MARKDOWN

[TO](ebook://)

Author
: K-2052

Markdown To Ebook

K-2052

This book is for sale at http://leanpub.com/markdown-to-ebook

This version was published on 2013-12-16

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

An Introduction

Writing a book can be daunting, learning the tools to write a book can be even more daunting. A quick google search will turn up a multitude of services, tools, apps and frameworks designed to write a book. Apps like; iBooks, Adobe Acrobat, inDesign Skyreader, Kindle Publisher etc. Formats like; PDF, ePUB, LaTeX….

It's not so far fetched that a writer when faced with this mess of "solutions" might simply choose a trusty old typewriter or even pen and paper. Or he might get so caught up writing tools for writing that he never gets the time to write.

This book is a result of my search for a get out of the way method of writing ebooks. Something that keeps my focus on the writing and not on the process of writing. It doesn't cover everything, it gives you just enough knowledge so you can focus on what matters, the writing.

This book follows the hackers ethos of not obscuring things. Problem areas are not glossed over, they are put right out in the open, so you can learn from them. Where I struggled you will know I did and why I did. I wrote this book through countless frustrations with Markdown parsers, LaTeX packages and varying bugs; to ignore that would skip a crucial part of how I learned what I know.

Most technical books appear smooth when read, but they were not written smoothly. To get that snippet down to something slim and slick requires a lot of breaking things and figuring out what doesn't work. Making things look easy is hard work. I tried not to make things look easier than they are. By showing you my failures I hope to help you avoid them.

This is not a book for those that like quick summaries and step by step processes. It's a book for those that like to learn something. The only way to eliminate frustration is to internalize your knowledge, to make it so much a part of you that it cant be ignored.

Unless you truly know how to do something then doing it is going to get in the way and take energy away from your writing. You cant for instance, be looking up how to create a section every time you want to create a section. Such a workflow will kill a book before it is even start. Things have to flow smoothly, without thought, so you can just write.

With this in mind, I have kept the approach to tools as simple as possible, so they can be internalized as quickly as possible. Our book' source files will be written in Markdown. Markdown is simple enough to internalize in a day and most importantly, it is flexible enough that it can be adapted to support a variety of extras.

Kramdown has been chosen as our flavor/implementation of Markdown. This is for a few reasons;

1. It is used in a variety of publishing platforms (specifically Leanpub[1])

[1] https://leanpub.com

2. It closely resembles many other implementations like PHP Markdown Extended.
3. It's well documented.
4. It's coded in Ruby. Tools like pandoc are just as powerful as Kramdown but pandoc is unfortunately written in Haskell. Haskell would only increase the learning curve and add more annoying parts to getting our ebook published.

What you need to know

You wont need to know much to get through this book but you will need to be comfortable hacking around. Comfortable is a dangerous word for communication, one's interpretation can vary greatly from what is intended. Let's go over some examples to help clarify what I mean by *comfortable*

Can you do this?

```
1 homebrew install
```

Or if you are on Linux this?

```
1 sudo apt-get install
```

If this leaves you scratching your head then you are not ready for this book. If you are left feeling comfortable and at ease, not queasy and fearful, then you are ready.

Some rudimentary ability to read and edit Ruby will also be necessary. If you are comfortable enough to edit Gemfiles and install gems using bundler then you should be fine.

We are going to need a workflow

I'm pulled in a thousand different directions by a thousand different solutions. iBooks Author looks nice. Nathan Barry went with that right? I think that is also how Josh Long wrote Execute. But what about epub? Isn't epub just html at its core? Maybe I'll write in HTML and use an HTML to PDF converter? That would allow me to craft all sorts of customizations in CSS. I could really style things awesomely. I could release a web version with fancy JS stuff and blow people's minds with a really unique ebook release.

But what about organization of my writings – writing stuff in pure HTML is going to be a mess. I guess I could use a static site generator and write in a templating language to keeps things organized into chapters. But maybe I should use a ebook authoring tool that handles this for me? What about Scrivener? Don't screenplay authors use that? What if I want to deliver a screenplay for my book in case they decide to adapt into a movie?

Choosing a workflow from a place of ignorance is an exercise in futility. Any choice we make will be clouded by a lack of knowledge, too many options and possibly delusions of grandeur.

If we want to find a workflow that works we are going to have start with very basic questions. Questions we can see the answers to immediately. If we answer the simple questions first then the complex ones will become clearer. Answering some basic questions will give us constraints and illuminate the path ahead.

