MultiMarkdown User's Guide
Version 2.0.b6

Fletcher T. Penney
http://fletcherpenney.net/

November 9, 2010

Formatted for LaTeX
by MultiMarkdown

Contents

Chapter 1

Introduction to MultiMarkdown

This document is an introduction to MultiMarkdown[1] — what it is, how to use it, how you can help make it better. This document exists in multiple formats: a plain text document, a pdf, a Scrivener[2] document, etc. Find the format that best suits your needs, or create your own. That is what MultiMarkdown was designed to be used for!

1.1 What is Markdown?

To understand what MultiMarkdown is, you first should be familiar with Markdown[3]. The best description of what Markdown is comes from John Gruber's Markdown web site:

> Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).
>
> Thus, "Markdown" is two things: (1) a plain text formatting syntax; and (2) a software tool, written in Perl, that converts the plain text formatting to HTML. See the Syntax page for details pertaining to Markdown's formatting syntax. You can try it out, right now, using the online Dingus.
>
> The overriding design goal for Markdown's formatting syntax is to make it as readable as possible. The idea is that a Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions. While Markdown's syntax has been influenced by several existing text-to-HTML filters, the single biggest source of inspiration for Markdown's syntax is the format of plain text email. [1]

1.2 What is MultiMarkdown?

Markdown is great, but it lacked a few features that would allow it to work with documents, rather than just pieces of a web page.

I wrote MultiMarkdown in order to leverage Markdown's syntax, but to extend it to work with complete documents that could ultimately be converted from text into other formats,

[1] http://fletcherpenney.net/multimarkdown
[2] http://www.literatureandlatte.com/scrivener.html
[3] http://daringfireball.net/projects/markdown/

including complete XHTML documents, LaTeX, PDF, RTF, or even (shudder) Microsoft Word documents.

In addition to the ability to work with complete documents and conversion to other formats, the Markdown syntax was lacking a few things. Michel Fortin added a few additional syntax tools when writing PHP Markdown Extra[4]. Some of his ideas were implemented and expanded on in MultiMarkdown.

John Gruber may disagree with me, but I really did try to stick with his proclaimed vision whenever I added a new syntax format to MultiMarkdown. The quality that attracted me to Markdown the most was its clean format. Reading a plain text document written in Markdown is *easy*. It makes sense, and it looks like it was designed for people, not computers. To the extent possible, I tried to keep this same concept in mind when working on MultiMarkdown.

I may or may not have succeeded in this...

In the vein of Markdown's multiple definitions, you can think of MultiMarkdown as:

1. A perl script to convert plain text to XHTML
2. The syntax used in the plain text to describe how to convert it to XHTML
3. The system of programs (perl scripts, shell commands, XSLT transforms, php scripts, etc) used to convert plain text to XHTML, and then to convert XHTML into LaTeX, PDF, RTF, etc)

1.3 How do I use MultiMarkdown?

You can use MultiMarkdown in a variety of ways:

- As a command-line perl program (the "default" approach)
- As a drag and drop application for Mac OS X
- As a TextMate[5]bundle[6]
- Within the Scrivener[7] application
- In a blosxom[8], Movable Type[9], Oddmuse[10], or other web site

1.4 Where can I find MultiMarkdown?

The MultiMarkdown package can be downloaded:

- `http://fletcher.github.com/MultiMarkdown/`

Information about MultiMarkdown is available on my web site:

[4]http://www.michelf.com/projects/php-markdown/extra/
[5]http://macromates.com/
[6]http://fletcher.github.com/markdown.tmbundle/
[7]http://www.literatureandlatte.com/scrivener.html
[8]http://blosxom.sourceforge.net
[9]http://www.movabletype.org/
[10]http://www.oddmuse.org/

- `http://fletcherpenney.net/multimarkdown/`

John Gruber's original Markdown is available at his site:

- `http://daringfireball.net/projects/markdown/`

Michel Fortin's PHP version of Markdown is at his site:

- `http://michelf.com/projects/php-markdown/`

1.5 Where can I get more information about MultiMarkdown?

As above, check my web site.

Also, you can check out the MultiMarkdown discussion list:

- `http://groups.google.com/group/multimarkdown/`
- multimarkdown@googlegroups.com

If you questions are specific to Scrivener, you can also browse the Literate and Latte forum:

- `http://www.literatureandlatte.com/forum/`

Chapter 2

Quickstart Guide to MultiMarkdown

Quick start instructions, for those in a hurry...

2.1 General Instructions

1. Download the MultiMarkdown package:

   `http://fletcher.github.com/MultiMarkdown/`

2. Unzip/untar it

3. MultiMarkdown can be run from anywhere, but is easiest when installed in a "common" location:

   - Windows:
     - C:\Documents and Settings\All Users\MultiMarkdown
     - C:\Documents and Settings\<user>\MultiMarkdown
   - Mac OS X or *nix
     - ~/Library/Application Support/MultiMarkdown (preferred on Mac OS X)
     - ~/.multimarkdown
     - /Library/Application Support/MultiMarkdown (preferred on Mac OS X)
     - /usr/share/multimarkdown

4. In the "bin" directory, there are a couple of perl scripts designed to take a MultiMarkdown text file and convert to XHTML, or LaTeX, or pdf. These scripts are designed to be able to be run from anywhere. You can leave them where they are, or install them somewhere in your path directory:

   - mmd2XHTML.pl
   - mmd2LaTeX.pl
   - mmd2PDF.pl
   - mmd2PDFXeLaTeX.pl
   - mmd2letter.pl

5. To use these files, do something like the following:

```
cd MultiMarkdown
bin/mmd2XHTML.pl file.txt
```

where "file.txt" is the MultiMarkdown file you wish to process. "file.html" will be created automatically

6. You can now open `file.html` in your web browser, or do what you like with it.

Chapter 3

MultiMarkdown Syntax Guide

This is a guide to the markup syntax used in the MultiMarkdown system.

3.1 Metadata

MultiMarkdown has support for metadata, meaning that you can include information about a document that is not necessarily part of the document contents.

To use metadata, simply add information to the top of a Markdown file:

```
Title:  A New MultiMarkdown Document
Author: Fletcher T. Penney
        John Doe
Date:   July 25, 2005
```

The key is the text before the colon, and the data is the text after the colon. In the above example, notice that there are two lines of information for the Author key. If you end a line with "space-space-newline", the newline will be included when converted to other formats.

There must not be any whitespace above the metadata, and the metadata block ends with the first whitespace only line. The metadata is stripped from the document before it is passed on to the syntax parser.

While not required, I recommend including two spaces at the end of each line of metadata. In this way, if you pass your document through a regular version of Markdown, the metadata will be properly formatted as plain text with line breaks, rather than joined into a single run-on paragraph.

I have included information about some of the "standard" metadata keys — I welcome feedback and suggestions for additional standard keys that would be useful. If you add keys that are not listed, they are included in the XHTML and LaTeX as custom variables that can still be used if you desire.

Remember, XHTML snippets have no means to use metadata. To make use of these features, one must be using `Format:  complete` to create full XHTML documents, which can then be processed using XSLT to create other document types. As an example, I use metadata for information that is used to add title, author, keyword, and copyright metadata to PDF's created by MultiMarkdown.

Note: I make multiple mentions to the use of these keys for LaTeX documents. This is simply because the LaTeX output format currently makes the most use of the metadata

information. Any export format could be modified to make use of additional metadata keys.

Address

Use this to include the author's mailing address. You can have more than one line in this field — use two extra spaces at the end of a line, and a newline character will be used in LaTeX. Also used as return address for letterhead and envelope templates.

Author

Self-explanatory. I strip this out to provide an author string to LaTeX documents. Also used as the sender for letterhead and envelope templates.

