Big Markdown/etc. Reference File

kukkurovaca

2014

Markdown Variants

There are lots. This is both a great and a terrible thing about Markdown. Great because it's nice to have choices, but terrible because without One True Markdown Flavor, there's uneven implementation across the tons of programs, apps, etc. that use Markdown. Everybody supports the basic, original recipe Markdown, but that doesn't have footnotes, for fucks' sake.

At this point, I'm pretty much settled on MultiMarkdown. It's very full-featured and well-supported – at least on OSX – and I like how it does metadata a lot. Most people probably aren't interested in using metadata in their Markdown files, of course. (I'm not even consistently doing so yet.)

MultiMarkdown

MultiMarkdown Links:

- Multimarkdown
  - Download
  - User Guide
  - Cheat Sheet
  - LaTeX support files

MMD Metadata

I don't currently use MMD Metadata very extensively. What I am using it for is:

- Very basic info like title, author, date, which are passed through to converted output in useful ways. For example, setting the title in the metadata means that the title will look right in LaTeX/PDF files.
- LaTeX-specific metadata, for using the MMD LaTeX templates
- Scratchpad stuff – it's a handy place to put general todos that I don't necessarily want to show up when I'm printing or previewing the document, and which don't necessarily belong in any particular point in the document. (Where I might insert a comment or a CriticMarkup note.)

Example:

```
latex input: mmd-tufte-handout-header
Title: Big Markdown/etc. Reference File
Author: kukkurovaca
Date: 2014
Todo: Stuff and things
latex input: mmd-tufte-handout-begin-doc
latex footer: mmd-tufte-footer
```

I keep forgetting to add extra space at the end of each line, per Penney's suggestion:

> 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.

References in MMD

Official examples:

```
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.

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]
```

Note that:

```
If you are creating a LaTeX document, the citations will be
included, and natbib will be used by default. If you are not using
BibTeX and are getting errors about your citations not being
compatible with 'Author-Year', you can add the following to your
documents metadata:

latex input:        mmd-natbib-plain

This changes the citation style in natbib to avoid these errors,
and is useful when you include your citations in the MultiMarkdown
document itself.
```

Interestingly, I can't seem to make citations go into the margins on tufte-latex, even if I manually modify them.

Would a possible workaround be to use footnotes [1] and insert MMD metadata variables? Yes, that works fine, but it does mean typing more characters and no title formatting.

Note It appears that this is an area where `pandoc -f markdown _mmd` does not yield MMD-like results. Interesting.

MMD Support Files

One of the annoying things about MMD for non-technical users is that there's a goodly chunk of functionality that won't work if you don't have certain template and support files in certain directories, which are installed separately from MMD itself.

- The LaTeX Support Files need to go in the TeXMF folder, which may not always be in the same place, depending on how/where you installed LaTeX.
  - On my mac this is `/Users/kukkurovaca/Library/texmf/tex/latex/mmd`
- You may *also* need the *MMD-Support* files. On OSX, there is an installer for these; I'm not sure yet where you need to put them on Windows.
  - On my mac this folder is `/Users/kukkurovaca/Library/Application Support/MultiMarkdown`
  - Also had to add this to my .bash_profile: `export PATH=/Users/kukkurovaca/Library/"Application Support"/MultiMarkdown/bin:$PATH`

The documentation is also pretty fuzzy on how to actually the xslt support file to produce a table of contents. It may be that you just put `xhtml xslt: xhtml-toc-h2.xslt` in your metadata?

Pandoc

References in Pandoc

From the demos section: `pandoc -s -S --biblio biblio.bib --csl chicago-author-date.csl CITATIONS -o example24a.html`

```
Example:

#Pandoc with citeproc-hs

-   [@nonexistent]

-   @nonexistent
```

[1] [%Doe2006]

```
-   @item1 says blah.

-   @item1 [p. 30] says blah.

-   @item1 [p. 30, with suffix] says blah.

-   @item1 [-@item2 p. 30; see also @item3] says blah.

-   In a note.[^1]

-   A citation group [see @item1 p. 34-35; also @item3 chap. 3].

-   Another one [see @item1 p. 34-35].

-   And another one in a note.[^2]

-   Citation with a suffix and locator [@item1 pp. 33, 35-37,
     and nowhere else].

-   Citation with suffix only [@item1 and nowhere else].

-   Now some modifiers.[^3]

-   With some markup [*see* @item1 p. **32**].

# References

[^1]: A citation without locators [@item3].

[^2]: Some citations [see @item2 chap. 3; @item3; @item1].

[^3]: Like a citation without author: [-@item1], and
now Doe with a locator [-@item2 p. 44].
```

Testing with a reference to *Wind Whales of Ishmael*. [@farmer_wind_1979] Or how about [@badger_pleasures_2010]

CriticMarkup

CriticMarkup seems cool, b/c one thing that plain text editing is not natively great at is helping you keep track of your comments and managing how those comments are or are not exported to the output.

It's pretty well-integrated into MMD. However, it is not supported by Pandoc's `markdown_mmd` extensions. I know CriticMarkup comments come through as body text in Pandoc comments, which is undesirable. I need to figure out if that's something that can be addressed via the CriticMarkup CLI. (MMD doesn't seem to be a help here becuase it doesn't output Markdown).

So, I guess I might need to actually go ahead and use the actual CriticMarkup tool. But then I have to install Python on my Windows machine? ::sigh::

Another alternative would be to just ditch CM and use some other commenting format. This seems to be the safest against both HTML and LaTeX output:

```
<!--
```

```
% Comment
..>
```

Command Line Tools

Multimarkdown

{>>TODO: List & differentiate the various MMD commands.<<}

Misc. MMD Notes

MMD's smart typography follows Gruber's Smartypants:

- Straight quotes (" and ') into "curly" quote HTML entities
- Backticks-style quotes ("like this") into "curly" quote HTML entities
- Dashes ("–" and "—") into en- and em-dash entities
- Three consecutive dots ("...") into an ellipsis entity

MMD definition lists are made like:

```
Item
:   definition
```

Pandoc

Pandoc is really, really cool. It does conversion between tons of formats, but my main interest is in its ability to take Markdown and in one step turn it into good-looking PDF or MS Word files. It's crazy that it works at all, and in fact it works really well.[2]

There are OSX services and something vaguely similar on Windows to do these conversions with a right click, which is great for those of us who are command line averse. However, I found that in both cases I needed to modify them, since I want to be able to use MMD metadata and in the case of Windows because my Pandoc install is in a nonstandard place.

