Official Riven Manual

Write Markdown. Get a nice PDF. Have fun.

Contents

1. What is Riven?

In one sentence: Riven is a command line tool, which converts a bunch of markdown files into a PDF file.

The question is: Why would you do that?

There are several scenarios where the usage of a tool like Riven would make a lot of sense for you. Let's take a look at some of them:

1.1. Scenario A: You want to write a book

You are a bit familiar with Linux or OSX. Maybe you even know what markdown is, but it's okay if you don't know markdown. And you want to write a book for some reason. Maybe a subject book, maybe a novel or maybe some fanfiction for whatever you like. But you have no idea of LaTeX and you don't want to write it in LibreOffice or some other kind of office suite for some reason. Maybe because you want to have full control over all formatings and you love a text editor like atom, sublime editor or vim. You also know HTML and CSS maybe, but you don't want to use it since it's to much code to write and you have to test to it too often in you browser while writing.

What you want is just a very simple and lean tool chain which let's you write a very minimalistic markup in your favorite text editor and compiles everything in a beautiful, readable and stylish PDF (and mobi, epub etc in the future). And if something doesn't fit your requirements, you just change it via CSS or configuration.

1.2. Scenario B: Your a coder and want to document your software

You have a software project, you're familiar with Linux or OSX, HTML, CSS, Markdown and others. And you want to document your software in a plain text readable format. But you also want a very simple and lean tool chain which compiles everything in a beautiful, readable and stylish PDF (and mobi, epub etc in the future). And if something doesn't fit your requirements, you just change it via CSS or configuration. Also you want to use your favorite text editor like atom, sublime editor or vim but don't want to write the documentation in HTML and CSS since it's to much code to write and you have to test to it too often in you browser while writing. And you don't want to write it in LibreOffice or some other kind of office suite for some reason. For example because you're working in a team and you're using some kind of version control system like git. Therefore a plain text format is much more beneficial then a binary blob.

1.3. What is Riven not?

Now you may have a clue what Riven is. But it's maybe even more important to understand what riven actually is **not**. Well Riven is neither a editor like atom nor a typesetting system like TeX nor a markup language like markdown or reStructuredText nor a office suite nor a web based service like Lit Lift or Fast Pencil. It's just a command line tool which transforms markdown files into PDF documents (and HTML, mobi, ePub etc in the future). So it's more comparable with the sphinx documentation generator.

Also Riven doesn't run on windows machines. Sorry for that.

Additionally Riven lacks some features you may know from sphinx. For example HTML site generation is not there (yet!) and Riven is (yet!) not able to generate man pages, ePub and mobi formats. But those features are planned and will be added in the future. Also Riven intentionally doesn't generate chapter numeration like you may know from sphinx. You have to do it on your own (and you can just omit it if you don't like it or even mix it up). But maybe there will be an option in the future which makes Riven to auto generate chapter numbers for you.

1.4 Riven compared to ...

1.4.1 ... Sphinx Documentation Suite

Sphinx is an awesome tool if it comes down to documentation of software. Unfortunately it requieres you to maintain a big python file to configure the build. Rivens goal is to provide a very simple but yet powerful way to write your book without any programming knowledge. Additionally it's hard to modify the layout of Sphinx' PDFs. You need to know LaTeX and write a lot of it in order to change the not so fancy default layout of the PDFs. Riven tries to give you the best result with as few configuration as possible to provide a beautiful PDF out of the box. And if the layout doesn't fit your requirements you can change it with simple CSS rules. Additionally Sphinx lacks of the feature to define a cover page. On the other hand Sphinx has a nice referencing feature for chapters

1.4.2 ... GitBook Toolchain

The toolchain of GitBook is a well crafted build tool to generate PDFs, ePub, mobi and HTML books out of markdown. The downside of GitBook is the fact it requires you to call many different commands in order to build your book. Riven tries to keep it simple: Just call the `riven` command and you'll get your PDF (and HTML, mobi, ePub etc. in the future). Additionally it's easy to change the layout of the PDF with the GitBook Toolchain. With Riven

you may change the layout of the PDF via simple CSS rules. Further GitBook has a very strong opinion how a book has to be structured. So there has to be a README.md file, which doesn't have a chapter number in the HTML representation, but in the table of contents of the PDF and so on. Riven let's you setup and structure your book just as you like without doing to much magic.