Affiliation

Use this to include an organization that the author is affiliated with, e.g. a university, company, or organization. You can include address information here as well, or use the `Address`, `email`, `web`, and `phone` metadata fields. You can have more than one line in this field — use two extra spaces at the end of the line, and a newline character will be used in LaTeX.

Base Header Level

Used by my XSLT script tool to change the default header level. For example, if using the memoir class, you might want a first level header to be interpreted as a chapter, rather than as a part. To do this, simply set `Base Header Level` to `2`.

Base URL (Deprecated)

Deprecated - WikiWords and WikiLinks no longer supported.

Bibliography Title

Change the title used for the references section (e.g. "References" or "Bibliography"). The default value is "Bibliography".

Bibliography Style

The name of the BibTeX style you wish to use.

BibTeX

This should be the name of a `.bib` file (a BibTeX file used to store references). If you use my xhtml2latex.xslt file, this will convert external citations into markup for BibTeX (see Bibliography Support (section 3.8) for more information).

You must have `bibtex` installed and working, and the `.bib` file must be in your working directory.

Chapterstyle

This is used to designate the `chapterstyle` in LaTeX memoir documents.

Copyright

This can be used to provide a copyright string.

CSS

Used to specify a CSS stylesheet when creating the complete XHTML output.

Date

Provide a date for the document.

Email

Use this to include the author's email address.

Format

Set to `complete` to indicate that a fully-formed XHTML document should be produced. Such a document is ready for processing by an XSLT tool, such as the XSLT files to convert XHTML into LaTeX.

Set to `snippet` to indicate that no <head> or other information should be included. This might be useful for generating (X)HTML output ready for pasting into a weblog, for example.

Note: Some MultiMarkdown tools add this for you (e.g. TextMate using my bundle, and Scrivener.) Duplicating the `Format` key in these programs should not cause a problem — let me know if you have trouble.

Keywords

Provide a list of keywords for the document. I use these to add keywords to PDF's that are produced as well. Keywords can be separated by commas, or placed on separate lines.

Language

Currently, the language field is used to specify which version of SmartyPants[1] to use. In the future, it may be used for other purposes as well.

The languages are written using the English word (e.g. "german" not "deutsch").

LaTeX XSLT

Used to designate an XSLT file to convert an XHTML document to a LaTeX document. The LaTeX document can then be converted to PDF by `pdflatex`. This key used to be called `XSLT File`.

Pagestyle

This is used to designate the `pagestyle` in LaTeX memoir documents.

[1] http://daringfireball.net/projects/smartypants/

Phone

Use this to include the author's phone number(s). You can have more than one line in this field — use two extra spaces at the end of the line, and a newline character will be used in LaTeX.

Recipient

Used by letterhead and envelope templates.

Recipient Address

Used by letterhead and envelope templates.

Revision

You can use a string to declare the current version of the document. Displayed on the copyright page when using my memoir XSLT transform.

RTF XSLT

This key is used to provide an XSLT file that can alter the XHTML output prior to conversion to RTF. Useful for further customizing the output of MultiMarkdown specifically for the RTF format. I have no plans to create any such files myself, but others may find it useful.

I strongly encourage you to use another route to convert XHTML to RTF. I've had the best results with Google Docs[2]*. For non-Mac users, that's definitely the way to go.*

Subtitle

Used to provide a subtitle. It ends up in the meta tags, but can be extracted by XSLT for other uses.

Title

Used to provide the official title of a document. This is set as the <`title`> string within the <`head`> section of an HTML document, and is also used by other export formats.

Use WikiLinks (Deprecated)

Set to `true` or `1` to enable the use of `WikiWords` and `[[Free Links]]`. Requires that you also set `Base URL`. See WikiLinks (Deprecated) (section 3.5) for more information.

Web

Use this to include the author's web URL.

[2] http://docs.google.com/

XHTML Header

This is used to include raw XHTML information in the header of a document. You can use this field to add information that will be included in the header of the generated XHTML file. This can be CSS formatting data, or javascript code, or just about anything. I am not responsible for getting that code to work. MultiMarkdown just includes it as is.

Anything included in this field is inserted, unaltered, in the `<head>` section of the XHTML output. If you do add anything here, the XSLT stylesheet may have to updated to ignore what you added if you want to convert to LaTeX. Let me know what you add, and I can consider updating the XSLT stylesheet to ignore it. Currently it ignores `<style>` sections.

XHTML XSLT

This is the name of the XSLT file to use to post-process the XHTML file. This can be used to further customize the XHTML output generated by MultiMarkdown. For example, the `xhtml-toc.xslt` file can add a Table of Contents to the start of XHTML page.

XMP

This is used to provide a file to be included using xmpincl[3]. Basically, this adds the ability to provide Creative Commons Licensing information in a PDF's metadata[4]. It can also be used for other purposes (beyond the scope of this document.)

XSLT File (deprecated)

This metadata key has been deprecated in favor of `XHTML XSLT`, `RTF XSLT`, and `LaTeX XSLT`.

3.2 Automatic Cross-References

An oft-requested feature was the ability to have Markdown automatically handle within-document links as easily as it handled external links. To this aim, I added the ability to interpret `[Some Text][]` as a cross-link, if a header named "Some Text" exists.

As an example, `[Metadata][]` will take you to the section describing metadata (section 3.1).

Alternatively, you can include an optional label of your choosing to help disambiguate cases where multiple headers have the same title:

```
### Overview [MultiMarkdownOverview] ##
```

This allows you to use `[MultiMarkdownOverview]` to refer to this section specifically, and not another section named `Overview`. This works with atx- or settext-style headers.

If you have already defined an anchor using the same id that is used by a header, then the defined anchor takes precedence.

In addition to headers within the document, you can provide labels for images and tables which can then be used for cross-references as well.

[3] http://www.ctan.org/tex-archive/macros/latex/contrib/xmpincl/
[4] http://wiki.creativecommons.org/XMP

3.3 Image Support

Obviously, images are handled just fine by Markdown (with the exception of attributes as noted above.) However, without some more information, images are not easily translated into other document formats (e.g. PDF).

To handle this, my XSLT files will make use of <img> dimensions (e.g. `height` and `width`). If present, the image will be scaled. If only one dimension is specified, the image will be scaled proportionately. If neither `height` nor `width` is specified, then the image will be scaled such that it's width is the same as a column of text. This is to prevent high resolution images from overflowing the page. Unfortunately, it has the side effect of "zooming" in on smaller images. So, if you have images that are being scaled in a way that you do not desire, simply specify at least one dimension.

Note: XHTML only allows for units of `px` and `%` on <img> tags. LaTeX allows for several others. So, my XSLT file allows for other units to be used, even if they screw up the XHTML version. You have to choose appropriate units for your purpose. Unfortunately, the only way around this is to make sure that all of your images contain actual dimension information, and then remove the \resizebox part from the XSLT.

3.4 Anchor and Image Attributes

Adding attributes to links and images has been requested for a long time on the Markdown discussion list. I was fairly opposed to this, as most of the proposals really disrupted the readability of the syntax. I consider myself a "Markdown purist", meaning that I took John's introduction to heart:

> The overriding design goal for Markdown's formatting syntax is to make it as readable as possible. The idea is that a Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions. While Markdown's syntax has been influenced by several existing text-to-HTML filters, the single biggest source of inspiration for Markdown's syntax is the format of plain text email.

Because there was not a syntax proposal that I felt fit this goal, I was generally opposed to the idea.

Then, Choan C. Gálvez proposed[5] a brilliantly simple syntax that stayed out of the way. By simply appending the attributes to the link reference information, which is already removed from the text itself, it doesn't disturb the readability.

For example:

```
This is a formatted ![image][] and a [link][] with attributes.

[image]: http://path.to/image "Image title" width=40px height=400px
[link]:  http://path.to/link.html "Some Link" class=external
         style="border: solid black 1px;"
```

This will generate width and height attributes for the image, and a border around the link. And while it can be argued that it does look "like it's been marked up with tags [and]

[5] http://six.pairlist.net/pipermail/markdown-discuss/2005-October/001578.html

formatting instructions", even I can't argue too strongly against it. The link and the title in quotes already look like some form of markup, and the the additional tags are hardly that intrusive, and they offer a great deal of functionality. They might even be useful in further functions (citations?).