What this means is that I can write a document using MMD in a text editor, save it, right click and convert to PDF or .docx, and have instant, good-looking, consistent documents to print or give to folks with very little fuss.

Misc. Pandoc Notes:

To include a table of contents: `--toc`. However this seems to only work with `-s` (standalone) also set.

Pandoc's typography:

[2] The one hiccup I've had with Pandoc is that in the current version (1.12.3.2?), there's a bug with reference citations when one of the extensions is turned on. Bug should be fixed in the next release.

Produce typographically correct output, converting straight quotes to curly quotes, — to em-dashes, – to en-dashes, and ... to ellipses. Nonbreaking spaces are inserted after certain abbreviations, such as "Mr." (Note: This option is significant only when the input format is markdown, markdown_strict, or textile. It is selected automatically when the input format is textile or the output format is latex or context, unless –no-tex-ligatures is used.)

Pandoc and PDFs via LaTeX

Pandoc's default LaTeX output is shockingly "just works," especially compared to working with MMD's LaTeX output or (god forbid) actually composing LaTeX. However, up to this point, I've been a little vague on how to go about customizing the output or in general getting anything other than the very stodgy default LaTeX output.

Pandoc LaTeX Metadata and Command Line Options

These options can be set via the command line using `-V` or `--variable`. They may also be able to be set via header metadata in the file.

I need to actually test this using MMD headers to see if it works with `-f markdown_mmd`. *Update*: Okay, it definitely works for fonts, but I can't figure out how to make it work with toc.

One thing that I'm having difficulty deciding so far is what to set via the command line (or via anything that uses a shell script) vs. what to set via the document header vs. what to set via a new template. In broad terms, there are some things that *have* to be set via template, b/c they don't fit into stock variables. (I forget whether you can define new variables – but of course that would still require template modification if so.) But beyond that, it's sort of tricky.

header-includes I think this lets you specify arbitrary commands to be inserted into the preamable of the document. Here's an example for a YAML header –

```
---
title: Test
author: Author Name
header-includes:
    - \usepackage{fancyhdr}
    - \pagestyle{fancy}
    - \fancyhead[CO,CE]{This is fancy}
    - \fancyfoot[CO,CE]{So is this}
    - \fancyfoot[LE,RO]{\thepage}
abstract: This is a pandoc test . . .
---
```

documentclass document class for LaTeX documents

classoption option for LaTeX documentclass, e.g. oneside; may be repeated for multiple options

geometry options for LaTeX geometry class, e.g. margin=1in; may be repeated for multiple options

linestretch adjusts line spacing (requires the setspace package)

fontfamily font package to use for LaTeX documents (with pdflatex): TeXLive has bookman (Bookman), utopia or fourier (Utopia), fouriernc (New Century Schoolbook), times or txfonts (Times), mathpazo or pxfonts or mathpple (Palatino), libertine (Linux Libertine), arev (Arev Sans), and the default lmodern, among others.

mainfont, sansfont, monofont, mathfont fonts for LaTeX documents (works only with xelatex and lualatex)

colortheme colortheme for LaTeX beamer documents

fonttheme fonttheme for LaTeX beamer documents

linkcolor color for internal links in LaTeX documents (red, green, magenta, cyan, blue, black)

urlcolor color for external links in LaTeX documents

citecolor color for citation links in LaTeX documents

links-as-notes causes links to be printed as footnotes in LaTeX documents

toc include table of contents in LaTeX documents

toc-depth level of section to include in table of contents in LaTeX documents

lof include list of figures in LaTeX documents

lot include list of tables in LaTeX documents

biblio-style bibliography style in LaTeX, when used with –natbib

biblio-files bibliography files to use in LaTeX, with –natbib or –biblatex

Paper size, etc.

Geometry package options:

```
\usepackage[a4paper]{geometry}

\usepackage[paperwidth=5.5in, paperheight=8.5in]{geometry}

\usepackage[landscape]{geometry}
```

Note re: ebook sizing:

```
Those who want to read on tablets or other handheld digital devices need to create documents
```

Also:

```
The dimensions 90 mm× 120 mm are generally used, because the majority of eBook readers have
```

Also:

I'll give an example of the `\newgeometry` command. First, let's say you have an initial page layout defined in you preamble:

```
\usepackage[a4paper]{geometry}
```

Note that this is purely illustrative, you do not have to chose for a4paper.

Then you stumble upon a page of which you want to adjust the margins. In you LaTeX code, go to a line that is one that page and add the following:

```
\newgeometry{left=3cm,bottom=0.1cm}
```

Hence, the page will have a left margin of 3cm and a bottom margin of 0.1cm. The right and top margins remain unchanged.

In order to go back to the original (a4paper) layout, type:

```
\restoregeometry
```

This would be handy if I ever decided to try to lay out a photobook in LaTeX. *Note: That would be a very, very bad idea.*

Memoir options:

`ebook` and `landscape` are class options for memoir. It also accepts options like `twoside`, `oneside`, `onecolumn`, `twocolumn`, `openright`, `openleft`, `openany`, `article`, which semi-emulates the stock article class, and `ms`, which forces a totally typewriter-esque manuscript.

For temporary margin changes in `memoir`:

```
\begin{adjustwidth}{left}{right} text \end{adjustwidth}
\begin{adjustwidth*}{left}{right} text \end{adjustwidth*}
```

To increase the width of the text area, use positive values for `left` and `right`; to decrease it, use negative values. The starred version treats left as inner and right as outer margin respectively.

Some Font choices for Pandoc's `fontfamily` LaTeX option

See also LaTeX Fonts.

Example commands:

```
pandoc QueryGuide.md -f markdow_mmd
-o QueryGuide.libertine.md.pdf -V fontfamily=libertine

pandoc QueryGuide.md -f markdow_mmd
-o QueryGuide.tgscholatgherostgcursor.md.pdf
-V fontfamily=tgschola,tgheros,tgcursor
```

Some values that seem decent:

`libertine` Linux Libetine. Family includes serif, sans, and mono, and they look okay. Bit basic?

`ebgaramond` EB Garamond. (TUG, EB Garamond) I really, really like Garamond, but I have to concede that it is usually probably not ultra-appropriate for my uses. Doesn't come with a mono. Uses text figures, which is kind of neat, depending on application. Also: this doesn't seem to work for me in OSX unless I use XeLaTeX. This also requires setting the fonts using the independent `mainfont`, `monofont`, `sansfont` settings.