Where will we distribute the book?

For distribution it would be nice to have something that works well for the writing phase as well as the publishing phase. Perhaps even something that generates our ebooks from our source formats for us. Such a service exists and it's called Leanpub[2].

What does Leanpub use as its source format? Leanpub uses Markdown. Thus, we will need to use Markdown; we now have our first constraint.

What final format does it need to be in when distributed?

There are a zillion different options for ebook formats so whatever we choose it's going to have to be capable of converting into a variety of formats. We will need something that can go into PDF,

[2] https://leanpub.com

epub, kindle and mobi formats. It just so happens that with the exception of PDF all these formats are HTML at there core.

What is a good option for writing HTML ebooks? Markdown of course. Markdown is simple, flexible, looks great as plaintext and most importantly, it is built with HTML as the intended output format.

How then do we turn Markdown into PDFs? The answer lies in LaTeX, which will be our in-between format of choice. LaTeX is to PDF as Markdown is to HTML. They're best buds and get together all the time to just chat and mark things up.

Does LaTeX mean we will have to maintain two separate formats? Nope. Markdown can be easily converted into LaTeX using tools like Kramdown.

LaTeX

LaTeX is one of those things you probably know about but have never bothered learning. I mean, it's for technical papers right? When have you ever needed to write a technical paper. One day when you have the time, you plan to study it but not now. For now, you'll just use Markdown. Markdown works for now, but one day you're going to have to touch LaTeX if you want to move beyond the HTML realm.

I could cover LaTeX the normal way; mention the apps you use to use it, the commands you use to install it, the syntax you use to craft it, and then send you on your way. Most books on LaTeX are written this way. I could tread the same ground other books do and I'm sure you'd get some benefit. After all, countless books cover things that way, it's not a completely useless approach.

You might even be motivated enough to play with LaTeX a bit. Perhaps you will even write a few a things in it. In the long run though, you'll slowly find its usefulness lacking. Markdown will just be quicker and more productive. When you write Markdown you'll get things done. When you use LaTeX you will feel like you're fighting it; spending more time on your tool than on writing. When you truly need the power of LaTeX you'll do what you always do, live without it or hack a solution onto your favorite Markdown parser.

I know you'll do this because I do it. We all do it. We learn new tech because we are curious, but if the logistics don't work out, if the practicalities of using it don't align with execution, actually getting stuff done, then we don't use it. It's the same reason we never blog on our amazing overly designed custom built blogging engines, it just isn't productive. One day a Medium, Tumblr or Jekyll comes along and we finally take the plunge and simplify enough to just focus on our writing. Things need to be too easy or they won't be made use of. Your tools need to blend into the background and get out of the way of creation or you'll never use them.

If we are actually going to make real use of LaTeX it's going to have to work for us, not us for it. It will have to complement our workflow, not supplement it. We will adopt a workflow that makes LaTeX work for us.

Leanpub

Let's visit Leanpub, get registered, and create a test book.

After we get signed up, Leanpub will inform of us the basics. We will upload our ebook files via Dropbox. They will be written in plain text using Markdown.

We will get a link sent to us for our first book. Get Dropbox setup if you haven't already.

You can utilize the web interface for Dropbox. No need to download/install.

Visit the getting started page for your book *leanpub.com/bookname/getting_started* e.g *https://leanpub.com/markdo*
to-ebook/getting_started

The first thing we can take away from this page is that we will use Markdown. If we incorporate fancy stuff it will be by customizing the conversion of our Markdown. I actually like this as it keeps our text straightforward and simple, we will only have to use convoluted LaTeX when absolutely necessary.

We need to know a little about the Markdown flavor we are dealing with though. After some quick googling I found this HN comment[3]:

```
1  Hey, co-founder of Leanpub here.
2
3  If you're interested in the differences between Pandoc and Leanpub, there are two\
4   places to look. Leanpub is based
5  off of Kramdown, so there's the Kramdown documentation[1]. We have made a few ext\
6  ensions to Kramdown to support
7  things that books need, so you'll also want to look at the Leanpub manual[2].
8
9  If you have any questions, send us an email at hello@leanpub.com.
10
11 [1]: http://kramdown.rubyforge.org/syntax.html [2]: https://leanpub.com/help/manu\
12 al
```

So, Leanpub uses Kramdown, this is good news. This makes incorporating Leanpub into our workflow even easier.