The attributes must continue after the other link/image data, and may contain newlines, but must start at the beginning of the line. The format is `attribute=value` or `attribute="multi word value"`. Currently, MultiMarkdown does not attempt to interpret or make any use of any of these attributes. Also, you can't have a multiword attribute span a newline.

3.5 WikiLinks (Deprecated)

Note: The WikiLinks feature was more trouble than it was worth, and has been removed. One can still use the wiki software to manage these links. For example, my MultiMarkdown Extension[6] for Oddmuse[7] supports Oddmuse styled WikiLinks.

3.6 Footnotes

I have added support for footnotes to MultiMarkdown, using the syntax proposed by John Gruber. Note that there is no official support for footnotes yet, so the output format may change, but the input format sounds fairly stable.

To create a footnote, enter something like the following:

```
Here is some text containing a footnote.[^somesamplefootnote]

[^somesamplefootnote]: Here is the text of the footnote itself.

[somelink]:http://somelink.com
```

The footnote itself must be at the start of a line, just like links by reference. If you want a footnote to have multiple paragraphs, lists, etc., then the subsequent paragraphs need an extra tab preceding them. You may have to experiment to get this just right, and please let me know of any issues you find.

This is what the final result looks like:

> Here is some text containing a footnote.[8]

3.7 Tables

I have implemented a syntax for tables similar to that used by Michael Fortin's PHP Markdown Extra[9].

Basically, it allows you to turn:

[6] http://www.oddmuse.org/cgi-bin/oddmuse/Markdown_Extension
[7] http://www.oddmuse.org/
[8] Here is the text of the footnote itself.
[9] http://www.michelf.com/projects/php-markdown/extra/

```
|             |          Grouping           ||
First Header  | Second Header | Third Header |
 ------------ | :-----------: | -----------: |
Content       |          *Long Cell*        ||
Content       |   **Cell**    |         Cell |

New section   |     More      |         Data |
And more      |           And more           |
[Prototype table]
```

into a table (Table 3.1).

Table 3.1: Prototype table

	Grouping	
First Header	Second Header	Third Header
Content	*Long Cell*	
Content	**Cell**	Cell
New section	More	Data
And more	And more	

The requirements are:

- There must be at least one `|` per line
- The second line must contain only `|`,`-`,`:`,`.`, or spaces
- Cell content must be on one line only
- Columns are separated by `|`
- The first line of the table, and the alignment/divider line, must start at the beginning of the line

Other notes:

- It is optional whether you have `|`'s at the beginning and end of lines.
- To set alignment, you can use a colon to designate left or right alignment, or a colon at each end to designate center alignment, as above. If no colon is present, the default alignment of your system is selected (left in most cases). If you use a period character (`.`), then `char` alignment is used - in the future this will allow columns of decimal formatted numbers to be aligned on the decimal character. Browsers do not currently support this feature, so it is somewhat useless at the moment. It could be used in an XSLT stylesheet for other output formats (e.g. LaTeX).
- To indicate that a cell should span multiple columns, there simply add additional pipes (`|`) at the end of the cell, as shown in the example. If the cell in question is at the end of the row, then of course that means that pipes are not optional at the end of that row....

- You can use normal Markdown markup within the table cells.
- Captions are optional, but if present must be at the beginning of the line immediately preceding or following the table, start with `[`, and end with `]`. If you have a caption before and after the table, only the first match will be used.
- If you have a caption, you can also have a label, allowing you to create anchors pointing to the table. If there is no label, then the caption acts as the label
- Cells can be empty.
- You can create multiple <tbody> tags within a table by having a **single** empty line between rows of the table. This allows your CSS to place horizontal borders to emphasize different sections of the table.
- If there is no header for the first column, then cells in that column will be treated as headers, and formatted as such.

3.8 Bibliography Support

I have included support for *basic* bibliography features in this version of MultiMarkdown. Please give me feedback on ways to improve this but keep the following in mind:

1. Bibliography support in MultiMarkdown is rudimentary. The goal is to offer a basic standalone feature, that can be changed using the tool of your choice to a more robust format (e.g. BibTeX, CiteProc). My XSLT files demonstrate how to make this format compatible with BibTeX, but I am not planning on personally providing compatibility with other tools. Feel free to post your ideas and tools to the wiki.
2. Those needing more detailed function sets for their bibliographies may need customized tools to provide those services. This is a basic tool that should work for most people. Reference librarians will probably not be satisfied however.

To use citations in MultiMarkdown, you use a syntax much like that for anchors:

```
This is a statement that should be attributed to
its source[p. 23][#Doe:2006].

And following is the description of the reference to be
used in the bibliography.

[#Doe:2006]: John Doe. *Some Big Fancy Book*.  Vanity Press, 2006.
```

The XHTML that is generated is as follows:

```
<p>This is a statement that should be attributed to its source
<span class="markdowncitation"> (<a href="#Doe:2006">1</a>, <span
class="locator">p. 23</span>)</span>.</p>

<p>And following is the description of the reference to be used
```

```
in the bibliography.</p>

<div class="bibliography">
<hr />
<p>Bibliography</p>

<div id="Doe:2006"><p>[1] John Doe. <em>Some Big Fancy Book</em>.
Vanity Press, 2006.</p></div>

</div>
```

You are not required to use a locator (e.g. p. 23), and there are no special rules on what can be used as a locator if you choose to use one. If you prefer to omit the locator, just use an empty set of square brackets before the citation:

```
This is a statement that should be attributed to its
source[][#Doe:2006].
```

There are no rules on the citation key format that you use (e.g. `Doe:2006`), but it must be preceded by a `#`, just like footnotes use `^`.

As for the reference description, you can use Markup code within this section, and I recommend leaving a blank line afterwards to prevent concatenation of several references. Note that there is no way to reformat these references in different bibliography styles; for this you need a program designed for that purpose (e.g. BibTeX).

If you want to include a source in your bibliography that was not cited, you may use the following:

```
[Not cited][#citekey]
```

The `Not cited` bit is not case sensitive.

MultiMarkdown References

If you define your references (as in the example above), MultiMarkdown will automatically append a basic bibliography to the end of your document. The citations will of the form:

```
<span class="markdowncitation"> (<a href="#citekey">#
</a>, <span class="locator">p. 23</span>)</span>
```

If you don't define a locator, you will get:

```
<span class="markdowncitation"> (<a href="#citekey">#
</a>)</span>
```

When you click on the `#` (which is replaced with the specific reference number), it takes you to the appropriate point in the Bibliography. Unlike footnotes, there is no reverse link.

External References

If you do not define references, then MultiMarkdown will substitute different markup that can be used by XSLT to transform it into markup for an external tool, e.g. BibTeX.

```
<span class="externalcitation"> (<a id="citekey">citekey</a>, <span
class="locator">p. 23</span>)</span>
```

If you don't define a locator, you will get:

```
<span class="externalcitation"> (<a id="citekey">citekey</a>)</span>
```

Obviously, the citekey that you use in MultiMarkdown must match that used by your external tool.

Multiple Citations

When you need to combine multiple citations together, simply add them serially:

```
[p. 3][#Doe:1996][p. 10][#Smith:2005]
```

giving the output:

```
(1, p. 3) (2, p. 10)
```

I recognize that this is not really a standardized format, but again I remind you that the bibliography support in MultiMarkdown is minimal. If you want more control, or adherence to proper style rules, you need a more powerful bibliography tool.

I have written a perl script that will join these serial citations into one, `cleancites.pl`. It is run by default by the default MultiMarkdown usage scripts.