`bera` Another serif, sans, mono option. Looks okay, but the mono is a little bit space age. I think I like Droid better.

`droid` See above.

`fouriernc` A New Century Schoolbook type font. Does not come with a mono, and finding a mono that looks good with it is tricky. I've tried `tgcursor` and `libertineMono`.

TeX Gyre Fonts There are several. `tgschola`, `tgheros`, `tgcursor` and `tgbonum`, `tgheros`, `tgcursor` seems good. Schola is CNS-like, Bonum is bookman-like. Cursor is Courier-like. Compared to `fouriernc`, `tgschola` is a bit bigger and more spaced out.

`fbb` Bembo-like font. As with Garamond, it needs a good mono, which is tricky. Maybe `luximono`? Or `inconsolata`? One quibble: its dashes seem too heavy for its text.

`newtxtt` A thickish mono. Better than the default Latin Modern option.

Pandoc LaTeX Templates

Update: I'm still vague on template customization, but at least now I know where the templates *go*. The default location for installing the template on my OSX installation is (User folder)/.pandoc/templates. (The user data directory can be verified with –version.) (*Note: PDF/LaTeX templates are expected to end in .latex, apparently.*)

Regarding the templates themselves, here is the relevant bit of the user guide. Here are the default templates in GitHub. Here are some user-contributed templates.

So far, the only one I've tried is this one for Tufte-handout. In my templates folder it's named `--template=wcaleb-tufte-handout`. Here's the OSX service I'm using now:

```
PATH=$HOME/.cabal/bin:/usr/local/bin:/usr/texbin:$PATH

for file in "$@"
do
    cd $(dirname "$@")
    output=$@.pdf
    pandoc -f markdown_mmd "$file" -o "$output" --template=wcaleb-tufte-handout
done
```

One interesting thing about that template is that it uses symbols for footnotes instead of numbers. This seems cool, but may cause problems on longer documents. There's a possible useful thing at Stackexchange about this:

```
You may also want to reset the footnote counter for each page.
People are not familiar with the symbols beyond the double dagger,
and more than (at most) 16 footnotes in a single chapter will
result in an error message. To reset the counter per page, either
use the footmisc option perpage or the following code:

\usepackage{perpage}
\MakePerPage{footnote}
```

Dates

One thing I'd like to include in a latex template for Pandoc is date formatting. E.g., if I enter the date in a format that's intended to be friendly to Pelican (hyphenated year-month-date), I'd like that to be reformatted in a more reader-friendly way in a PDF.

Googling suggests there's a `datetime` package that should provide this functionality.

> Of course, it might happen that your desired format is not predefined. No worries! Just define your own format. As an example, say you only want the current month and current year to be printed (that would be September 2011). To do this, we will define a new date format called mydate:
>
> `\newdateformat{mydate}{\monthname[\THEMONTH] \THEYEAR}` Let's explain this piece of code. The `\newdateformat{}{}` is the command to define a new date format, which will be called mydate. Next, I want to print the current month in full and that what the `\monthname[]` command does. The command `\THEMONTH` produces an integer value of the current month: for September that would be 9. Similar, `\THEYEAR` produces an integer value of the current your (2011).
>
> How to actually change the date format Normally, you write `\today` (or `\date{...}`) for the date. With this package nothing much will change. Following the example of the previous section, say we've defined a date format called mydate. In your LaTeX document, so after `\begin{document}`, you need to write:
>
> `\mydate\today` In the more likely case you use `\maketitle` to print the date (together with the `\title` and `\author`), this is what you should do:
>
> ```
> \author{Author Name}
> \title{Title}
> \date{\mydate\today}
> \begin{document}
> \maketitle
> ```

From the package documentation:

> `\shortdate`

This declaration will redefine `\today` to produce the current date displayed in the form Wed 8th Mar, 2000 if the package option dayofweek is used, or 8th Mar, 2000 if the package option nodayofweek is used.

`\textdate`

This declaration will redefine `\today` to produce the current date displayed in the form: Wednesday the Eighth of March, Two Thousand if the package option dayofweek is used, or Eighth of March, Two Thousand if the package option nodayofweek is used. Note that `\textdate` is defined for use with English, it won't look right if it is used when another language has been selected . If you want to define a similar command for another language, you will first need to check that the fmtcount package supports that language.

You can also change the date separator from slashes to something else with `\renewcommand{\dateseparator}{}` which seems handy if you want to be a hipster and use periods or something.

Verbatim in Pandoc Templates

One of the things that bugs the heck out of me with the default Pandoc template is the lack of differentiation between verbatim blocks and the rest of the text by any means except serif vs. mono.

There are lots of *alternative* environments available via different packages, but for ease of use with Pandoc, I needed to redefine the "verbatim" command itself, which proved hard to google. But this seems to work, more or less:

```
%\usepackage[usenames,dvipsnames]{color}
%\usepackage{fancyvrb,relsize}
%
%\DefineVerbatimEnvironment{verbatim}{Verbatim}{frame=leftline,formatcom=\color{Sepia},frame
```

Note: Those aren't necessarily sane values; just parking them for later reference.

Default acceptable colors: `white, black, red, green, blue, cyan, magenta, yellow.`

Update: Okay, here's how to also catch the the material between backticks, which Pandoc puts into LaTeX as ‘

```
%\renewcommand\texttt[1]{{\ttfamily\color{Sepia}#1}}
```

And here's an approach to coloring `verbatim` that doesn't need `fancyvrb`.

```
%\usepackage[usenames,dvipsnames]{color}
%\makeatletter
%  \renewcommand\verbatim@font{\normalfont\ttfamily\color{Sepia}}
%\makeatother
```

With the `color` package you can use something like

```
\definecolor{Gray}{gray}{0.3}
```

or

```
\definecolor{orange}{RGB}{255,127,0}
```

to (re)define a color for later use.

So, here's an example for setting verbatim and texttt to a dark gray, in other words basically just lightening that text a little bit from the default black to let it be a bit more distinct:

```
\usepackage[usenames,dvipsnames]{color}
\definecolor{Gray}{gray}{0.3}
\usepackage{fancyvrb,relsize}

\DefineVerbatimEnvironment{verbatim}{Verbatim}{frame=leftline,formatcom=\color{Gr

\renewcommand\texttt[1]{{\ttfamily\color{Gray}#1}}
```

MMD and PDFs via LaTeX

Using MMD to produce LaTeX is *somewhat* more involved than doing so with Pandoc, for a few reasons:

1. It requires the installation of support files (templates)
2. You have to include 2 (or more?) kinds of template-related metadata
3. It's a two stage process; MMD doesn't produce the PDF for you, just the LaTeX file. You then feed that file to whatever LaTeX tool you use.[3]
4. There doesn't seem to be an obvious way to prevent LaTeX programs from generating a crazy number of extra files in addition to your desired output. And some of these you are apparently supposed to keep around? It's crazy.