1.4. Why was Riven created?

I've two hobbies. Ok I got more hobbies, but only two of them are relevant for Riven.

One is programming. I code a lot. Additionally it's my job so I code even more. And a good programmer documents his software. In my company we're working on a high innovative CMS with a very versatile framework which allows us to implement nearly every kind of project. We had to document that framework in some way for the other developers who would use it and implement the customer projects based on the CMS. One requirement to the documentation was portability. It should be readable on the PC, on Tablets and it should be printable. As we started to write the documentation, we used OpenOffice and saved it as a ODT file which is basically a ZIP-File with some XML within. In order words: We had a big binary file in our git repository. It doesn't took a long time until we had merge conflicts due the fact that up to three people have been working on the document at the same time. And merging a big binary blob doesn't make fun. So I wanted to switch to another tool chain to write the documentation. The obvious suggest was HTML of course. But to write HTML is much work, we didn't want to do that. We know HTML, since we're web developers, but we wanted a solution which would be more lean and allows a faster workflow without much testing. Of course we could have used our CMS, but we wanted to have the documentation in our git repository. However we wanted to use plain text to write your documentation and we wanted a PDF as a result. Unfortunately there wasn't any good comparable tool that time.

My second hobby is Pen and Paper. Dungeons and Dragons, you know (Cthulu, Aborea and Pathfinder to be exact). And I was writing a campaign. I was writing that campaign in markdown, because I love my editor and I wanted to use git to manage everything since GitLab renders markdown nicely. Now I needed a build chain to produce a PDF in order to have one document which can be printed or sent via mail.

For these both use cases I hacked the initial proof of concept version of a md2pdf ruby script, which worked pretty well. But I wanted some additional features like styling of the PDF via CSS, a cover page, syntax highlighting and some other stuff. So the script started to grow to a bigger project. After some time I've renamed it to riven, since, well, it sounds cool. And yes it's reference to the Myst series.

In order to keep even big documents structured and clear, riven 1.1.0 introduced the *include* syntax, which allows a better split up of the content over many files, subdirectories and allows an easier refactoring of the chapters.

Riven solved both use cases very well. I'm using riven to write both large Pen and Paper campaings and some small documents. Either quick or within weeks of hard work. Both cases produce a nice PDF. Today the CMS framework documentation of my company includes over 200 PDF pages generated out of 9500 lines of markdown, which are changed on a weekly basis and Riven does a nice job.

1.5. How can I support Riven?

Riven is a MIT licensed Open Source project hosted on GitHub. Just vist https://github.com/phortx/riven. You'll find the issues section there containing all bugs and feature requests. Also you can fork the project to send pull requests. Feel free to contribute code, documentation or bugfixes!

If you find any bugs, please open a new issue in the issue tracker and I'll try to fix it.

2. Setup Riven

In this chapter we'll tackle the setup of Riven. You'll learn everything you need to know regarding the installation and especially the prerequisites and pitfalls.

2.1. Prerequisites

Riven comes with two dependencies, which you have to statisfy before you can start to use it. But no panic it's not that hard.

First of all you'll need wkhtmltopdf in order to generate PDFs, since that's the PDF generator backend of Riven. And you should use the QT patched version of `wkhtmltopdf` to get all the features of riven. You may also use the version without patched qt, but that will disable the following features of riven: Page numbers, table of contents and covers.

All right, Let's go ...

- If you got Arch Linux, you can just install the packages `wkhtmltopdf-static` and `icu48` from the AUR and you're done. Pretty easy, isn't it?
- If you got another Linux Distribution (like Ubuntu) or OSX you can download wkhtmltopdf from the official website. Not that hard.
- You can also compile `wkhtmltopdf` from the sources. See the official GitHub repository for details. This may take some time: On my i7 it takes about 20 minutes.