BibTeX Support

If you are a user of BibTeX, you may use it to control your references. Simply set the `Bibtex` and `Bibliographystyle` metadata as described in the section on Metadata (section 3.1), and use my xhtml2latex XSLT files as examples.

If you use this, you are not required to define your references within your MultiMarkdown document.

Advanced Citations with natbib

Advanced LaTeX users are probably familiar with the natbib[10] package, which adds additional features for bibliographic citations. It offers two new citation commands, `\citet` and `\citep`.

To use the advanced natbib features:

1. You must have the natbib package installed for LaTeX

[10] http://www.ctan.org/tex-archive/help/Catalogue/entries/natbib.html

2. You must use an appropriate XSLT file that enables the natbib package (`memoir-natbib.xslt` is an example - you can make your own)

By default, citations occur using the `\citep` command.
To use a `\citet` citation, follow the example below:

```
In their seminal paper, [Smith and Jones; p 42][#Smith1990] argue
convincingly that....

[#Smith1990]: Smith, R, and Jones, K. *Some Fancy Article* etc...
```

The text before the semi-colon indicates that we want a textual citation. In the XHTML version, the text you enter becomes the text in the sentence. When converted to LaTeX, your text is actually removed and the natbib package handles it for you. The text after the semi-colon is the usual locator text (if you don't want a locator, just leave it blank after the semi-colon).

If you don't include a semi-colon, then the `\citep` command is used in the usual fashion.

3.9 Math Syntax

Introduction to Math support

Note: *Math support within MultiMarkdown is created using MathML. MathML is not fully supported in many browsers, so your mileage may vary (I honestly don't care whether Internet Explorer works — get a real browser. Support within Firefox is pretty good, but not perfect.) This feature is quite useful, however, when generating a PDF via LaTeX.*

To view a file with MathML properly in Firefox, it must have the file ending ".xhtml". I don't know why, and it seems dumb that file extensions are so important in 2007. But for now, that's the way it is.

MultiMarkdown supports ASCIIMathML[11] a syntax for converting mathematical equations from plain text into MathML[12]. MathML can be used within properly formatted XHTML documents to display well typeset mathematical formula.

The conversion used to managed by ASCIIMathPHP[13], which was a PHP script that had to be run separately from MultiMarkdown itself. As of version 2.0b.b4, however, I am using the Text::ASCIIMathML[14] Perl module for support built into the MultiMarkdown script.

MultiMarkdown Math Syntax

Basically, use use << and >> as delimiters to indicate that you are including math in your document. You can use this to create an inline formula, or you can create independent equations, each in it's own paragraph. These can also then be converted properly into LaTeX math environments.

Additionally, you can include a `[label]` tag at the end of the equation to allow you to reference it elsewhere in your text with the label. For example:

[11] http://en.wikipedia.org/wiki/ASCIIMathML
[12] http://en.wikipedia.org/wiki/MathML
[13] http://www.jcphysics.com/ASCIIMath/
[14] http://search.cpan.org/~nodine/Text-ASCIIMathML/

```
<< e^(i pi) + 1 = 0 [Euler's identity]>>

<< x_(1,2) = (-b+-sqrt(b^2-4ac))/(2a) [quadratic equation solution]>>

You can also include formulas within a sentence, such as
<<x^2 + y^2 = 1>>.  You can then make a reference to
[Euler's identity].
```

is converted into:

$$e^{i\pi} + 1 = 0 \tag{3.1}$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.2}$$

You can also include formulas within a sentence, such as $x^2 + y^2 = 1$. You can then make a reference to Euler's identity (Equation 3.1).

Superscripts

By using the math mode above, you can include superscripts and the like in MultiMarkdown documents that don't necessarily have to be separate formulas. For example:

```
<<2^pi>>
```

becomes

2^{π}.

This is, of course, subject to the same limitations as MathML in general.

MathML Difficulties

There are some glitches in this process. First, many browsers don't fully support MathML, and sometimes you have to go through great lengths to get the browser to recognize it properly. Firefox, for instance, requires an `.xhtml` extension to properly recognize the file as `XHTML` instead of `HTML`. This may not be an ideal solution for everybody, but it **does** allow you to use a plain english syntax to represent mathematical formulas and symbols within Markdown documents, which was my goal. Others may prefer to use custom solutions using raw LaTeX source, but I didn't want to have to learn the LaTeX math syntax.

On the up side, however, this does give wonderful output when combined with my XSLT scripts to generate LaTeX documents and PDF's. I am open to input on this feature, and suspect it will become increasingly useful as browser support for MathML improves.

For more information on supporting MathML in web browsers, I have written a brief introduction to Supporting MathML[15] on my web site.

[15] http://fletcherpenney.net/2008/08/supporting_mathml

3.10 Definition Lists

MultiMarkdown has support for definition lists using the same syntax used in PHP Markdown Extra[16]. Specifically:

```
Apple
:   Pomaceous fruit of plants of the genus Malus in
    the family Rosaceae.
:   An american computer company.

Orange
:   The fruit of an evergreen tree of the genus Citrus.
```

becomes:

Apple Pomaceous fruit of plants of the genus Malus in the family Rosaceae.
An american computer company.

Orange The fruit of an evergreen tree of the genus Citrus.

You can have more than one term per definition by placing each term on a separate line. Each definition starts with a colon, and you can have more than one definition per term. You may optionally have a blank line between the last term and the first definition.

Definitions may contain other block level elements, such as lists, blockquotes, or other definition lists.

Unlike PHP Markdown Extra, all definitions are wrapped in <p> tags. First, I was unable to get Markdown *not* to create paragraphs. Second, I didn't see where it mattered - the only difference seems to be aesthetic, and I actually prefer the <p> tags in place. Let me know if this is a problem.

See the PHP Markdown Extra[17] page for more information.

3.11 Appendices

If you want to designate the final subgroup of chapters as appendices, you can include an `h1` or `h2` level header (as appropriate based on your document) with the title `Appendices`. The chapters that follow would be considered appendices when the document is converted to LaTeX using the memoir class. Since XHTML doesn't have a concept of appendices, it has no real meaning, but would at least designate this to the reader.

3.12 Glossaries

MultiMarkdown has a feature that allows footnotes to be specified as glossary terms. It doesn't do much for XHTML documents, but the XSLT file that converts the document into LaTeX is designed to convert these special footnotes into glossary entries.

The glossary format for the footnotes is:

[16] http://www.michelf.com/projects/php-markdown/extra/
[17] http://www.michelf.com/projects/php-markdown/extra/

```
[^glossaryfootnote]: glossary: term (optional sort key)
    The actual definition belongs on a new line, and can continue on
    just as other footnotes.
```

The `term` is the item that belongs in the glossary. The `sort key` is optional, and is used to specify that the term should appear somewhere else in the glossary (which is sorted in alphabetical order).

Unfortunately, it takes an extra step to generate the glossary when creating a pdf from a latex file:

1. You need to have the `basic.gst` file installed, which comes with the memoir class.
2. You need to run a special makeindex command to generate the `.glo` file: `makeindex -s ‘kpsewhich basic.gst‘ -o "filename.gls" "filename.glo"`
3. Then you run the usual pdflatex command again a few times.

Alternatively, you can use the code below to create an engine file for TeXShop (it belongs in ∼`/Library/TeXShop/Engines`). You can name it something like `MemoirGlossary.engine`. Then, when processing a file that needs a glossary, you typeset your document once with this engine, and then continue to process it normally with the usual LaTeX engine. Your glossary should be compiled appropriately. If you use TeXShop[18], this is the way to go.