The supplied MMD LaTeX templates are in some cases a little fiddly. For example, it's really cool that there's a template supplied for working with Tufte-LaTeX, but the way the it works with `tufte-handout`'s hilarious limitation of only two kinds of header is... weird. I guess you're expected to *always* set the base header level as 3 in the metadata? And if you use more/other header levels, it doesn't fail very gracefully.

In order to try to get better results with a document that has `#`, `##`, and `###`, I added this:

[3] This is of course one of those things that's not an issue for technical users, because scripting.

```
% NS: Trying to make MMD's output play nicer w/tufte.
% The issue appears to be that Markdown's tufte template header
% assignment appears to follow the memoir headers, i.e.,
% "Header levels are: h1 part, h2 chapter, h3 section, h4 subsection,
% h5 subsubsection, and h6 paragraph.", but tufte-handout doesn't
% seem to have been intended to use part, and it doesn't
% have chapter at all. Just section and subsection.

%\let\part\section
%\let\chapter\subsection
%\let\section\newthought
```

to the `mmd-tufte-handout-header.tex` file. It seems to help, but it's not really a fix.[4] In order to really get anywhere with this, I will unfortunately need to learn something about LaTeX. Gross.

Kokoi

Kokoi is a command line watching/previewing tool? Seems to work?

A major limitation is ease of navigation on larger documents. Adding a table of contents helps (see below) but is not as useful as Marked's jump-to-last-edit.

Example usage

```
kokoi -c "pandoc -s --toc -f markdown_mmd"
```

```
kokoi -c "pandoc -s -S --toc -f markdown_mmd --smart" -t
../../SharedTemplates/kokoi-toc.html -s
```

Note: Pandoc apparently needs `-s` *in order for the TOC to actually generate.*

Example with bib file (Note: Pandoc citations don't work w/markdown_mmd):

```
kokoi -c "pandoc -s -S --toc -f markdown_mmd --biblio
../../SharedTemplates/kukkurovaca.bib" -t ../../SharedTemplates/kokoi-toc.html
-s
```

Example for using MMD w/CriticMarkup highlighting:

```
kokoi -c "multimarkdown --smart -a -r" -t ../SharedTemplates/kokoi-toc.html
-s
```

Not as good as using Marked, especially becuase it does *not* render CM comments, which are the main CM markup I use. But it does do highlighting and addition/deletion, which could be useful in some cases.

Also, I cannot so far find a way to use MMD in conjunction with a TOC, which is too bad.

Kokoi's template

It's helpful if the TOC is fixed on the side. Some quick and probably terrible CSS:

[4] Using newthought as a header isn't a great idea; I just used it because it was a class that wasn't otherwise touched by the template, and I don't know anything about LaTeX.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>$title$</title>
    $scripts$
    <style>
      /* Fixing TOC on left */
          #TOC {
              position: fixed;
              left: 0px;
              top: 0px;
              padding-right: 2em;
              width: 20em;

          }

          body {
             margin-left: 24em;
          }

      /* Links */

        #TOC a:link, #TOC a:visited {

          color: black;
          text-decoration: none;
        }
      /* Title and Author */

        .title {
          text-align: center;

        }

        .author {
          text-align: center;
          font-style: italic;
        }
    /* Pre */

        pre {
             white-space: pre-wrap;       /* css-3 */
             white-space: -moz-pre-wrap;  /* Mozilla, since 1999 */
             white-space: -pre-wrap;      /* Opera 4-6 */
             white-space: -o-pre-wrap;    /* Opera 7 */
             word-wrap: break-word;       /* Internet Explorer 5.5+ */
            }


    </style>
  </head>
  <body>
    $body$
  </body>
```

```
    </html>
```

This needs to be saved as part of a template passed to kokoi as `-t templatename.html`.

Seems to work tolerably well.

Of course, it's dumb to add custom styling directly to the html template like that, b/c kokoi only loads it when starting up. I should put the CSS in a separate file.

Markmon

Markmon is another realtime Markdown previewing tool, for which there is a Sublime package. It defaults to Pandoc, and **most importantly**, it will jump to the most recent edit, just like Marked does. This is extremely useful if you're jumping around a large file.

There are two drawbacks I've noticed so far to using Markmon+Sublime as a Windows alternative to Marked:

- On my machine, Markmon seems to run noticeably slow
- It only wants to preview files that have their Syntax set to "Markdown", while the must-have MarkdownEditing package also supports GFM and MultiMarkdown syntaxes. (I normally use MultiMarkdown)

Settings file

When I installed the package on my Windows copy of Sublime, the default settings file didn't come over. So, I just copied it from Github and pasted into the user settings file.

The file advises trying markmon.cmd on Windows, which I did find to be necessary.

I also made some modifications to the command (smart typography, MMD extensions, TOC).

```
{
//On Windows try "markmon.cmd" if you get errors.
//If markmon is not on your path you'll need to use a full path to it
"executable": "markmon.cmd",
"port": 3002,
//  Original pandoc command:
//   "command": "pandoc -t HTML5 --mathjax",
//  Pandoc from MMD w/smart typography
    "command": "pandoc -S -f markdown_mmd -t HTML5 --mathjax",
//  Pandoc from MMD w/ smart typography and table of contents
//   "command": "pandoc -S -s --toc -f markdown_mmd -t HTML5 --mathjax",
"stylesheet": null,
"projectdir": null
}
```

I haven't tried using stylesheets with Markmon yet. I also have played around with the projectdir setting. The Github page says it's for the "Root directory of your project, useful for local image resources"

System Tools & Utilities

Textexpander

I used to think that Textexpander was a tool for people who just never learned to type quickly. I still sort of think that, but I'm coming around. Gradually.

Here's a snippet for creating an MMD metadata block:

```
Title: %filltext:name=Title:default=Title%
Author: %fillpopup:name=Author:default=kukkurovaca:Nick%
Date: %Y-%m-%d
Keywords: %filltext:name=Keywords:default=MMD-spec metadata%
Base Header Level: %fillpopup:name=Base Header Level:1:2:default=3%
Tags:  %filltext:name=Keywords:default=Metadata for Pelican (Multiple)%
Category:  %filltext:name=Pelican Category:default=Category (Single) for Pelican%
CSS: %filltext:name=CSS:default=http-//plaintextadventure.com/
    style/white-antique-2.css%
latex input:  %fillpopup:name=latex input 1:mmd-article-header:
    mmd-beamer-header  :mmd-memoir-header:default=mmd-tufte-handout-header%
latex mode: %fillpopup:name=late mode:default=memoir:beamer%
latex input: %fillpopup:name=latex input 2: mmd-article-begin-doc:
    mmd-beamer-begin-doc  :mmd-memoir-begin-doc:default=mmd-tufte-handout-begin-d
latex footer: %fillpopup:name=latex footer:mmd-memoir-footer:
    mmd-beamer-footer  :default=mmd-tufte-footer%
Todo: %filltext:name=Todo:default=Just a notes to self field%
```

Things about this:

- I set menu options for several fields where only a handful of values are used. This is particularly sensible for the LaTeX settings, which have long, arbitrary names that I find difficult to remember. - This could also be a good idea for CSS, if I had multiple stylesheets in common use for various documents. This is a good idea, I just don't at present. - Some settings here are for Pelican rather than for general MMD use + `Category` + `Tags`, which I've made identical to keywords + The format of the date

This seems like a lot of junk to the cram at the top of what is supposed to be a readable plain text file. And it is – anyone who doesn't need PDFs or who is satisfied with Pandoc's default PDFs (or is good enough to create their own Pandoc LaTeX templates) should stay far, far away from MMD's LaTeX settings.

But on the other hand, there is something to be said for defining as much as possible within the file itself. It can save a lot of time later on.

Markdown Quicklook Generator

{>>TODO<<}

- Current version somewhere on this page: http://multimarkdown.com/download/
- Copy the file to ~/Library/QuickLook
  - [ ] Need to add a little text

OSX Services

{>>TODO<<}

- Brett Terpstra's Markdown Service Tools
- https://github.com/dsanson/Pandoc-Droplets-and-Services
  - Need to update with my modifications (if any) as well as more specifics on what I'm using

At some point, Pandoc changed how it handled relative paths, which *totally* borked these services for me.[5]

I don't know bash scripting, but some furious googling enabled me to fix them. Here's an example:'

PATH=$HOME/.cabal/bin:/usr/local/bin:/usr/texbin:$PATH

```
for file in "$@"

do
    cd $(dirname "$@")
    output=$@.pdf
    pandoc -f markdown_mmd "$file" -o "$output"
done
```

The `cd $(dirname "$@")` is the operative bit.

Windows whatever the equivalent of sevices is

{>>TODO<<}

- http://www.terminally-incoherent.com/blog/wp-content/uploads/2012/05/markdown-menus.zip
  - [ ] Need to update with my modified version, as well as the source blog post

Modified just to add `-f markdown_mmd`:

```
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\*\shell\mkd2doc]
[HKEY_CLASSES_ROOT\*\shell\mkd2doc\command]
@="\"C:\\Users\\nshere\\AppData\\Local\\Pandoc\\pandoc.exe\"
-s -S \"%1\" # -f markdown_mmd # -o \"%1.docx\""

[HKEY_CLASSES_ROOT\*\shell\mkd2pdf]
[HKEY_CLASSES_ROOT\*\shell\mkd2pdf\command]
@="\"C:\\Users\\nshere\\AppData\\Local\\Pandoc\\pandoc.exe\"
\"%1\" -f markdown_mmd -o \"%1.pdf\""
```

[5] If you only ever link to stuff on the web, this probably wouldn't be an issue. But if you want to use image files stored in the same directory as your document or a subdirectory, then using the full file path is (a) a PITA and (b) a serious problem for working in, say, Dropbox, where your absolute path may be totally different on the various machines you're syncing.

Editors / etc.

MultiMarkdown Composer

This is my current favor Markdown editor.

- It (of course) plays nicely with MMD
- It also plays nicely with CriticMarkup
- It's the only Markdown editor I've used yet that does a really good job of synchronized scrolling of the source and the preview. Usually this feature is nigh-useless if you use more than a couple images, or if text size in your preview varies much from the source, or if you work on long documents. But MMD Composer gets this right. It's a little spooky.
- Many Markdown editors keep track of your TOC, so you can easily navigate between sections by header. But MMD composer also has a change panel (for your CriticMarkup notes) and a reference panel for footnotes, etc. That last is super-useful if you're like me in tending to accidentally re-use footnote IDs when writing Markdown.
- It does stylesheets and PDF output. Although I don't use these features too much, because I prefer to use Marked for previewing/export.

Byword

Byword is much more attractive, elegant, and simple than MMD Composer. It's not as powerful, but it does still play (mostly) nice with MultiMarkdown, and it does do PDFs, although it does not do stylesheets. The results are decent, really, but the lack of control is irksome.

Byword's iOS app is also very good, and unlike any other iOS Markdown editor I've found, Byword is capable of producing well-formatted PDFs from Markdown files in iOS. I actually used this feature pretty extensively at work, when I didn't have my laptop with me – before I decided it was worth it to go ahead and install pandoc and MMD on my Windows machine.

MarkdownPad 2

This is a Markdown editor on Windows. It is *halfway* decent. There are basically no other halfway decent Markdown editors on Windows.

Of course, do you really need a Markdown editor to write in Markdown? Goodness no. But sometimes it is nice to have a live preview of your document, or syntax highlighting, or Markdown-relevant shortcut keys. If you don't care about that stuff, any old text editor will work just fine.

Anyway, if you want a Markdown editor on Windows, this is one. It doesn't do much in the way of Markdown variants – the free version only does original recipe, and if you pay up, it still only does, I think, Markdown Extra and Github. Paying also unlocks the ability to set stylesheets and do PDF output, but (hilariously) the two features have nothing to do with each other, and the PDF output is *awful*, so why even bother?

Marked 2

It's pretty hard to explain the appeal of Marked to anyone who isn't already pretty far down the Markdown rabbit hole. It's not a Markdown editor – it's just a tool for previewing and exporting Markdown.

Why is that better than just using MMD Composer, which has good preview and okay export options already? Well, Marked is really, really, really polished. It does lots of neat little things, and the new version also does printing/PDF output really well, which is surprisingly uncommon. Most Markdown editors on OSX (and the original version of Marked) print the same way Safari does, which is to say, poorly, especially with regard to margins and pagination. Marked 2 does its own printing thing (not sure on the details), it actually works, and you can exercise a demi-decent degree of control using CSS.

Plus, if you use or may use a lot of different text editors, it's really handy to standardize on one previewing tool for all of them.

Marked Stylesheets

Marked comes with several stylesheets, but I actually find all of them somewhat problematic. In several cases this is just a matter of me not liking to look at documents with colored backgrounds, and in some cases, it's purely a matter of taste.

- Swiss is well-done, but all that sans serif clumped together is *really* not my cup of tea.
- I like Upstanding Citizen okay, but there's like a whole font or something embedded in that CSS file, and for large documents, that creates impressive performance problems.
- Manuscript is actually great, but it's specialized. It's not a look that's appropriate for most kinds of document.
- Antique is almost something I'd use, but it's an obnoxious color, and it attaches tildes to the headers for some reason. (Maybe it's an Icarus Proudbottom Teachers Typing reference.)

But, fortunately, you can add your own stylesheet or modify the default ones.

Warning: I don't really know all that much about CSS, and what I do know, I mostly forgot.

My solution for now is to adjust Antique to remove the sepia tone and make the headers more sane.

Also, so far as I can tell, all of the Marked templates need to be tweaked a little bit for best results if you want to print or generate paginated PDFs.

Some stuff that's in my modified copy of Antique:

Avoid page breaks after headers:

```
/* Page break? -Nick Also color */
h1, h2, h3, h4, h5, h6 {
page-break-after: avoid;
color:#000;
    }
```

Don't avoid page breaks inside blockquotes (I don't know about you, but I quote some pretty long passages sometimes.)

```
img, pre, /*blockquote, commented out -Nick*/ table, figure {
page-break-inside: avoid;
}
```

And take a shot at widows and orphans. (CSS isn't great for this yet, but it's better than nothing, and whatever Marked 2 is using to print does seem to sort of respect these values.

```
/* Nick */
p, li, blockquote {
    orphans: 2;
    widows: 2;
}
```

Also, if you end up using the spreadsheet to preview on anything other than Marked—for example, to keep the previews in your Windows workflow looking something like what you see in Marked—then it may be a good idea to set something re: max widths.

```
body {

    max-width: 40em;
    margin: 15em;
}

img {

    max-width: 700px;
}
```

Misc. Marked Notes

Marked doesn't seem to like it if you put latex input in the first MMD header,[6] which is interesting. It seems to prefer leading off with Title.

TOC

To display a TOC in Marked, you seemingly have to insert `<!-- TOC -->`

Note: I may want to duplicate what I did with the kokoi template…

Blogging

Octopress

Links

- Octopress

[6] I'm not sure if there's any special reason to do this. It's the way the metadata fields are ordered in all of the official MMD LaTeX/etc. examples. Happily, Putting Title first doesn't seem to create problems in the LaTeX output.

- Semi-useful link re: using Pandoc with Octopress
- Deploying Octopress to Github Pages
- Octopress Themes
- Koenigspress, a really lovely text-oriented theme
- Rake tips

Using Pandoc with Octopress

To use Pandoc/MMD, add a file `pandoc.rb` to the Octopress plugins directory like so:

```
require 'open3'
module Jekyll

# Just return html5
class MarkdownConverter
  def convert(content)
    flags = @config['pandoc']['flags']
    output = ''
    Open3::popen3("pandoc -t html5 #{flags}") do |stdin, stdout, stderr|
      stdin.puts content
      stdin.close
      output = stdout.read.strip
    end
    output
    end
  end
end
```

Source: here

Then modify `config.yml` like so:

```
markdown: pandoc
pandoc:
    skip: false
    flags: '-markdown_mmd --smart'
```

LaTeX Notes

Since I'm new to LaTeX, this section is going to be a mess.

Reference links:

- http://tobi.oetiker.ch/lshort/lshort.pdf

Devanagari

I'm currently using xetex-itrans while experimenting on my old fan translation of the MMK ch. 18.

Here's the preamble:

```
% xetex

\usepackage{ifxetex}
\RequireXeTeX
\usepackage{xltxtra}

% Converting footnotes to endnotes, for better page layout

\usepackage{endnotes}
\let\footnote=\endnote

% Dunno what this does -- it's from the xetex-itrans readme

\setlength{\parindent}{0mm}

% Here are the snippets from the xetex-itrans readme that define
% the commands for Devanagari and Romanization; I've named them
% dev and rom, creatively. The fonts were downloaded from the
% Itranslator website, and are not part of any TeX distribution.

\newcommand\dev{\catcode`\^=11
           \catcode`\~=11
           \fontspec[Script=Devanagari,Mapping=itrans-dvn]{Sanskrit 2003}}

\newcommand\rom{\catcode`\^=11
           \catcode`\~=11
           \fontspec[Script=Latin,Mapping=itrans-iast]{URW Palladio ITU}}
```

And here's a template for creating each verse:

```
%
% % Note that section tags don't have to be closed
% \section{}
%
% % samepage needs to begin and end, and within it,
% % \nopagebreak on each non-breaking line.
% \begin{samepage}
%
% {\dev
% Line 1 \\
% Line 2 \\
% }
%
% \nopagebreak
%
% {\rom
% Line 1 \\
```

