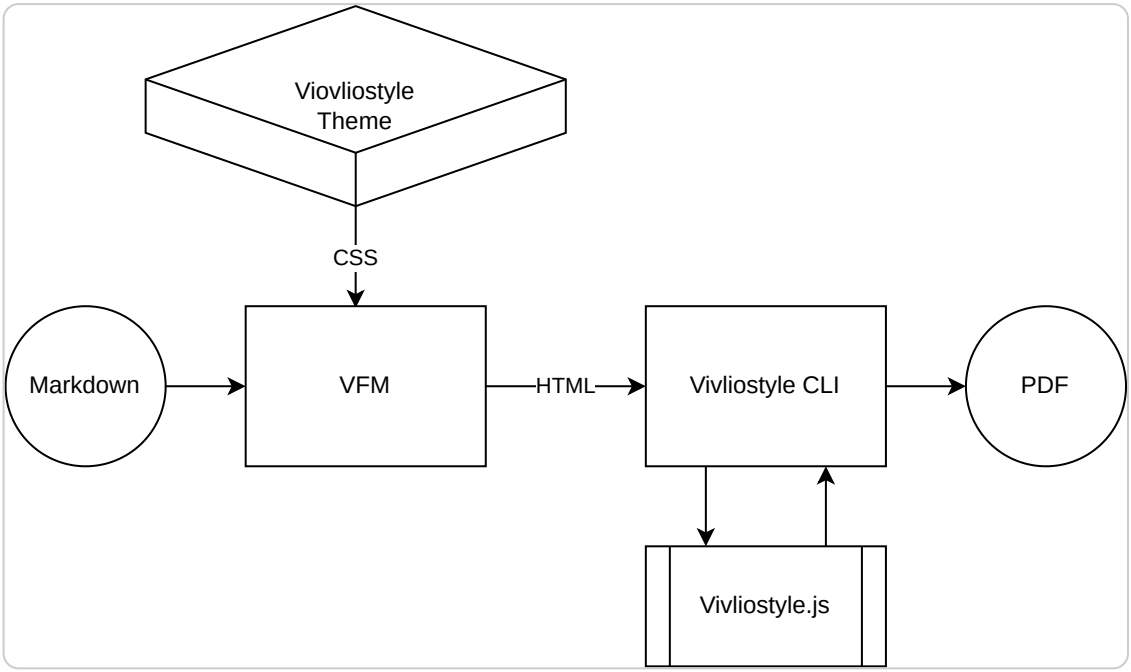
VFM (Vivliostyle Flavored Markdown) Documentation

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), a Markdown syntax optimized for book authoring. It is standardized and published for Vivliostyle and its sibling projects. VFM is implemented top on CommonMark (https://commonmark.org/) and GitHub Flavored Markdown (GFM) (https://github.github.com/gfm/).

Table of contents

VFM syntax and features are listed in ascending alphabetical (`A` - `Z`) order.

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 uses Prism (https://prismjs.com/) for syntax highlighting.

with caption

VFM

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

or

````
```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 {
}
```

Footnotes

Define a footnotes, like Pandoc (https://pandoc.org/MANUAL.html#footnotes).

VFM

```
VFM is developed in the GitHub repository[^1].
Issues are managed on GitHub[^Issues].
Footnotes can also be written inline^[This part is a footnote.].

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

[^Issues]: [Issues](https://github.com/vivliostyle/vfm/issues)
```

HTML

```
<p>
  VFM is developed in the GitHub repository <a id="fnref1" href="#fn1" class="footnote-ref
  Issues are managed on GitHub <a id="fnref2" href="#fn2" class="footnote-ref" role="doc-n
  Footnotes can also be written inline <a id="fnref3" href="#fn3" class="footnote-ref" rol
</p>
<section class="footnotes" role="doc-endnotes">
  <hr>
  <ol>
    <li id="fn1" role="doc-endnote"><a href="https://github.com/vivliostyle/vfm">VFM</a><a
    <li id="fn2" role="doc-endnote"><a href="https://github.com/vivliostyle/vfm/issues">Is
    <li id="fn3" role="doc-endnote">This part is a footnote.<a href="#fnref3" class="footn
  </ol>
</section>
```

CSS

```
.footnotes {
}
```

Frontmatter

Frontmatter is a way of defining metadata in Markdown (file) units. Write YAML at the beginning of the file.

I'm using js-yaml (https://www.npmjs.com/package/js-yaml) for parse in YAML. Schema is JSON_SCHEMA (https://yaml.org/spec/1.2/spec.html#id2803231).

The js-yaml parse makes `key:` to `key: null`. However, VFM treats this as an empty string. If `key:` or `key:""` is specified as the property of the attribute value, it is output as `key=""`.

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'
---

Text
```

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>Text</p>
  </body>
</html>
```

CSS

