Comparing Markdown Flavors

Contents

Purpose

This document intends to compare various flavors of Markdown for the purposes of use in the MetaBrainz projects, and ultimately recommend one of them.

> ***Note***
>
> “Notable” differences means non-edge case differences throughout this document, even if they are small.

This takes into account how commonly known the flavor is, how hard it is to learn, how useful the additions or changes it provides are to MetaBrainz, implementations available [1], and other relevant factors.

Flavors compared

- Gruber Markdown (the original specification)
- CommonMark
- Github Flavored Markdown (GFM)
- Markdown Extra
- CriticMarkup
- MultiMarkdown (MMD)

Gruber Markdown

Specification: Gruber Markdown

Implementations

- python-markdown: does what it says on the tin - only notable difference is no intraword emphasis for _.
- Markdown.pl: the original Perl implementation [2]
- Text::Markdown: a somewhat newer CPAN module
- markdown-js: a JS markdown parser that uses Gruber by default.

CommonMark

Specification: CommonMark

Implementations

- CommonMark.js: official JS reference implementation
- remarkable: configurable and extendable JS CommonMark parser
- CommonMark-py: port of the JS implementation to Python

- paka.cmark: python FFI bindings to the commonmark C library
- perl-commonmark: perl bindings to the commonmark C library

Description

CommonMark is the result of an effort to make a “standard, unambiguous syntax specification for Markdown, along with a suite of comprehensive tests to validate Markdown implementations against this specification” [3]. It’s comprised of various large-scale Markdown users including representatives from GitHub, Stack Exchange, and Reddit. As such, it’s one of the most used standards of Markdown on the web [4].

Differences

The differences from Gruber Markdown are listed here. Notable among them include the addition of fenced code blocks with triple backticks or ~~~, making a new list when a different type of bullet is used:

```
* hello
* wow
+ new
+ list
1. this is a new
2. one too
```

being able to use both *1)* and *1.* when writing an ordered list, and intraword emphasis being disabled for _.

Uses in MetaBrainz

CommonMark is a large enough flavor that it includes most everything that’s needed for MetaBrainz, and a small enough flavor that it’s still usable in other places like when converting with Pandoc. CommonMark is already used in the discussion forums and is a subset of Github Flavored Markdown.

It doesn’t, however, include tables, which may be useful in MusicBrainz edit notes, as a way to (say for example) clearly document separately the sources for each change:

```
| Change                | Source               |
| --------------------- | -------------------- |
| Changed Gender to Male | Some source          |
| Added Release          | Some source          |
| Did another thing      | No definitive source |
```

With a UI rendering these could be very pretty and convenient. Additionally, tables are already supported in the discussion forums, so it’s also good to maintain consistency.

Github Flavored Markdown

Specification: GFM Spec

Implementations

- marked: JS markdown parser that uses GFM by default
- py-gfm: technically an extension to python-markdown

remarkable can also be worked into something similar to GFM.

Description

GFM is a superset of CommonMark that adds a few additional syntax features, mostly Github-specific ones. [5] It's used pretty much everywhere that text is typed in Github, but not anywhere else.

Differences

Differences from CommonMark are highlighted in the GFM Spec. Notable ones include the addition of tables, task list items with the *[ ] item* syntax, and *~~strike-through~~*. It adds the ability for links to be automatically recognized without any special syntax surrounding them (like the *<rawlink.com>* syntax in Gruber & CommonMark), and finally disallows certain HTML tags like *<iframe>*, *<noscript>*, and *<style>*.

Uses in MetaBrainz

Since GFM is a superset of CommonMark, it has all the same advantages that it does, plus tables and check lists. Tables would be helpful to a certain extent in MusicBrainz edit notes, but checklists are nearly useless.

Markdown Extra

Specification: Markdown Extra (only specifies differences from Gruber Markdown)

Implementations

- python-extra: a collection of python-markdown plug-ins that behave like Markdown Extra.
- js-markdown-extra: port of the PHP verasion of Markdown Extra.

Descrption

Markdown Extra is, as the name suggests, intended to be an extension to Gruber Markdown that implements a certain extra set of features.

Differences

The Markdown Extra spec only lists differences from Gruber Markdown. Essentially the same as CommonMark with differences in code block syntax and the addition of tables, definition lists, footnotes, and abbreviations.

Uses in MetaBrainz

Markdown Extra is somewhat less well known but still carries the advantages of CommonMark. Other than tables, it doesn't provide a significant feature that MetaBrainz can use.

CriticMarkup

Specfication: CriticMarkup (only specifies differences from Gruber Markdown)

Implementations

I could not find any independent CriticMarkup implementations. However, the CriticMarkup github has files that can easily be massaged into proper Python implementations. remarkable also has similar features.

Description

CriticMarkup is “intended to provide basic editorial change tracking in plain text files”. It adds syntax to Gruber Markdown that can be parsed into HTML to be used when reviewing, or read as plaintext easily.

Differences

See the CriticMarkup spec. Essentially, it adds markers to text for additions, deletions, substitutions, reviewer comments, and highlighting.