```
% Line 2 \\
% }
%
% \nopagebreak
%
% Translation
%
% \nopagebreak
%
% Translation\footnote{}
%
% \end{samepage}
```

Misc. LaTeX Notes

- Command `\pagestyle{headings}` will add the chapter or section to the top of each page. `\thispagestyle{style}` will change the pagestyle of just the current page.
- This seems like it might be handy: `%\usepackage[parfill]{parskip}`
  `% Activate to begin paragraphs with an empty line`
  `rather than an indent`
- `\include{filename}` inserts a file *after a page break*
- Spaces between lines represent paragraphs, which LaTeX apparently takes seriously as a semantic construct, which is an amusing idea. You can force line breaks with `\\` or `\newline` when you need to break up lines with in a "paragraph."
- Aha! `\\*` will create a new line without starting a new paragraph, *and* it will prohibit a page break after it. (or not)
- `\linebreak[n]`, `\nolinebreak[n]`, `\pagebreak[n]`, `\nopagebreak[n]` take values of 0-4. Below 4 gives LaTeX the ability to override your preference.
- "`\sloppy` command. It prevents such over-long lines by increasing the inter-word spacing—even if the final output is not optimal. In this case a warning ("underfull hbox") is given to the user. In most such cases the result doesn't look very good. The command `\fussy` brings LATEX back to its default behaviour." Via
- `\documentclass[draft]` will mark in the margin lines about which it is giving warnings.
- `\mbox{whatever}` keeps whatever together no matter what. `\fbox{whatever}` also puts a big box around it
- Oh, right you have to use double back ticks and apostraphes instead of " marks. So obnoxious.
- Double `--` for ranges & en dashes, triple `---` for em
- `\ldots for ellipsis`
- Use the asterisk everytime you want to remove the numbering in the different parts in LaTeX (e.g., `\section*{}`)

- To suppress the word “abstract” before an abstract, use
  `\renewcommand{\abstractname}{\vspace{-\baselineskip}}`

For changing from indented paragraphs to non-indented paragraphs separated by linespacing:

% %

> The plus and minus parts of the length above tell TEX that it can compress and expand the inter-paragraph skip by the amount specified, if this is necessary to properly fit the paragraphs onto the page.
>
> this also has its effect on the table of contents. Its lines get spaced more loosely now as well. To avoid this, you might want to move the two commands from the preamble into your document to some place below the command `\tableofcontents`

Document class options

`twoside`, `oneside` – what you’d think.

`landscape`

`openright`, `openany`

Sections

- Chapters are only on `report` and `book`.
- Part is available for situations where you don’t want to impact numbering of sections. or just use an asterisk, as in

  ## Title

- `\appendix` changes chapter numbering to letters
- `\tableofcontents` to insert…something. Have to compile 2 or more times, b/c LaTeX is fucking stupid.
- Referencing: `\label{marker}`, `\ref{marker}`, and `\pageref{marker}`

Environments

Lists

```
\flushleft
\begin{enumerate} % or itemize or description
\item You can nest the list
environments to your taste:
\begin{itemize}
\item But it might start to
look silly.
\item[-] With a dash.
\end{itemize}
\item Therefore remember:
\begin{description}
```

```
\item[Stupid] things will not
become smart because they are
in a list.
\item[Smart] things, though,
can be presented beautifully
in a list.
\end{description}
\end{enumerate}
```

Quotation / Verse

As you'd expect.

Verbatim

Either as an environment or as a command, w/arbitrary characters wrapping the argument?

Floats

`\begin{figure}[placement specifier]` or `\begin{table}[. . . ]`

"A placement specifier is constructed by building a string of float-placing permissions" Well duh.

> `\begin{table}[!hbp]`
>
> The placement specifier [!hbp] allows LATEX to place the table right here (h) or at the bottom (b) of some page or on a special floats page (p), and all this even if it does not look that good (!). If no placement specifier is given, the standard classes assume [tbp].
>
> a figure that cannot be placed pushes all further figures to the end of the document

h : here t : at the top of some page b : at the bottom of some page p : on a floats-only page ! : Overriding "internal parameters"

Floats are queued, and an unplaceable float will block later floats from being placed.

`\caption{caption text}` adds captions. `\caption[Short]{LLLLLooooooonnnnngggggg}` provides a short version of the caption for the listings.

`\listoffigures` and `\listoftables` generate those things

`\label` and `\ref` are used to refer to figures from the text. Labels go after captions.

```
Figure~\ref{white} is an example of Pop-Art.
\begin{figure}[!hbtp]
\makebox[\textwidth]{\framebox[5cm]{\rule{0pt}{5cm}}}
\caption{Five by Five in Centimetres.\label{white}}
\end{figure}
```

`\clearpage \cleardoublepage` will force the creation of float pages, I guess?

When putting a footnote in in the argument of a section-like(?) command, put `\protect` in front of it.

Bibliography

Bib entries go in environment `thebibliography`

```
\bibitem[label]{marker}
```

The marker is referenced with `\cite{marker}`.

Index

Both these go in the preamble:

`\usepackage{makeidx} \makeindex`

Index entries are specified as:

`\index{key@formatted_entry}`

```
Example                    Index Entry    Comment
\index{hello}              hello,         1 Plain entry
\index{hello!Peter}        Peter,         3 Subentry under 'hello'
\index{Sam@\textsl{Sam}}   Sam, 2         Formatted entry
\index{Lin@\textbf{Lin}}   Lin, 7         Formatted entry
\index{Kaese@K\"ase}       Käse, 33       Formatted entry
\index{ecole@\'ecole}      école, 4       Formatted entry
\index{Jenny|textbf}       Jenny, 3       Formatted page number
\index{Joe|textit}         Joe, 5         Formatted page number
```

use `\printindex` to actually insert the index in the body

Fancy Headers

I'm going to skip this for now. It's section 4.4 here.

LaTeX Fonts

- `fontenc` packages lets you choose fonts, I guess?
- `fontspec` "handles fond loading" for XeLaTeX
- Option `Ligatures=TeX` will make stuff like em/en dashes work
- `\newfontfamily\russianfont[Script=Cyrillic,(...)]{(font)}` defines font choices for languages

Inline font commands:

```
\textrm{...} roman        \textsf{...} sans serif
\texttt{...} typewriter
\textmd{...} medium       \textbf{...} bold face
\textup{...} upright      \textit{...} italic
\textsl{...} slanted      \textsc{...} Small Caps
\emph{...} emphasized     \textnormal{...} document font
```

Latin Modern

Supposedly you just need to add this to the preamble:

```
\usepackage{lmodern}
\usepackage[T1]{fontenc}
\usepackage{textcomp}
```

PDF-friendly Postscript fonts

```
\usepackage[T1]{fontenc}
\usepackage{pxfonts}
```

for Palatino-esque or

```
\usepackage[T1]{fontenc}
\usepackage{txfonts}
```

for Times-esque.

PDF links

> The hyperref package will take care of turning all internal references of your document into hyperlinks. For this to work properly some magic is necessary, so you have to put `\usepackage[pdftex]`
>
> so you have to put `\usepackage[pdftex]{hyperref}` as the **last** command into the preamble of your document

Options include:

```
colorlinks (=false,true)

linkcolor (=red) colour of internal links (sections, pages, etc.)
citecolor (=green) colour of citation links (bibliography)
filecolor (=magenta) colour of file links
urlcolor (=cyan) colour of URL links (mail, web)
```

For colored links with default value:

```
 \usepackage[pdftex,colorlinks]{hyperref}
```

Colored links may not print well, so for printable best results, use those stupid colored frames (which are omitted when printing):

```
\usepackage{hyperref}
\hypersetup{colorlinks=false}
```

Or black links:

```
\usepackage{hyperref}
\hypersetup{colorlinks,%
            citecolor=black,%
            filecolor=black,%
            linkcolor=black,%
            urlcolor=black,%
            pdftex}
```

If a `destination with the same identifier` error occurs, use `plainpages=false` in hyperref options.

Footnotes

```
If the text within the footnote is very long, LaTeX may split the footnote over s

\interfootnotelinepenalty=10000
```

XeLaTex

Here's the preamble of TexShop's XeLaTeX "stationery":

```
\documentclass[12pt]{article}
\usepackage{geometry}
    % See geometry.pdf to learn the layout options. There are lots.
\geometry{letterpaper}
    % ... or a4paper or a5paper or ...
%\geometry{landscape}
    % Activate for for rotated page geometry
%\usepackage[parfill]{parskip}    %
     Activate to begin paragraphs with an empty line rather than an indent
\usepackage{graphicx}
\usepackage{amssymb}

% Will Robertson's fontspec.sty can be used to simplify font choices.
% To experiment, open /Applications/Font Book
% to examine the fonts provided on Mac OS X,
% and change "Hoefler Text" to any of these choices.

\usepackage{fontspec,xltxtra,xunicode}
\defaultfontfeatures{Mapping=tex-text}
\setromanfont[Mapping=tex-text]{Courier Prime}
\setsansfont[Scale=MatchLowercase,Mapping=tex-text]{Gill Sans}
\setmonofont[Scale=MatchLowercase]{Andale Mono}
```

(Oh, but I changed the font to Courier Prime. It's growing on me.)

Languages in XeLaTeX

```
\setdefaultlanguage{english}
\setotherlanguage[babelshorthands]{german}


Englisch text.
\begin{german}
```

```
Deutscher Text.
\end{german}
More English text

Englisch text. \textgerman{Gesundheit} is actually a German word.
```

XeLaTeX Fonts

To use weird ligatures:

```
\setmainfont[Ligatures=Rare]{(font)}
\setmainfont[Ligatures=Historic]{(font)}
\setmainfont[Ligatures=Historic,Ligature=Rare]{(font)}
```

You may have to set the language:

```
\setmainfont[Language=Polish]{(font)}
```

To install opentype fonts that are already included with TeXLive, go to (in my case):

```
usr/local/texlive/2014/texmf-dist/fonts/opentype
```

Random list of pleasant fonts available

In TeXLive

- fbb
- CMU Serif & friends
- Heuristica
- Linux Libertine O
- Quattrocento
- Raleway
- TeX Gyre Pagella
- TeX Gyre Schola
- TeX Gyre Termes

Other

- Courier Prime
- Hoefler
- Optima
- Goudy Bookletter 1911

Fontspec

```
\usepackage{fontspec}

\setmainfont{TeX Gyre Bonum}
\setsansfont{Latin Modern Sans}[Scale=MatchLowercase]
\setmonofont{Inconsolata}[Scale=MatchLowercase]
```

Fonts can be set like commands, either on a family basis or on a face basis. (E.g., just italic, or whatever.

```
\newfontfamily\notefont{Kurier}
\notefont This is a \emph{note}.

\newfontface\fancy{Hoefler Text Italic}%
[Contextuals={WordInitial,WordFinal}]
\fancy where is all the vegemite
% \emph, \textbf, etc., all don't work
```

Font variants can be specified directly, for example, if a font family has multiple italic options or whatnot.

```
BoldFont = <font name>
ItalicFont = <font name>
BoldItalicFont = <font name>
SlantedFont = <font name>
BoldSlantedFont = <font name>
SmallCapsFont = <font name>
```

An asterisk can be used in place of the family name:

```
 \setmainfont{Baskerville}[BoldFont={* SemiBold}]
```

> As a matter of fact, this feature can also be used for the upright font too:

```
\setmainfont{Baskerville}[UprightFont={* SemiBold},BoldFont={* Bold}]
```

New commands & environments

Example of a command that takes arguments:

```
\newcommand{\txsit}[2]
{This is the \emph{#1}
#2 Introduction to \LaTeXe}
% in the document body:
\begin{itemize}
\item \txsit{not so}{short}
\item \txsit{very}{long}
\end{itemize}
```

`\renewcommand` will specify a new command that overrides an existing command.

New environments can also be specified: `\newenvironment{name}[num]{before}{after}`

```
\newenvironment{king}
{\rule{1ex}{1ex}%
\hspace{\stretch{1}}}
{\hspace{\stretch{1}}%
\rule{1ex}{1ex}}
\begin{king}
My humble subjects \ldots
\end{king}
```

Ifthen

here's an example of a conditional for use with the command line. Not my use case, but I could see it being useful in some cases:

```
  \usepackage{ifthen}
  \ifthenelse{\equal{\blackandwhite}{true}}{
  % "black and white" mode; do something..
  }{
  % "color" mode; do something different..
  }
```

```
Now call LATEX like this:

  latex '\newcommand{\blackandwhite}{true}\input{test.tex}'
```

Memoir Class notes

Manual:

- 1.2.1 / p. 3 Extended font sizes imply Latin Modern options
- 1.3 / p. 4 Printing options, including manuscript
- 1.4 / p. 5 article option

  > "simulates the article class, but the \chapter command is not disabled. Chapters do not start a new page and chapter headings are typeset like a section heading. The numbering of figures, etc., is continuous and not per chapter. However, a `\part` command still puts its heading on a page by itself.

- 1.5 / p.5 Default:

  > Calling the class with no options is equivalent to:
  >
  > `\documentclass[letterpaper,10pt,twoside,onecolumn,openright,final]{memoir}`
  >
  > The source file for this manual starts
  >
  > `\documentclass[letterpaper,10pt,extrafontsizes]{memoir}`

- "If you use the `ebook` option you may well also want to use the 12pt and oneside options."
- Font samples on p. 36 on

Note: eventually I mostly stopped taking notes b/c it's 600+ goddamn pages.

- Wrapped verbatim:
  `\wrappingon` `\wrappingoff` `\verbatimindent`

Ultramiscellaneous

Meeting notes boilerplate:

```
Title: Title of meeting
Date: January 1, 2014
Base Header Level: 2
Keywords: Whatever

# Meeting Notes Template Meeting: January 1, 2014

Something about the meeting topic, if necessary.

Attending:

* Name
* Name

## Preparatory Notes/Agenda

* Item
* Item

<!-- ## Major Concerns: -->

<!-- ## Tasks: -->

<!-- ## Notes: -->
```