Krambook

Write Books like an UB3R 1337 *(Hacker)*

Yann Esposito

November 24, 2010

I Introduction

`krambook` provide you a *cool* way to write a book.

You use a markdown syntax and the book can then be generated:

- as PDF using XeLaTeX
- into a static HTML website
- into a website that display you PDF in SVG

1.1 Why this project?

Markdown is easier to read than LaTeX

The best typesetting system I know is LaTeX. Unfortunately LaTeXwas created a long time ago and its syntax is full of backslashes. Here is an example of a standard minimal LaTeXdocument:

```
\documenttype{article}
\usepackage[utf-8]{inputenc}
\usepackage{fontenc}
\usepackage{amsmath}

... % This is the ritual header

\begin{document} % ---- end of the preamble
\section{First section}
I begin by making a list of bullet points:
\begin{itemize}
\item the first point is
    \LaTeX is a bit verbose
\item the second point is
    \Latex has \textem{more} \textbackslash{} than Markdown
\item I believe you understood now.
\end{itemize}
\end{document}
```

To achieve a similar result using `markdown` syntax:

```
First section
=============

I begin by making a list of bullet points:

- the first point is LaTeX is a bit verbose
- the second point is LaTeX has _more_ \ than Markdown
- I believe you understood now
```

The HTML end result using the markdown will be:

> **First section**
>
> I begin by making a list of bullet points:
>
> - the first point is LaTeXis a bit verbose
> - the second point is LaTeXhas *more* \ than Markdown
> - I believe you understood now

Then I believe this example should be enough to convince you that the `markdown` is more natural than the LaTeXone.

Markdown does not scale

LaTeXhas many incredible properties that makes it scalable even for very long document. On the other hand Markdown wasn't created for this purpose. Markdown was done to provide a standard syntax to transform some text file into HTML. Markdown lack many features that many other project have added to it. One of this project is Kramdown. There is many other project that expanded the abilities of Markdown.

But I believe not any of these project is scalable because the power of these language is *stricly* inferior to the power of the TeX language. In fact TeX is Turing complete -- considering we have the ability to make many compilations until reaching a fixed point.

How can LaTeX be Turing complete? Simply with the power of provided by *macros*. In LaTeXyou can declare macros like this:

```
\newcommand{\un}{\sum_{n=0}^\infty u_n}
```

And each time you type:

```
Here is a formula $\un = \pi$
```

It will be equivalent to:

```
Here is a formula $\sum_{n=0}^\infty u_n = \pi$
```

Imagine a thesis where this formula is present a hundred times and you begin to understand why macros are a necessity for long documents. But in LaTeXyou could also declare macros with parameters and that use other declared macros:

```
\newcommand{\ratlang}[2]{\mathcal{S}_{#1}^{\mathrm{rat}}(#2)}
\newcommand{\sr}[2]{\ratlang{\mathbb{R}}(\Sigma)}
...
Let us denote $\sr$ the class of rationnal
stochastic language over $\mathbb{R}$ with alphabet $\Sigma$.
```

Now you see the power of LaTeX.

There is also another thing that make LaTeXscalable. You can include other source files. This make it easy to separate work and also to work with many other people.

Another good point with LaTeXand markdown is that you write only in text file and you can then version these file using `git` for example.

The purposes of this project are

- Handle long documents by:
  - adding macros to kramdown
  - working with many small and versionnable text files
- generate high-quality PDF *and* HTML documents.

For now, the power of this superset of kramdown syntax is *not* Turing complete. You can declare macros, but without any parameters and you cannot use already declared macros inside other macros declaration. But this simple addition to markdown is already powerful enough for most of usage.

2 Install

2.1 Prerequisite

If you are reading these lines chances are great that your system contains all necessary packages. But here are the dependencies:

- ruby,
- rake,
- `kramdown`[1]

Optionally you'll need:

- LaTeX(more precisely XeLaTeX) to generate PDF output,
- MathJax to draw correctly math formulae inside HTML website,
- `pdf2svg` to generate the SVG oriented website.

2.2 The steps

1. You'll need to install ruby and rake. They should be present on your system. But if you are using Ubuntu the following command line should be enough:
2. In order to install the `kramdown` gem:
3. To install XeLaTeX, I suggest you to use TexLive full install to be certain not lacking anything. Of course you are free to use any other distribution that suit you better.
4. Download MathJax
5. Finally Download the source code and copy the MathJax directory into `site/js/`.

Verify if all work correctly by running:

```
> rake
> rake html
```

Congratulation you are ready.

[1] `kramdown` is an amelioration of the original markdown format.

3 Tutorial

3.1 Firsts steps

I suppose you have a correct install of LaTeX.
If you had not yet verified try to launch the following:

```
> rake
```

It should create a `krambook.pdf` file.
Now it is time to create your own book:

- Edit the `config.rb` file (set title, author name and the pdf filename)
- Create and write files in the `content` folder.
  You should write them using the kramdown format. Very close to the

  Remark: by default file are sorted by name. I suggest you to name your files and folder with number prefixes. For example like `00_intro.md`, `01_section/01_subsection.md`, etc... You can make a bit of `ruby` (search `@filelist` in the `Rakefile` file) to change this behaviour.

- run `rake` (or `rake compile`) to create and show a `.pdf` file.
- run `rake html` and launch `unicorn` (`gem install unicorn`) then look at the website at `http://localhost:8080/`,
- run `rake clean` to remove temporary files,
- run `rake clobber` to remove all generated files

With just that you can already write a book. You can make as many as file as you want. Every file of the form: `content/**/*.md` will be used to create the book.

3.2 Macros

Now you can write the content of your book mostly in `kramdown` format. But with some simple additions: *macros*.

Remark: For now Krambook accept only macros *without* any parameter. Here are some examples:

These transformations will occur on the markdown file before it is transformed in LaTeX.

You can also declare macro that will be processed after the file was transformed in LaTeXor in HTML.

To use them simply write `%macroname` or `%code` and it will be transformed correctly in your pdf or HTML.

3.3 Some other macros examples

It is a simple demonstration of how macros are working. They were declared inside the markdown like this:

Now if I write:

```
%tldr A simple demonstration of how macros are working.
```

It renders as:

> *Too long don't read:* A simple demonstration of how macros are working.

The `%multine` macro render as:

```
a
multiline
macro
```

The output should be in LaTeX and was compiled from a markdown-like format.

- Simple list ;
- Example ;
- Another one item.

```
Hello there

this is some code block
```

LaTeX Some LaTeX definition

A simple math mode x_i and a protected one $$x_i$$ A long formula now:

$$\sum_{i=0}^{n} \sqrt{x_i + y_i}$$

Even with some ruby code inside:

Here is the result of the `%ruby` macro:

```
aaa
```

and a more complex one (`%complex`):

```
1 : 4 : 9 : 16 : 25
```