```
.my-class {
}

.foo.bar {
}
```

Defined properties

Property	Type	Description
`id`	`String`	`<html id="...">`
`lang`	`String`	`<html lang="...">`
`dir`	`String`	`<html dir="...">`, value is `ltr`, `rtl` or `auto`.
`class`	`String`	`<html class="...">` and `<body class="...">`
`title`	`String`	`<title>...</title>`, if missing, very first heading of the content will be treated as title.
`html`	`Object`	`<html key="value">`, key/value pair becomes attribute of `<html>`.
`body`	`Object`	`<body key="value">`, key/value pair becomes attribute of `<body>`.
`base`	`Object`	`<base key="value">`, key/value pair becomes attribute of `<base>`.
`meta`	`Object[]`	`<meta key="value">`, key/value pair becomes attribute of `<meta>`.
`link`	`Object[]`	`<link key="value">`, key/value pair becomes attribute of `<link>`.
`script`	`Object[]`	`<script key="value">`, key/value pair becomes attribute of `<script>`.
`vfm`	`Object`	VFM settings.
`head`	-	Reserved for future use.
`style`	-	Reserved for future use.
Other	`String`	`<meta name="key" content="value">`, key/value pair becomes one `<meta>`.

vfm

Property	Type	Default	Description
`math`	`Boolean`	`true`	Enable math syntax.
`partial`	`Boolean`	`false`	Output markdown fragments.
`hardLineBreaks`	`Boolean`	`false`	Add ` ` at the position of hard line breaks, without needing spaces.
`disableFormatHtml`	`Boolean`	`false`	Disable automatic HTML format.
`theme`	`String`	-	Vivliostyle theme package or bare CSS file.

Priority with options

If there are multiple specifications for the same purpose, the priority is as follows.

1. Frontmatter
2. VFM options

In Frontmatter, if there is a duplicate of the root `id` and `id` in the `html` property, the root definition takes precedence.

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

In this example, `sample1` is adopted.

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

Merge class properties

The `class` properties of the root, `html`, and `body` are combined separated by spaces.

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

In this example, merged.

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

Hard new line

- A newline puts `<br/>` to the end of a line.
- Consecutive 2 newlines creates a new sentence block.

This feature is optional. The Node.js API is enabled by specifying `hardLineBreaks: true` as an option and the CLI by specifying `--hard-line-breaks`.

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 {
}
```

Image

VFM

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

HTML

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

CSS

```
img {
}
```

with caption and single line

Wraps an image written as a single line and with a caption in `<figure>`.

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 {
}
```

Math equation

Outputs HTML processed by MathJax (https://www.mathjax.org/).

It is Enabled by default. To disable it, specify the following.

- `stringify` API options: `math: false`
- `VFM` API options: `math: false`
- CLI options: `--disable-math`
- Frontmatter: `math: false` of `vfm:` property
  - refs: Frontmatter
  - It takes precedence over `stringify`, but `VFM` does not.

The VFM syntax for MathJax inline is `$...$` and the display is `$$...$$`.

It also supports multiple lines, such as `$x = y\n1 + 1 = 2$` and `$$\nx = y\n$$`. However, if there is a blank line `\n\n` such as `$x = y\n\n1 + 1 = 2$` , the paragraphs will be separated and it will not be a mathematical syntax.

OK:

- `$...$`, `$$...$$` ...Range specification matches
- `$...\n...$`, `$$\n...\n$$` ...Within the same paragraph
- `$...\$...$`, `$...\$...\\\$..$`, `$$...\$...\\\$...$$` ...Escape `$` by odd `\`

NG:

- `$...$$`, `$$...$` ...Range specification does not match
- `$...\n\n...$`, `$$...\n\n...$$` ...Separated into paragraphs by line breaks
- `$ ...$` ...Spaces (space, tab, new line, ...etc), immediately after `$` at start of inline
- `$... $` ...Spaces (space, tab, new line, ...etc), immediately before `$` at end of inline
- `$...$5` ...Digit `0...N` immediately after `$` at end of inline

VFM

```
inline:$x = y$

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

HTML

It also outputs `<script>` for processing MathJax if `math` is enabled and the math syntax or the `<math>` tag is actually existed.

```
<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 {
}
```

Raw HTML

VFM

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

HTML

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

with Markdown

VFM

```
<div class="custom">

# Heading

</div>
```

HTML

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

Ruby

VFM

```
This is {Ruby|ルビ}
```

HTML

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

CSS

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

Escape pipe in ruby body

If want to escape the delimiter pipe `|`, add `\` immediately before it.

VFM

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

HTML

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

Sectionization

Make the heading a hierarchical section.

- Do not sectionize if the heading line starts with `#`s and ends with equal or greater number of `#`s.
  - `### Not Sectionize ###` (enclosed by equal number of `#`s) -- not sectionize
  - `### Sectionize ##` (insufficient number of closing `#`s) -- sectionize
- A line with only `#`s can be used to end the section whose depth matches the number of the `#`s.
  - e.g., the section starting with `### Heading 3` can end with `###`.
- Do not sectionize if parent is `blockquote`.
- Set the `levelN` class in the section to match the heading depth.

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>
```