Note: *Getting glossaries to work is a slightly more advanced LaTeX feature, and might take some trial and error the first few times.*

```
#!/bin/

set path = ($path /usr/local/teTeX/bin/powerpc-apple-darwin-current
    /usr/local/bin) # This is actually a continuation of the line above

set basefile = ‘basename "$1" .tex‘

makeindex -s ‘kpsewhich basic.gst‘ -o "${basefile}.gls" "${basefile}.glo"
```

3.13 Poetry Mode

By default, when you have a section of text indented with a tab, MultiMarkdown interprets this as a code block. This allows you to more exactly control the spacing and line endings, but it also applies a monospace font in both the XHTML and LaTeX outputs. This is the usual way of demonstrating source code in documents.

Some authors, however, don’t write about source code, but would like a way to control line endings (when writing poetry, for example).

To accomplish this, there are several alternate XSLT files included within the MultiMarkdown distribution that are labelled with a `poetry` filename. These XSLT files handle the code blocks in a slightly different way to make them more suitable for text, rather than code. I encourage you to give this a try.

[18] http://www.uoregon.edu/∼koch/texshop/

At the current time, there is no way to use both formats within the same document, except to format them manually. This may change in the future, depending on some decisions John Gruber needs to make about the standard Markdown syntax.

3.14 Miscellanea

In addition to what is mentioned elsewhere in this document, MultiMarkdown does a few things slightly differently:

- `&copy;` entities are converted to `&#xA9;` so that they can pass through an XSLT parser
- `*` and `_` are not interpreted as or when they occur in the middle of words. This caused too many problems with URL's.

MultiMarkdown supports the conversion of colored spans of text from XHTML to LaTeX using the xcolor package. For example:

```
<span style="color:#888888">net</span>
```

becomes:

```
{\color[HTML]{888888} net}
```

There is not currently a syntax shortcut for this, you have to manually add the information. This technique is used to support annotations from Scrivener, for example.

Chapter 4

MultiMarkdown and LaTeX

LaTeX[1] is a professional quality typesetting system that can be used to take plain text markup and produce a high quality pdf, complete with table of contents, index, glosssary, etc. It's a fairly complicated program, but capable of doing most of the work for you. One of my goals with MultiMarkdown was to make it even easier to create a LaTeX document, with minimal knowledge of the LaTeX syntax. In fact, you can create fairly complex documents without any understanding of how LaTeX works, as long as you have it installed correctly.

That said, MultiMarkdown is not simply a preprocessor for LaTeX files, so there will always be LaTeX commands that are just not available from within MultiMarkdown. If you're a LaTeX expert, you might find that after MultiMarkdown runs, you want to go and hand tweak a few parts to get things just right. But for the average user and average document, I suspect the default output will be just fine.

The settings to pay particular attention to:

- You must choose an XSLT file to convert the MultiMarkdown-generated XHTML into LaTeX; you do this by setting the `LaTeX XSLT` metadata. If you do not choose one, the default is `memoir.xslt`. Most of my XSLT files are based around the `memoir` package — it's the one I'm familiar with, it's very flexible, and has high quality output, and lots of features. That said, you are welcome to create your own XSLT files to use whatever packages you prefer. The beauty of the XSLT transformation process is that it can be completely reconfigured however you like.

- Depending on what sort of document you are creating, you may need to set the `Base Header Level` metadata. For example, if you are creating a `memoir` based document, and wish for your top-level section to be a chapter, rather than a "part", you could set `Base Header Level` to 2. It's easier to do than explain, but basically it moves all levels of your structure by the specified number of steps.

- You likely will want to set as much of the basic metadata as possible (e.g. `Title`, `Author`, `Date`, `Keywords`, etc) as most of this is converted to a format that is used in the resulting PDF.

Also, MultiMarkdown has support for BibTeX[2], glossaries, html links, internal links between sections of the document, math formatting, etc. Most of the "major" features of

[1] http://www.latex-project.org/
[2] http://en.wikipedia.org/wiki/BibTeX

LaTeX are available using the standard MultiMarkdown syntaxes. If there is something you don't see, just ask — it may exist, or I might be able to add it if appropriate.

The general process of creating a PDF via LaTeX is the same as the normal use of MultiMarkdown, with one additional step:

1. Create your text source file
2. Using your method of choice, convert the text file to XHTML, and then convert the XHTML to LaTeX (most of my tools will do this as a single step as far as the user is concerned).
3. Convert the LaTeX source file to PDF using the tool of your choice (my Drag and Drop application[3], TeXShop[4], latexmk[5], manually, etc.)

Due to the complexity of the LaTeX source, it can be hard to troubleshoot when using an automatic tool. If something doesn't work, I recommend first trying to get your MultiMarkdown text file converted to XHTML and verify that it is correct. Then convert the XHTML to LaTeX and be sure that you can watch the status messages that occur during processing of the LaTeX file - they will usually give you a hint as to where the problem lies. Remember, just because the XHTML version of a MultiMarkdown document is valid XHTML does not mean the resulting LaTeX will be totally valid.

[3] http://fletcherpenney.net/MultiMarkdown_Drag_and_Drop
[4] http://www.uoregon.edu/~koch/texshop/
[5] http://www.phys.psu.edu/~collins/software/latexmk-jcc/

Chapter 5

Advanced Features and Customization

I believe that MultiMarkdown works pretty well "out of the box" for the vast majority of users (of course, I'm not biased or anything...) But more advanced users will eventually start thinking about features that they wish existed. Some of these features are very specific to their own documents and style, but others are more general and would be of use to everyone.

5.1 How do I find out about feature x?

My recommended approach is:

1. Make sure you check through the documentation on the web site (there is a search feature). An increasing number of feature requests are for things that already exist.
2. Check the MultiMarkdown discussion list to see if someone has already suggested your feature, or better yet, has already solved it.
3. Decide whether it's something you could try and do yourself, or whether you need to ask for help to accomplish it. Either way, the results can be shared on my web site to help others.

5.2 How do I customize MultiMarkdown?

The first step in trying to customize MultiMarkdown is to figure out where in the workflow the customization needs to occur:

1. Does the MultiMarkdown perl script need to be modified to add a new syntax, or change the way the output is generated? There should be fewer and fewer necessary changes in this step as the MultiMarkdown syntax matures. Also, note that I am hesitant to add new features at this level that increase the complexity of markup. It's not impossible, but I will definitely need to be convinced it's the only way to go.
2. Can the desired feature be implemented through a modification of one of the XSLT files? XSLT is a powerful tool, and can be used to really customize the XHTML or LaTeX output from a MMD document. (*Many users would likely benefit from a generic XHTML to RTF XSLT stylesheet - I have been unable to locate one that would work, and I have no need of RTF documents. This would be too much work for too*

little gain for me, but I am sure someone out there needs exactly this sort of tool.) Browse through the `XSLT` directory and look to see if there is a stylesheet that could be modified to do what you want. The XSLT syntax is not that complicated, but does take some getting used to. As examples, the `xhtml-toc.xslt` script parses the header tags in the XHTML output, and creates an automatic table of contents at the top of the XHTML file. The `xhtml-poetry-support.xslt` file looks for code blocks that start with `[poetry]` and changes them to a poetry mode, rather than code (basically removing the monospace font).

3. Does the desired feature need to be implemented in a separate post-processing script? For example, for LaTeX documents I use a script called `cleancites.pl` that looks for strings of multiple citations to shorten the syntax. You could easily create a script to do whatever you like and incorporate it into your work flow.

In summary, a great many features and customizations can be added to MultiMarkdown by users. I also recommend that you consider sharing any of your customizations back to the MultiMarkdown community - I am happy to put any files or links on my site, if you are interested.

5.3 Where do I dig in the MultiMarkdown package to find out more?

Again, places to look for inspiration:

- MultiMarkdown/bin - this is where the "glue" scripts live that manage different MultiMarkdown workflow patterns. You can create your own shell scripts that can add additional steps to your workflow here.
- MultiMarkdown/Utilities - a couple of utility scripts and the `cleancites.pl` post-processing script live here; you can add files here and incorporate them into your work flow.
- MultiMarkdown/XSLT - XSLT files for modifying XHTML files or creating LaTeX files go here. Lots of examples for different styles of output or customizing the way various features work.
- `http://fletcherpenney.net/multimarkdown/xslt_files/` - this is where I will place various user submitted files that may be of interest, or offer a starting point for further customization. Please consider submitting your own improvements here as well.