After wkhtmltopdf is compiled, make sure you can execute the `wkhtmltopdf` command in your shell:

```
$ wkhtmltopdf -V
```

If it works, everything is nice and you may proceed with the next step. If not, please make sure, `wkhtmltopdf` is correctly installed and the executable is within your `PATH`.

In the second step, we need ruby. I recommend you to use RVM to install Ruby. It's pretty simple:

```
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3
$ \curl -sSL https://get.rvm.io | bash -s stable
$ rvm install 2.2.3
$ rvm --default use 2.2.3
```

After that you should be able to use ruby:

```
$ ruby -v
```

If so, we're ready to install Riven in the next chapter!

2.2. Install Riven

The installation of Riven is pretty simple. Just run:

```
$ gem install riven
```

And you're done!

After that you should have a `riven` command:

```
$ riven -V
```

That's it!

3. Learn the basics

Now as Riven is up and running, let's have some fun and start writing! In this chapter you'll learn the basics and how to use Riven to generate PDFs.

However if you're looking for an good example: You're actually reading one! This documentation is generated with Riven and you can read the markdown source code at the doc directory in the GitHub repository of riven. The generated PDF can be found in that directory too.

If you want to learn markdown, you should take a look at the Learn Markdown book from GitBook which can be read online.

3.1. A single file

Let's start small and transform a single file into your first beautiful PDF!

Create a new markdown file where ever you want and name it `awesome.md`. Put some markdown into the file just like this:

```
# Hello World!

**This** is my first Riven generated PDF!
```

Now we compile the PDF with a simple Riven call:

```
$ riven awesome.pdf
```

You should get a PDF file called `awesome.pdf`. Open it and admire your first PDF.

3.2. Many at once

Now we know how to convert a single file into a PDF document. But usually we have more then one file. For example one file for each chapter. Riven may take a bunch of markdown files and compile them into a single PDF file.

In order to use that feature, we need some more markdown files:

`chapter-1.md`:

```
# Chapter 1

Hello World!
```

`chapter-2.md`:

```
# Chapter 2

Another example
```

`chapter-3.md`:

```
# Chapter 3

Last but not least
```

Now you can merge all these files into a single PDF with the following command:

```
$ riven -o awesome.pdf chapter-1.md chapter-2.md chapter-3.md
```

You'll get a single PDF file, just as in chapter 3.1., which contains all three chapters and the regarding content. The `-o` param determines the name of the generated PDF file.

Riven is also able to guess the PDF file name if all markdown files are located in one directory. So let's create a directory called `awesome` and put our three markdown files into it. Now you should have a file structure like this:

```
$ ls -1
awesome/

$ ls -1 awesome/
chapter-1.md
chapter-2.md
chapter-3.md
```

The following command will generate exactly the same PDF file as the command we used before, but you don't have to provide a PDF file name. Riven will take the directory name as the PDF file name.

```
$ riven awesome/
```

You'll receive a `awesome.pdf` file like before.

Consider that Riven will merge the markdown files of the directory in alphabetical order.

3.3. Structure your files

In the two previous chapters you learned how to convert a single file or a bunch of files into a PDF. In this chapter you'll learn a nice mixed way how to structure bigger documents with many markdown files without the need to pass all those files to the riven command or to ensure the alphabetical order is the order you want.

The magical feature of Riven which allows us to bring a clean tidiness in your files, is called *include directive*. It's a markdown syntax which is not part of the original markdown language,

but a feature of the Riven Extended Markdown - or REM. REM is a superset of the GitHub Flavored Markdown introduced by Riven in order to provide some additional features. The include directive allows you to define another markdown file. Riven will merge the content of that markdown file into the outer markdown file by replacing the include directive.

The include syntax looks like that:

`<<[ file ]`

where `file` is the path to another markdown file.

Here's an example:

`a.md`:

```
Hello World!

<<[ b.md ]

Bye World!
```

`b.md`:

```
This content is **included**!
```

If you call

```
$ riven -o a.pdf a.md
```

Riven will merge the two files like that:

```
Hello World!

This content is **included**!

Bye World!
```

This feature allows you to organize your document over many markdown files and subdirectories. Take a look at https://github.com/phortx/riven/tree/master/doc to see a good example for the usage of includes and how to structure a document.

