the cover page

Table of Contents

# Chapter 1

## About this documentation

### The dilemma

At the time of this writing, I faced a dilemma, which word processor is the best processor for writing this documentation.

The most obvious choices were:

- Microsoft Word
- Apple Pages
- Google Docs

Surely any of the choices above would be suitable, however, in my humble opinion, such options can be somewhat 'evil' when it comes to document formatting, turning most documentation written in one platform unreadable in the other or to the very least almost certainly not looking as intended.

In general, converting the document to a pdf format remediates this problem, however, I am now left with the fact that my content is tightly coupled to the editor that created the documentation. Modification is difficult, for example, someone else may wish to edit the document, or perhaps if at some point in time in the near future I decide to present my documentation as an html document on the web, or perhaps as a deck of slides or even publish it in a book format, I may have to revise the entire text looking for any 'evil' formatting issues that was not visible in the former format.

Finally, as a writer, when writing documentation, I should concentrate on writing the documentation, and not about auto generated formatting issues that may arise and drag productivity in a typical writing session. The writer should only worry about the semantics and the content of his/her writing, formatting should be done separately, perhaps not even by the writer himself, or better, simply choose a new format from predefined options written by a talented designer.

LaTeX is a really good option to move away from the formatting problems mentioned above, and beyond, however, it does trap the writer with a little clutter to tinker with in terms of settings and so on. What I am trying to say is, once the document is finalised, it no longer consists of the content and the content only, but also carries several formatting tags. This is one hand demonstrates how powerful LaTeX can be, but in the other, may confuse and distract the writer.

### The solution

Since this course is about computer science, I set off to find a solution that would allow me and any other developer to run away from the masses and write simple interchangeable documentation with ease. Writing a solution that could potentially be further enhanced to the point a non programmer could also benefit from.

The requirements were:

- Write content and content only, without distractions
- Formatting should be written separately and be interchangeable/themeable
    - This also means that if some time in future I want to publish my content to a different media, I should be able to do so without too much effort
- It should not be coupled to any specific text editor
- It should have an automated building and deployment solution

The answer was always there, markdown, markdown is a really simple and easy to use markup language, it

is the de facto standard for readme files in software development and widely used in blogs throughout the web.

A short introduction to markdown syntax can be found in the link below:

http://daringfireball.net/projects/markdown/syntax

Although markdown may not be as powerful as LaTeX, the community around it is immense and really keen in providing further enhancements to it, the following link lists many projects and plugins that address some of those short-comes:

https://github.com/cben/mathdown/wiki/math-in-markdown

The community also provides an excellent open source markdown editor called Mou, although Mac specific, all other major operating systems have free alternatives to Mou:

http://25.io/mou/

Finally, there is an open source project which is now maintained by Jakob Truelsen and Ashish Kulkarni, called wkhtmltopdf, this headless command line tool, allows any html document to be converted to pdf, and since markdown is easily convertible to html, we now have all the tools we need in order to create our documentation.

http://wkhtmltopdf.org/

## The workflow

To make this work, we will need to automate every step of the process, so that we can only write markdown, then compile/deploy our work with only one command and in the process, if we wish to do so, add some personalised styles to our document.

The application that holds the documentation has been scaffolded using Yeoman, more specifically, a generator called generator-jekyllrb

...

This generator includes the following:

-

- 

The source code for this documentation can be found at:

https://github.com/jbonigomesbbk/jbonigomesbbk.github.io

The latest build of this documentation can be found here:

http://jbonigomesbbk.github.io/docs

You will also find a link to download the pdf right below all chapters the chapter links in the left menu.

# Chapter 2

Content for chapter 2