Chapter 6

Component Software

The MultiMarkdown system is actually a patchwork of multiple programs, which are run in a specific order by shell scripts. I have written the glue utilities, and the MultiMarkdown modifications to John Gruber's original Markdown program, but I can't take credit for the rest.

6.1 MultiMarkdown

- by Fletcher T. Penney
- `http://fletcherpenney.net/multimarkdown/`

MultiMarkdown is my update to John Gruber's Markdown[1] software. It is what this bundle is based on. To learn more about why you would want to use this bundle, check out the web page for MultiMarkdown.

6.2 SmartyPants

- by John Gruber
- `http://daringfireball.net/projects/smartypants/`

SmartyPants is another program by John Gruber, and is designed to add "smart" typography to HTML documents, including proper quotes, dashes, and ellipses. Additionally, there are several variations of the SmartyPants files to handle different localizations (specifically, Dutch, French, German, and Swedish). These localizations were provided by Joakim Hertze.

6.3 Text::ASCIIMathML

- by Mark Nodine
- `http://search.cpan.org/\protect$\relax\sim$nodine/`

This perl module adds support for converting the ASCIIMathML syntax into MathML markup suitable for inclusion in XHTML documents.

[1] http://daringfireball.net/projects/markdown/

6.4 ASCIIMathPHP (Deprecated)

- by Kee-Lin Steven Chan
- `http://www.jcphysics.com/ASCIIMath/`

This bundle includes the MultiMarkdown specific variant of the original ASCIIMathPHP. It allows you to use the ASCIIMath syntax to describe mathematical formulas in plain text language.

This software has been replaced by Text::ASCIIMathML.

6.5 XSLTMathML

- by Vasil Yaroshevich
- `http://www.raleigh.ru/MathML/mmltex/index.php?lang=en`

This bundle includes the MultiMarkdown specific variant of the original XSLTMathML. It converts XHTML with MathML markup into LaTeX math environment code. Very handy for making well typeset documents that are math-heavy.

Chapter 7

Applications That Support MultiMarkdown

There are several applications and utilities out there that include support for MultiMarkdown, that can make it even easier to create your output documents.

If you know of something not included here, please let me know.

7.1 Movable Type

MultiMarkdown can be used with Movable Type. To install:

1. Place `MultiMarkdown.pl` in the `mt/plugins/Markdown` directory
2. Copy `ASCIIMathML.pm` into the same directory
3. Make sure `SmartyPants.pl` is also there

Now MultiMarkdown should be working with Movable Type. For some reason, however, it seems somewhat temperamental at times. I haven't been able to figure out why, but it works for me on my local machine and on my host's server. It has also worked for other users.

7.2 MultiMarkdown Drag and Drop

Early on, as MultiMarkdown became increasingly powerful (and complex) I realized that most people would want something a little easier to use than what had become a rather complicated command line string.

The first solution was a set of Drag and Drop applications created using Platypus[1]. These were designed to allow you to drop a MultiMarkdown text file on the application icon, and they spit out a .xhtml, .pdf, .rtf, or .tex file, depending on which application you used.

These utilities are still available, and have been updated to work with the "Common" MultiMarkdown Installation:

- `http://files.fletcherpenney.net/MultiMarkdownDragAndDrop.zip`

[1] http://www.sveinbjorn.org/platypus

7.3 Scrivener

Scrivener[2] is a:

> ... project management tool for writers that acts like your own little writing shed at the bottom of the garden, where you have cork notice-boards, ring-binders, photos, clippings paperclipped to jottings, notebooks and reams of typewritten pages piling up - along with a secretary who keeps it all in neat piles and uses his speed-reading skills to find what you need as soon as you need it. [2]

As of beta 3, Scrivener has the ability to export to a MultiMarkdown text file, or to run the conversion utilities to create XHTML, RTF, or LaTeX files. It also has support for MultiMarkdown metadata.

Scrivener's strengths, as they relate to MultiMarkdown, included the ability to arrange and re-arrange your document as desired using its outliner view, cork-board, and other features. It also has some limited ability to convert RTF bold and italic formatting into MultiMarkdown syntax, which can be useful when converting documents from other formats.

Scrivener is primarily focused towards creative writing, but when combined with MultiMarkdown it is very useful for academic and technical writing where a LaTeX file is highly desirable.

Keith Blount has done a great job with Scrivener, and I was happy to be able to help implement support for MultiMarkdown. I look forward to helping to continue to use and refine this program myself.

At this time, Scrivener is in public beta, and should be available for purchase towards the end of 2006 or beginning of 2007. But the beta is very usable as is, and gives you until Jan 2007 or so to try it out.

For more information, I have created a User's Guide to MultiMarkdown and Scrivener:

- `http://fletcherpenney.net/multimarkdown/using_multimarkdown_with_scriv/`

7.4 OmniOutliner

I have written an export plugin for OmniOutliner that allows you to craft your MultiMarkdown documents within OmniOutliner, and then export to a text file (or folder with text file and images), that can then be processed with Markdown or MultiMarkdown.

`http://fletcherpenney.net/multimarkdown/multimarkdown_and_omnioutliner/`

7.5 TextMate

TextMate[3] is a powerful text editor that:

> brings Apple's approach to operating systems into the world of text editors. By bridging UNIX underpinnings and GUI, TextMate cherry-picks the best of both worlds to the benefit of expert scripters and novice users alike....
>
> Created by a closet UNIX geek who was lured to the Mac platform by its ease of use and elegance, TextMate has been referred to as the culmination of

[2] http://www.literatureandlatte.com/scrivener.html
[3] http://macromates.com/

> Emacs and OS X and has resulted in countless requests for both a Windows and Linux port, but TextMate remains exclusive for the Mac, and that is how we like it! [3]

TextMate is somewhere between a text editor for programmers, and a writing tool. If you like being able to customize your writing environment, and like fancy tools to handle the formatting for you, then TextMate might be the app for you.

Allan Odgaard created an initial Bundle that added Markdown support to TextMate. It included some basic MultiMarkdown support as well. But to be honest, *I* had trouble getting it to work. And if I had difficulty, I can only imagine how much trouble others had.

So I created my own Bundle. It includes a lot of features that automatically format metadata, lists, tables, headers, etc. It can clean up the text to make it look as presentable as possible in plain text, and it can then automatically convert your text into XHTML, RTF, Word, or LaTeX/PDF.

I have subsequently rewritten this bundle as a fork of the original on github. This should make it easier to incorporate changes, and possibly to merge the two projects into a single bundle.

- `http://fletcher.github.com/markdown.tmbundle/`

7.6 Using Scrivener and TextMate Together

It is possible, using the "Edit in TextMate" feature from TextMate. Basically, it adds the ability to edit any Cocoa based text editor view in TextMate. This allows you to edit the text from a Scrivener document in TextMate, in order to take advantage of the automatic formatting, while still retaining the organizational features of Scrivener.

This feature has its limitations (it breaks the undo stack in Scrivener) and is only for advanced users. I take no responsibility for it, as I didn't write Scrivener or TextMate. But it can be useful...

To learn more, check out the information on Cocoa Text Fields:

- `http://manual.macromates.com/en/using_textmate_from_terminal`

7.7 The "Common" MultiMarkdown approach

During beta testing of the MultiMarkdown support with Scrivener, it was proposed that having a standard location for MultiMarkdown could make it easy to integrate with various applications, and to allow the user a single place to update their MultiMarkdown files, independent of the application it was being used with.