1.4. The config file

If you take a look at the Riven help (`riven -h`), you'll see that riven supports a bunch of command line options. It can be a bit painful to remember all those parameters in order to regenerate your PDF from time to time. So it's recommended to setup a config file for riven where you can set all your settings instead of passing them to the `riven` command. This can you save you a lot of time.

A Riven config file is a simple YAML file, like you may know from Rails or Symfony. But they're really easy to understand, so no panic if you never heard of it before.

An example riven.yml looks like that:

```
pdf_output_file: example.pdf
cover_file: cover.md
css_file: style.css
generate_toc: true
toc_headline: 'Contents'
dump_html: false
dump_cover_html: false
verbose: false
files:
  - index.md
```

You may provide as many files as you want under the `files` section. Each file gets one line. For example:

```
files:
  - index.md
  - chapter-1.md
  - chapter-2.md
  - chapter-3.md
```

Riven searches for a `riven.yml` in the current directory. But if you config file is located in another directory, you should pass it via the `-C` param. However if you structure your document like this:

```
$ ls -1

chapters/
cover.md
index.md
riven.yml
style.css
```

it will be pretty easy to generate your PDF, since you just have to call `riven` without any params:

```
$ riven
```

It will use the `riven.yml` from the current directory and load all settings from it.

You're also allowed to mix up command line parameters and the config file. For example you can override the stylesheet:

```
$ riven -s another_style.css
```

The command line params will override the settings from the `riven.yml`.

4. Advanced topics

In this last chapter we'll tackle all advanced features of Riven like the styling of the PDF with CSS, the table of contents, cover pages and syntax highlighting of code snippets.

4.1. Style your PDF

Riven allows you to style your PDF via CSS rules. This feature makes it possible to change colors, fonts, margins and many more.

However please consider that this is a PDF, so the capabilities of the layouting via CSS is limited. Not every CSS rule you know will work.

In order to use a CSS file, you have to create a `style.css` for example and pass it to the `-s` param of riven:

```
$ riven -s style.css documentation.md
```

Some hints:

- Every text paragraph is wrapped into a `<p>` tag
- The headlines will be transformed to `<h1>`, `<h2>`, `<h3>` ... tags, where `#` is `<h1>`, `##` is `<h2>` and so on
- Quotes (`>`) are transformed to a `<blockquote>` tag
- Bold text is transformed to a `<strong>` tag
- `<hr>` tags are used for horizontal lines (`-----`)
- Unordered lists are transformed to `<ul>` and `<li>` tags and ordered lists are transformed to `<ol>` and `<li>` tags. Nested lists are `<ul>` tags within a `<li>` tag
- Inline fixed width text (backticks) is transformed to `<code>` tags
- Multiline code blocks are transformed into `<pre>` tags

To style the cover page, you can use the `.cover-page` CSS class. The following example will change the h1 color on the cover page to red:

```
.cover-page h1 {
  color: red;
}
```

4.2. Automatic Table of Contents

For an automatic generated table of contents after the cover page, you can add the `-t` param and pass a headline for the table of contents:

```
$ riven -t "Contents" -c documentation/cover.md documentation/
```

Riven will generate a table of contents for all headlines. The user can click on the entries to jump directly to the regarding headline.

4.3. Cover pages

To start your PDF with a nice cover page, define a cover markdown file and pass it to the `-c` param. Riven will prepend that file before all other files and doesn't attach a page number.

```
$ riven -c documentation/cover.md documentation/
```

It can be helpful to use HTML within the cover page to have more control over it's layout.

4.4. Syntax highlighting

Riven comes with syntax highlighting for code blocks:

```
` ``ruby
  def foo
    puts 'bar'
  end
` ``
```

(Ignore the whitespace between the backticks)

This code sample will produce the following field in the PDF:

```ruby
  def foo
    puts 'bar'
  end
```

The syntax highlighting is powered by coderay and Riven is using a github theme.