパイプのエスケープ(Escape pipe in ruby body)

区切り記号となるパイプ `|` をエスケープ(無効化)したい場合は直前に `\` を追加します。

VFM

```
{a\|b|c}
```

HTML

```
<p><ruby>a|b<rt>c</rt></ruby></p>
```

セクション分け(Sectionization)

見出しを階層的なセクションにします。

- 見出しの行が `#` ではじまり同数以上の `#` で終わる場合はセクションを分けません
  - `### Not Sectionize ###` （同じ数の `#` で囲まれている）--セクション分けしない
  - `### Sectionize ##` （閉じの `#` の数が足りない）--セクション分けする
- `#` だけからなる行により `#` の数と一致する深さのセクションを終了させることができる
  - 例: `### Heading 3` で開始したセクションは `###` で終了させられる
- 親が `blockquote` の場合はセクションを分けません
- 見出しの深さへ一致するように、セクションの `levelN` クラスを設定します

VFM

```
# Plain

# Introduction {#intro}

# Welcome {.title}

# Level 1

## Level 2

### Level 3

##

Level 2 was ended by `##`.

## Not Sectionize {.just-a-heading} ##

> # Not Sectionize in Blockquote
```

HTML

```
<section class="level1">
  <h1 id="plain">Plain</h1>
</section>
<section class="level1">
  <h1 id="intro">Introduction</h1>
</section>
<section class="level1">
  <h1 class="title" id="welcome">Welcome</h1>
</section>
<section class="level1">
  <h1 id="level-1">Level 1</h1>
  <section class="level2">
    <h2 id="level-2">Level 2</h2>
    <section class="level3">
      <h3 id="level-3">Level 3</h3>
    </section>
  </section>
  <p>Level 2 was ended by <code>##</code>.</p>
  <h2 class="just-a-heading" id="not-sectionize">Not Sectionize</h2>
  <blockquote>
    <h1 id="not-sectionize-in-blockquote">Not Sectionize in Blockquote</h1>
  </blockquote>
</section>
```

CSS

```
body > section {
}

section:has(> #intro) {
}

section:has(> h1.title) {
}

.level1 {
}
.level2 {
}

blockquote > h1 {
}
```

Hooks

Replace

```
[
icon1
][notice]

[
person
][nod nod]
```

```
const rules = [
  {
    test: /\[(.+?)\]\[(.+?)\]/,
    match: ([, a, b], h) => {
      return h(
        'div',
        {class: 'balloon'},
        h('img', {src: `./img/${a}.png`}),
        h('span', b),
      );
    },
  },
];

stringify(markdown, {
  partial: true,
  replace: rules,
});
```

```
<p><div class="balloon"><img src="./img/icon1.png"><span>Notice</span></div></p>
<p><div class="balloon"><img src="./img/person.png"><span>Nod nod</span></div></p>
```