For Mac OS X users, this boils down to allowing a MultiMarkdown installation to be placed in one of two locations, where it is available to any application that knows to look for it there:

- ~/Library/Application Support/MultiMarkdown
- /Library/Application Support/MultiMarkdown

The first is available only to the user, and the second is available to anyone on that computer.

When Scrivener, or another application that supports this feature is run, it checks to see if a MultiMarkdown installation is available in either of those places. If not (the first time you run the program, for instance), then some programs might install a version of MultiMarkdown here; others might simply use an a copy of MultiMarkdown embedded within the application bundle.

The benefit of this approach is that if I update MultiMarkdown, you can simply replace the updated files in the Application Support folder, without having to update the other applications.

Please let me know if you have suggestions on improving this feature, or if you are interested in including support for MultiMarkdown in your own application.

Naturally, this approach only works with Mac OS X. If anyone is interested in working on a similar feature for other operating systems, please let me know.

7.8 Create Your Own

Between shell scripts, applescripting, Automator, and other tools, you can usually find an easy way to incorporate MultiMarkdown into your own workflow. If you find something that you think should be added here, let me know!

Chapter 8

Technical Issues

The MultiMarkdown system is actually a fairly complex group of programs, which includes multiple perl utilities, a PHP program, and multiple XSLT files. With some hand waving, I try to make it look like a single coherent program, but it actually uses multiple utilities written by multiple people.

This section is designed to address some of these issues, and implications they may have for users, programmers, etc.

8.1 XML Namespace Issues

As of version 2.0.a3, there has been a complete overhaul of the way XML namespaces are handled. This required changing all of the XSLT files to use an "html" alias for the "http://www.w3.org/1999/xhtml" namespace.

It appears to be working, including support for MathML.

If you have any custom XSLT files, you will need to make the same changes, specifically:

Make your stylesheet declaration look like:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:html="http://www.w3.org/1999/xhtml"
    version="1.0">
```

And relabel any references to XHTML elements, so that they are preceeded by "html:", e.g.

```
<xsl:template match="/">
    <xsl:apply-templates select="html/head"/>
    <xsl:apply-templates select="html/body"/>
    <xsl:call-template name="latex-footer"/>
</xsl:template>
```

should look like this:

```
<xsl:template match="/">
    <xsl:apply-templates select="html:html/html:head"/>
```

```
    <xsl:apply-templates select="html:html/html:body"/>
    <xsl:call-template name="latex-footer"/>
</xsl:template>
```

I have left the `-novalid` and `-nonet` options in place to prevent unnecessary errors when you are not connected to the internet, but these could be removed if desired.

8.2 XeLaTeX Tips

If you are using XeLaTeX to process your document (useful for utilizing Mac OS X fonts in your document), you want to use font declarations like this:

```
\font\addressbold="Garamond Bold:mapping=tex-text" at 8pt
```

By including the ":mapping=tex-text" portion, you regain use of smart quotes, en- and em- dashes, etc. I'm sure most XeLaTeX users know this, but it took me a bit of trial and error to discover it....

Chapter 9

Acknowledgments

Thanks to the individuals and groups below for their contributions to improving Markdown and MultiMarkdown:

- John Gruber
- Michel Fortin
- Jonathan Weber
- Mark Eli Kalderon
- Choan C. Gálvez
- Dr. Drang
- Robert McGonegal
- David Green
- Trey Pickard
- Saleem
- Melinda Norris
- Sean Wallace
- Allan Odgaard
- Stefan Brantschen
- Keith Blount
- Amber Vaesca
- Gerd Knops
- John Purnell
- Jonathan Coulombe

- Jason Bandlow
- Joakim Hertze
- Kee-Lin Steven Chan
- Vasil Yaroshevich
- Matt Neuburg
- James Howison
- Edward Nixon
- etherean
- Özgür Gökmen
- Chad Schmidt
- Greg (gr)
- Ben Jennings
- Silvan Kaiser
- Tomas Doran
- Rob Walton
- Dan Rolander
- Duoyi wu
- Dan Dascalescu
- Ingolf Schäfer
- Chris Bunch
- Oblomov
- Alex Melhuish
- Stephan Mueller
- Josh Brown
- Rob Person
- Matthew D. Rankin
- Dawid Ciężarkiewicz
- Joonas Pulakka
- ipetraka

and others I have surely forgotten....

Chapter 10

Known Issues

- The <<...>> syntax can cause only the first < to be encoded as < if there is no trailing space (e.g. <<`something`>> vs. << `something` >>). I suspect that I will have to manually look for any << and convert them. I guess this is technically an issue with Markdown and not MultiMarkdown, but it has apparently not come up before.
- I tried to remove dependence on the `varwidth` package. This screws up the formatting of footnotes in tables, and also RTF exporting of tables. I'm not sure what to do - varwidth is incompatible with xcolor and is not a standard package. Suggestions welcome.
- Creating a link to an image by label doesn't work properly anymore
- I'm having difficulty with getting the glossary feature to work in the non-memoir classes. At some point I will look into this, but if someone else out there can point out what I'm doing wrong, let me know.
- RTF support currently only exists for Mac OS X. Conversion from XHTML to RTF happens via Apples `textutil` tool. It is possible to write an XSLT file that converts from XHTML to RTF, but I have little to no interest in writing this myself, as I don't really use the RTF format very often. If someone were interested in developing this, I would help out. An added benefit would be that the XSLT could actually do a better job than Apple's tool in terms of footnote support and internal links. Contact me if you're interested. In the meantime, I suggest using Google Docs[1] to import the XHTML file, and then export as an RTF. It does a much better job.
- The sample MMD file creates two copies of the footnote in the `MultiMarkdown vs. Crayons` table, even though I only call for one. Not sure where the `a` footnote comes from.... Any help in tracking this down would be appreciated, as it didn't used to do this.

[1] http://docs.google.com/

Chapter 11

Things to Do

- Add a syntax to allow comments that can be stripped before passing the output to the parser
- write a routine (that would be separate from MultiMarkdown) to download linked images, save them to a tmp directory, and then convert them for use within a pdf.
- Decide on appropriate management of alignment when a cell spans multiple columns. Currently, the alignment of the first cell is used. (If Markdown goes to a whitespace-based alignment option, that could be used in this instance.)
- Consider whether there is a reasonable syntax for table cells that span multiple rows.
- Consider a syntax for superscripts (this has been discussed before) - could convert it to MathML syntax? Or just use math markup instead as described in Superscripts (section 3.9).
- Certain markup gets processed within headers and shouldn't, e.g. `<img>`
- Consider whether to incorporate the definition list syntax into a footnote to specify a glossary entry (or perhaps even without the footnote), or whether to leave well enough alone.

Chapter 12

Version History

> Release early, release often!
>
> Linus Torvalds

- 2.0.b6 - Fix support for base header level with Setext-style headers (thanks to Rob Walton); improve Windows support;

- 2.0.b5 - spaces at end of xslt filenames won't cause failure; use `\url{}` for "non-referenced" url's in LaTeX to allow linebreaks (though they still don't always break correctly — this is a problem with `hyperref` not MMD); don't convert `^` to exponents in the `clean-text-allow-latex.xslt` file so that math code works properly; the S5 XSLT file at least partially works again now; update the TextMate bundle to work with Leopard; updated the envelope and letterhead files; include `6x9book-real-poetry` XSLT that uses memoir's poetry features fairly well; rework the `clean-text` files to make them easier to update in the future and more modular; XHTML comments are now passed through as raw LaTeX; unescape encoding within comments;

- 2.0.b4 - empty labels for headers now produce valid XHTML (e.g. no `id=""`); fix bug in `clean-text.xslt` that caused a problem with closing double quotes; the `.xslt` extension is no longer required in metadata; added customizable letterhead XSLT; fix bug in table support that choked on extra spaces at end of lines; *Major Change*: switched to Text::ASCIIMathML for math support, meaning that everything is once again perl based (this enables math features on web sites using MultiMarkdown, for example); fix bug that occurred when 'Abstract' was not the first chapter;

- 2.0.b3 - move the `clean-text` routine from `xhtml2latex.xslt` into it's own file (to allow easier modification by users); create alternate version that does not protect certain characters in order to allow raw LaTeX code to be passed through; added `latex-snippet.xslt` stylesheet for inclusion in outside LaTeX template systems; added `xhtml-poetry-support.xslt` and `xhtml-toc.xslt` to demonstrate how to extend MMD functionality for XHTML output with new system; fix bug in SmartyPants that processed typography within <`style`> sections (thanks AmberV); fix handling of links by reference in headers and handling of attributes when links are referenced multiple times (thanks to Edward Nixon); fix bug in epigraphs (thanks etherean); improve id generation for footnotes - e.g. match behavior of PHP Markdown Extra

(thanks to Özgür Gökmen); fix bug in id generation for ToC for XHTML documents; fix problem with \ldots command (thanks to etherean and James Howison); fix issue with   and tilde character; fix bug where footnote special characters were not unescaped (thanks to Chad Schmidt); clean up documentation a bit;

- 2.0.b2 - fix processing of footnotes so that ending in a blockquote doesn't break validity; fix bug in letter.xslt; overhaul XSLT system to allow for different XSLT files for different output formats (e.g. HTML, RTF, LaTeX);

- 2.0.b1 - fix bug in _StripLinkDefinitions that prevented detection of single character labels; change \textwidth to \linewidth in LaTeX export XSLT files (let me know if this causes problems); add Windoze compatibility to the perl scripts (thanks to Jason Bandlow for pointing out this problem, as well as for suggesting a fix);fix issues with glossary support and document the process; complete overhaul of the way namespaces are handled (stripnamespace.pl is no longer needed, XSLT files are rewritten, -nonet and -novalid should be optional for xsltproc); update the Drag and Drop applications to use the "Common" MMD Installation; update to Markdown 1.0.2b8 codebase; add support for natbib and \citep and \citet;

- 2.0.a2 - fix some minor problems with XSLTMathML; allow math to be enclosed in parentheses; change matching for bottomrule in tables; improve handling of tables with no header row (only a header column);

- 2.0.a1 - strip spaces from metadata keys for XHTML validity; make XHTML footnote output more compatible with Gruber's website and PHP Markdown Extra; update XSLT to address these changes (*Note*: this breaks compatibility with prior versions); add support for definition lists; fix bug when escaping WikiWords in code; add XHTML Header metadata, and update XSLT to ignore <style> tags; add support for the XSLT File metadata tag, which allows a single command to parse any MultiMarkdown file; add additional XSLT files; add the multimarkdown2XHTML.pl and related commands; article XSLT now uses the article option in memoir, rather than the article class; delete the report class (use memoir instead); fix a **lot** of "minor" bugs; add the "6x9book.xslt" option; allow custom cross-reference labels to headers; give preference to defined links over automatic cross-references; add "poetry" versions of several XSLT files (treat code blocks as formatted text, rather than code — useful for formatting poetry)

- 2.0.a - New version numbering scheme; update to Markdown.pl 1.0.2b7 code; add support for [link reference] shortcut syntax (i.e. no trailing []) for MultiMarkdown crossrefs; add an extra newline in verbatims to add space before the next paragraph; synchronize numbering schemes of all related MultiMarkdown tools to make it easier to ensure compatibility; add revision numbers to source documents to help track incompatibilities; add LaTeX support for i.e. and e.g.; TextMate MultiMarkdown bundle available; update MultiMarkdownDragAndDrop tools to new codebase; now distributed as a zipfile.

- 1.0.1Multi19.4 - major update; fix issue where cross-references to images defined by alt text had to follow the image in the document; add support for MathML via ASCIIMathPHP; change name to id for footnotes; move _DoHeaders in front of _DoTables to allow cross-references inside tables; fix handling of citations without locator; a table with no header titles and no column alignment row is interpreted as a pull-quote -

this is experimental and may be changed; the `Bibliography Title` metadata field is available for LaTeX to rename the bibliography section; multiple changes to XSLT files to improve compatibility; support for << `math` >> syntax using ASCIIMathPHP; change `HeaderLevel` to `Base Header Level` and process it in XSLT rather than in the OmniOutliner tool; support for `Affiliation` metadata element; add equation label to possible cross-reference list; compatible with epigraph feature for XSLT conversion to LaTeX; document table labeling feature and default to caption if no label present;

- 1.0.1Multi19.2 - require leading space before unescaping \WikiWord; fixed bug where attributes not included with images; add `Bibliography Title` metadata key; fix bug with invalid leading characters in header id attributes; allow '-' and '_' in metadata; fix handling of citations in footnotes; fix issue with quotes in link attributes.

- 1.0.1Multi19.1 - minor change to bibliography formatting to allow translation into a \BibTeX compatible format *without* the use of a `.bib` file;

- 1.0.1Multi19 - Major update; fix bugs discovered by testing with MarkdownTest 1.0; don't add leading blank line if no metadata exists; fix parsing of link definitions, including attribute parsing; various clean- ups to code and documentation; improve cross-reference handling of special characters; fix bug in handling of wiki links (`/` is not automatically added any more); fix bug in `title` attributes of images; re-enable the inclusion of `DOCTYPE` in complete documents (this requires the use of the `-nonet` and `-novalid` options in xsltproc; fix bug in handling of `**`; fix bug where WikiWords in code blocks and spans were not unescaped; fix bug where digits were not allowed in metadata keys; fix numbering of footnotes so that they remain in proper order; add basic citation and bibliography features; major bug fixes and testing to precede the release of version 20 (2.0)

- 1.0.1Multi18 - further work to make WikiWord escaping work properly...

- 1.0.1Multi17 - add support for "char" alignment in table columns (**NOTE**: browsers do not currently support this); fix bug with \ in code spans when WikiWords are disabled; fix bug in bold/italic detection

- 1.0.1Multi16 - can now optionally have header in first cell of each row; fix bug in footnote counting (thanks to Mark Eli Kalderon for pointing this out);

- 1.0.1Multi15 - allow for multiple <`tbody`> span's within a table; ensure that the variable`$g_empty_element_suffix` is used everywhere; protect code spans from table parsing

- 1.0.1Multi14 - captions can now be before or after table; add syntax for column spanning within tables (body and header)

- 1.0.1Multi13 - added support for CSS metadata key; allow no alignment option on table cells; support for captions for tables

- 1.0.1Multi12 - added support for image/link attributes; fixed bug in table handling

- 1.0.1Multi11 - added support for table syntax

- 1.0.1Multi10 - allow emphasis at beginning of line

- 1.0.1Multi9 - fix bug in metadata parsing
- 1.0.1Multi8 - first draft of fix for "underscore within a word" problem that causes so many errors with URL's. Now a leading whitespace is required in front of the "opening" `_` or `*` for it to be interpreted as emphasis or strong.
- 1.0.1Multi7 - add Wiki Links support
- 1.0.1Multi6 - correct bug in footnote id handling (Thanks to Jonathan Weber for pointing this out)
- 1.0.1Multi5 - allow disabling of metadata feature
- 1.0.1Multi4 - convert `&copy;` entities to `&#xA9;` (compatible with XSLT); generate cross-refs for images
- 1.0.1Multi3 - fix metadata parsing in the event a key was empty
- 1.0.1Multi2 - add support for footnotes. **Major** change - no longer use templates, but rather will focus on using XSLT to convert from XHTML output to other formats. I think this will be more flexible and less error prone.
- 1.0.1M - initial release

Bibliography

[1] Daring Fireball: Markdown.
http://daringfireball.net/projects/markdown/ 1

[2] Literate and Latte - Scrivener.
http://www.literatureandlatte.com/scrivener.html 29

[3] TextMate — The Missing Editor for Mac OS X.
http://macromates.com/ 30