Uses in MetaBrainz

It may be a useful idea to use the CriticMarkup features in MusicBrainz when suggested changes to an edit. This way, the differences can be seen in plaintext. remarkable can achieve this with extensions or even the built-in ++*inserted text*++ and ==*marked text*== syntax extensions, given the otherwise lack of libraries. [6]

MultiMarkdown

Specification: MultiMarkdown Guide

Implementations

- Text::MultiMarkdown: CPAN module
- pymmd: Python wrappers for MMD 5. [7]

Description

MMD is a powerful superset of Gruber Markdown, adding features like internal cross referencing, footnotes, math rendering, the ability to add a glossary, etc.

Differences

See the MultiMarkdown Guide, while it’s not a spec, it’s very comprehensive. Nearly all the changes are notable.

Uses in MusicBrainz

While all the features present in MultiMarkdown are useful, they are only so to a very limited extent and are not as ubiquitously known as the alternatives on this list. The confusing version system [8] adds to the pain, as does the fact that there are no solid implementations.

Comparison Summary

Feature	CommonMark	GFM	Markdown Extra	CriticMarkup	MMD
Headers	✓	✓	✓	✓	✓
Lists	✓	✓	✓	✓	✓
Nested lists	✓	✓	✓	✓	✓
Linking	✓	✓	✓	✓	✓
Image linking	✓	✓	✓	✓	✓
Blockquotes	✓	✓	✓	✓	✓
Reference links [9]	✓	✓	✓	✓	✓
Inter-word emphasis [10]					
Inline code	✓	✓	✓	✓	✓
Indented code blocks	✓	✓	✓	✓	✓
Fenced code blocks	✓	✓	✓		✓
Syntax highlighting		✓	✓		✓
Tables		✓	✓		✓
Footnotes			✓		✓
Auto-linking		✓			
HTML sanitization		✓			
Strike-through		✓			
Definition Lists			✓		✓
Check Lists		✓			
Citations					✓
Math rendering					✓
Glossaries					✓
Editorial syntax [11]				✓	✓
Abbreviations			✓		✓
JavaScript impl.	✓	✓	✓		
Python impl.	✓	✓	✓		✓
Perl impl.	✓				✓

Notes		Could use remarkable to support linking to MusicBrainz edits		CriticMarkup github provides workable base for a Python impl	

Conclusion

Given the needs of MusicBrainz, I think a variation on **Github Flavored Markdown** without github-specific items is the best solution. Instead of giving the ability to link to issues/PRs, we could replace those with linking to specific edits on MusicBrainz, revisions on BookBrainz, or whatever works for the particular project. remarkable supports extensions (with remarkably) so this is easily implementable.

The combination of how well-known it is, plus the fact that it's not very that different from the CommonMark standard (which looks like it's going to be by far the most well-supported and well-known standard for the foreseeable future) plus the fact that it includes tables, makes it a near-perfect choice.

1 in Python, JavaScript, and Perl.

2 On a personal note, I read through the source of this. I was simultaneously horrified and satisfied.

3 It's near-obligatory to mention 927 here but it seems to have worked rather better in this case.

4 Despite this it would be amiss not to mention that sites still diverge on certain additional features, like spoilers, syntax highlighting (Github only highlights when the language is explicitly mentioned, but StackOverflow uses question tags to tell what to highlight). Even spec features like fenced code blocks differ (Stack Exchange doesn't support it altogether).

5 these refer to linking to issues and pull requests using *#number* and referencing a particular commit using its long SHA-1 hash.

6 You could say that the lack of CriticMarkup implementations is… remarkable.

7 which is technically an old version, the current one is MMD 6.

8 MMD has gone through multiple rewrites:

```
MultiMarkdown v3 (aka 'peg-multimarkdown') was based
on John MacFarlane's peg-markdown. It used a parsing
expression grammar (PEG), and was written in C in
order to compile on almost any operating system.
Thanks to work by Daniel Jalkut, MMD v3 was built so
that it didn't have any external library requirements.

MultiMarkdown v4 was basically a complete rewrite
of v3. It used the same basic PEG for parsing
(Multi)Markdown text, but otherwise was almost
completely rebuilt.

MultiMarkdown v5 was largely the same codebase as v4,
but the build system was restructured to use CMake.

MultiMarkdown v6 is the biggest rewrite since v3. The
parser was completely rewritten to improve accuracy
and (most importantly) performance. v6 includes
multiple new features, reimagines a couple of existing
features, and deprecates one or two old syntax
structures.
```

9 this refers to the ability to do this:

```
I get 10 times more traffic from [Google] [1] than from
[Yahoo] [2] or [MSN] [3].

  [1]: http://google.com/        "Google"
  [2]: http://search.yahoo.com/  "Yahoo Search"
  [3]: http://search.msn.com/    "MSN Search"
```

10 this refers to whether the 'hello' in *word_hello_world* is italicized.

11 examples include the addition syntax with *{++ ++}*, the substitution syntax with *{~~ x ~> y ~~}*, commenting with *{>> <<}*, etc.