FigureMark

This is the documentation for **FigureMark**, a trivial syntax for marking up *figures* in Markdown documents (i.e. HTML `<figure>` blocks) for use in digital and printed books, or online. It was created by Matt Gemmell.

FigureMark is intended to be useful in the creation of non-fiction books, particularly those on technical matters, such as learning a programming language, discussing complex concepts which benefit from detailed examples, and so on. Having said that, its focus is on simplicity, and meeting a set of common needs without complexity at the expense of comprehensiveness.

Features requests, bug reports, and general discussion are welcomed; you can find my contact details here.

Introduction

FigureMark is extremely simple in functionality, easy to understand, and hopefully not too burdensome to use. Let's begin with an example.

Here's the source of a given figure, using the FigureMark syntax:

```
```figuremark Demo of FigureMark {.example #demo}
normal text
[inserted text]{+}
[removed text]{-} [// comment]{/}
text with a reference {1}
[result of something]{>}
text with reference {2.1} and a [highlight]{!}
```
```

And the result, after processing and with suitable CSS:

Demo of FigureMark **Fig. 1**

```
normal text
inserted text
removed text // comment
text with a reference 1

> result of something

text with reference 2.1 and a highlight
```

There are several notable features of the syntax, as described below.

Figure block delimiters and attributes

Figures begin and end with lines which start with at least 3 ` (backtick) symbols followed by `figuremark`. You can use ~ tildes instead of backticks if you wish. The closing line should have no other content. The opening line may also have, in order:

1. A title, which if provided will become the figure's caption.
2. An *attributes block*, inside `{braces}`. This can contain any number and combination of:
   - An `#id` for the figure tag, e.g. `#chp1-example1`
   - CSS `.classes` for the figure tag, e.g. `.example`
   - `Key=value` pairs for the figure tag, e.g. `data-foo="baz"`

The title and attributes block may have whitespace between or around them. Key-value pairs may have the values in single-quotes, double-quotes, or neither, but all values will be double-quoted in the resulting HTML.

The figure's caption will automatically gain a `figure-number` span containing `Fig. 1` or such, which can readily be hidden or moved via CSS. The figure's number is automatically calculated on a global basis, and takes into account both FigureMark figures and *also* any pre-existing HTML `<figure>` blocks. If a title was specified for the figure, it will be within a `figure-title` span inside the caption.

Figures can be linked to via their `id` attribute, which will be of the form `figure-1` unless overridden in the attributes block as described above. If multiple IDs are specified in the attributes block, only the final one will be used.

The resulting figure tag will also automatically gain a `data-fignum` attribute, containing the figure number. The figure's entire content, *excluding* any caption, will be within a `div` with the `figure-content` class applied.

Escaping

Since the FigureMark marking syntax (described below) makes use of square brackets and curly braces, it may be necessary to escape those characters in the figure's content, if you do not wish them to be interpreted as marks.

This can be achieved by prefixing any literal brackets and braces with a backslash (like this: `\[`). Any such backslashes will be removed when the block is processed. If you wish to have a literal backslash before a bracket or brace, use two backslashes instead of one; only one will remain after processing.

Escaping these characters is especially important when annotating FigureMark syntax examples using FigureMark.

Mark types

There are 7 types of marks available to decorate content, reflecting commonly-needed kinds of annotations when illustrating material for discursive or educational purposes. They are split into three groups, detailed below.

All marks are transformed into `<span>` tags, each with their own classes as described. Additionally, every mark will have the `figuremark` class. The exact ren-

dering of the marks is of course determined by the CSS applied to the document.

1. Reference marks

Reference marks are simply numerical call-outs, inserted on their own so that the text can refer to them. The syntax is `{1}` for example, with full-stops/periods/dots and dashes/hyphens also permitted. The resulting spans will have the `reference` class, and *also* a class of the form `reference-1` etc.

References are the only standalone mark, with all of the remaining marks spanning some content.

2. Spanning short marks

Short marks are single-character marks, which decorate the spanned section of content within a line. There are five predefined short marks, as detailed below, with their syntax all following the same form: `[content to annotate]{type}`. Any occurrences of square brackets or curly braces within the bracketed content can be escaped with backslashes.

The defined types are:

- `+` mapped to the CSS class `insert`. Intended for added, inserted, or approved content.
- `-` mapped to the CSS class `remove`. Intended for removed, replaced, or disapproved content.
- `!` mapped to the CSS class `highlight`. Intended to draw attention to content.
- `>` mapped to the CSS class `result`. Intended to show a result or outcome, for example of code execution.
- `/` mapped to the CSS class `comment`. Intended for remarks or other asides, including source code comments.

Styling of these is up to your CSS, using the mapped classes mentioned above.

3. Class marks

Similar to the second group, the final type of mark is a class mark, whose annotation is simply interpreted as a list of CSS classes to apply to the resulting span. For example, the syntax `[some important text]{important}` will result in a FigureMark span with the `important` CSS class applied.

One or more classes can be specified, with or without the leading `.` dots in each case, and separated by whitespace. If dots are supplied, whitespace can be omitted; thus, the following are all equivalent: `[text]{one two}` and `[text]{.one .two}` and `[text]{.one.two}`.

Questions

The following questions are anticipated, and answers provided.

Is there a reference implementation of a FigureMark parser?

There is a Python implementation here. As a trivial markup format, implementation should be very straightforward in any language supporting regular expressions.

How can I annotate an example of FigureMark syntax using FigureMark?

To avoid ambiguity for the parser, follow these two rules:

1. The outer FigureMark block should either use a different number of backticks/tildes for its opening and closing lines than the inner example block does, or the outer and inner blocks should use different delimiters (for example, use tildes on the outer block and backticks on the inner).
2. Within the example block, escape each square bracket and curly brace with a backslash if they would otherwise be interpreted as FigureMark syntax. Any such backslashes will be removed when the block is processed.

This will allow FigureMark samples to be annotated, as shown below.

FigureMark Syntax **Fig. 2**

```
```figuremark Example of FigureMark {.example}
Some [marked up]{!} text. {1} [// comment]{/}
```
```

How can I move the caption to below the figure, instead of above?

CSS can be used for this purpose. For example:

```
figure { display: flex; flex-direction: column; }
figcaption { order: 1; }
```

The actual result will of course depend on your other CSS.

What's the fancy divider symbol between this document's sections?

It's a Manichaean fleuron. Thank you for noticing.