[3] https://news.ycombinator.com/item?id=4998144

Flavors of Markdown

You might think you know Markdown, but chances are you only know *a Markdown*. The Markdown you know might not be the Markdown you get to use. All Markdown implementations handle the basics but you might not be aware of what the basics [4] actually are. Automatic link parsing, for example, is not part of the standard, it's just something that everyone and their biological birthing entity implements. There are three major flavors of extended Markdown you are likely to encounter.

Github flavored Markdown

This implementation[5] has been made enormously popular by the ubiquity of Github in the development community. It has a few features not found in the original Markdown spec, namely; fenced code blocks and automatic link parsing. The de facto implementation of this flavor is redcarpet[6]

MultiMarkdown

Like many Markdown flavors, MultiMarkdown[7] originated as the de facto implementation of Markdown for a language; originally written for Perl, MultiMarkdown has since become a C library[8]. It is one of the most well documented flavors but it lacks a decent adoption rate/userbase.

Markdown Extra

This[9] variant came originally from PHP but has since been implemented in several languages. Markdown Extra has a few extra features like; footnotes, extensions, attribute lists etc.

We will be using an implementation of Markdown Extra called Kramdown. There are two reasons for this:

1. Leanpub[10] uses it.

[4] An official extension spec is being worked on but the details are pretty sparse. See The Future of Markdown and this commit
[5] http://github.github.com/github-flavored-markdown
[6] https://github.com/vmg/redcarpet
[7] http://fletcherpenney.net/multimarkdown
[8] https://github.com/jgm/peg-markdown
[9] http://michelf.ca/projects/php-markdown/extra
[10] https://leanpub.com

2. It is written in Ruby which is easy to hack on.

For the most part, the only two flavors you will have to deal with are Github's and Markdown Extra. The differences between the two aren't too significant. Most of what Github does is extra for Github and won't effect your needs for ebook writing. Utilizing Github features just adds stuff when your Markdown is used on Github and utilizing Kramdown features just adds stuff when using Kramdown.

The only difference you need to care about is the syntax for fenced code blocks. Github utilizes three ‘’s and Kramdown uses three ∼∼∼.

They look like:

Github

```
```ruby
require 'redcarpet'
markdown = Redcarpet.new("Hello World!")
puts markdown.to_html
```
```

Kramdown

```
~~~ ruby
require 'redcarpet'
markdown = Redcarpet.new("Hello World!")
puts markdown.to_html
~~~
```

Since you cant modify the parsing on Github, if you plan to publish the same Markdown on Github and need the code highlighting you'll need to modify Kramdown to use the same indicator as Github for starting and closing fenced code blocks.

> Alternatively you can just use indentation to specify your code blocks which is compatible across most implementations

Changing ∼ to a ‘ will get it parsing like Github. To do that just search for `FENCED_CODEBLOCK_START` in the Kramdown files. Monkey-patching is the simplest way to get the modification into Kramdown. In the interest of clarity, the following code snippet will accomplish said changes:

```
module Kramdown
  module Parser
    class Kramdown
      FENCED_CODEBLOCK_START = /^`{3,}/
      FENCED_CODEBLOCK_MATCH = /^(`{3,})\s*?(\w+)?\s*?\n(.*?)^\1`*\s*?\n/m
    end
  end
end
```

If you plan to use Leanpub you'll need to utilize indented codeblocks instead. You'll indent things using four spaces then manually specify your language using attribute lists. Like this:

```
This is a codeblock

{:lang="ruby"}
    cats = 'demons'
```

Beyond that, just follow the docs for Kramdown[11] and you'll be fine.

Learn your Markdown

You might be tempted to avoid learning Markdown and to add a Markdown GUI to your workflow. I urge you to resist that temptation. Jumping around in app menus for simple things like links and headings will kill your workflow. There is a reason why some writers from past eras still choose to use typewriters, being 100% comfortable with your tools is extremely important for productivity. You'll need to know your Markdown inside and out if you ever hope to finish a book.

Any app that attempts to replace the Markdown text process is going to interfere with your workflow in the long run. If you use Markdown apps only use them to complement your writing, with fancy previews and nifty publishing tools. Don't use an app as a crutch. Using crutches will make you disabled.

[11] http://kramdown.rubyforge.org/quickref.html