LINUX VOICE
TUTORIAL

MARKDOWN: WRITE ONCE, PUBLISH ANYWHERE

MARCO FIORETTI

Stop wasting time with word processors and follow the Markdown way to future-proof your words.

WHY DO THIS?

- We produce more and more text every year, and never know how it will be reused.
- Even Free Software can become obsolete, but a source format as simple as Markdown will always remain usable.
- If you produce lots of text, your productivity will increase. Trust us.

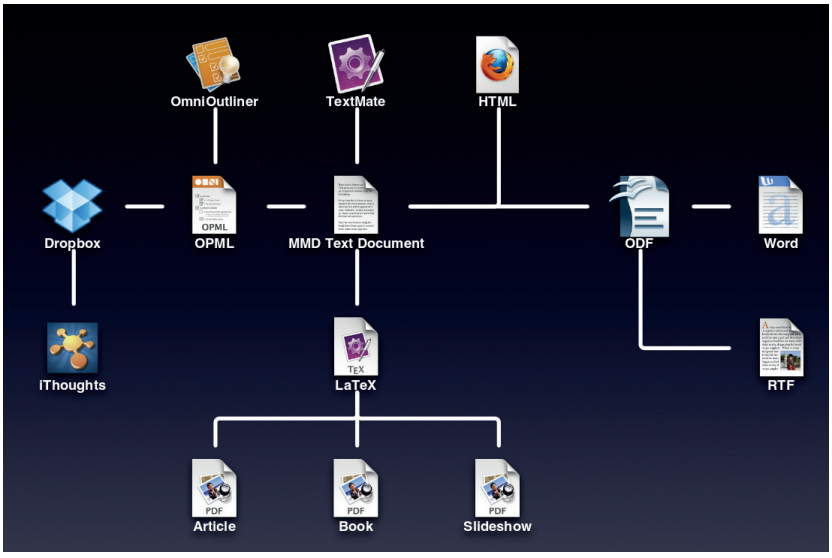
Markdown is just one of many plain text markup systems available as Free Software (we'll return to that definition in a moment). Writing in Markdown is simple enough to understand, but the real challenge behind Markdown is in understanding why it was created, why it can be good for you, and above all, how to change your writing and publishing habits in order to get the greatest advantage from it.

If you don't regularly create significant amounts of text of whatever nature, from poetry to company or parish newsletters, Markdown will be of little or no relevance for you. But if you do, or even if you just edit, manage and publish texts written by others, Markdown can be a real blessing, for two big reasons.

The first Markdown goal is to separate content from formatting as much as possible, to save time and concentration. After decades of word processing, and seeing too many "professionally formatted" texts looking horrible on the Web or in our smartphones, we have all learned that all too often, many of the functions in traditional office suites produce just a big waste of time and energy. We obsess ourselves with fonts, margins and similar formatting details instead of just writing clearly. Then, much of that effort goes down the drain every time somebody loads what we wrote into another program, or on a small screen.

The second, even bigger rationale for Markdown may be summarised with the slogan "Write once, publish anywhere". In other words, the simpler your initial format is, the easier it will be to reuse your content, automating as much as possible of the work. To really understand how and why Markdown is great, we have to remember the implications of the current "What You See is What You Get" (WYSIWYG) paradigm. In order to give you WYSIWYG, modern word processors automatically add tons of data and instructions to all the files they save in their native formats, and these instructions often aren't all that portable between formats. The result is more things that could go wrong whenever something changes, more temptation to make things look "just right" manually, and more work to move from one format to the other, eg from OpenDocument to HTML or PDF.

You never know when you will need to republish something you wrote – so why not use a multi-output source format right from the beginning?



Separate style and substance

A plain text file, instead, is a file that contains only the bytes corresponding to the actual letters, punctuation and typographic "operators" – that is spaces, tabs and newlines – that we type into it using a text editor. Such files sure don't look as pretty on screen as the pages of commercial magazines, but we should learn not to care. They are extremely portable and, consequently, much more future-proof than any other alternative. They also avoid distractions: the less eye candy you can add or see, the more you are forced to ask yourself if what you wrote IS worth reading.

Very often, however, the deliberate limitations of plain text files may hurt the clarity and readability of your writing. Structures and properties like indentation, lists, typefaces or embedded images do make text easier to understand, don't they? The solution is to express them by marking the text up by adding normal characters with a special meaning, called markers or, more frequently, tags. Let's take the italic typeface as an example – a markup user would never apply it by selecting text and then clicking on some "italic" button, or menu entry. They would, instead, adopt a convention like "//all text between a couple of slashes is meant to be italic//" and then just type (and see on screen or paper!) those extra characters, every time italic is needed. Primitive and ugly? Maybe, but also extremely efficient.

Markdown, finally

After this long, but absolutely necessary introduction, we can finally define what Markdown is, and how to use it. First, it's a set of rules to mark up plain text, defined by John Gruber in 2004. See the "Flash introduction to Markdown syntax" box for some

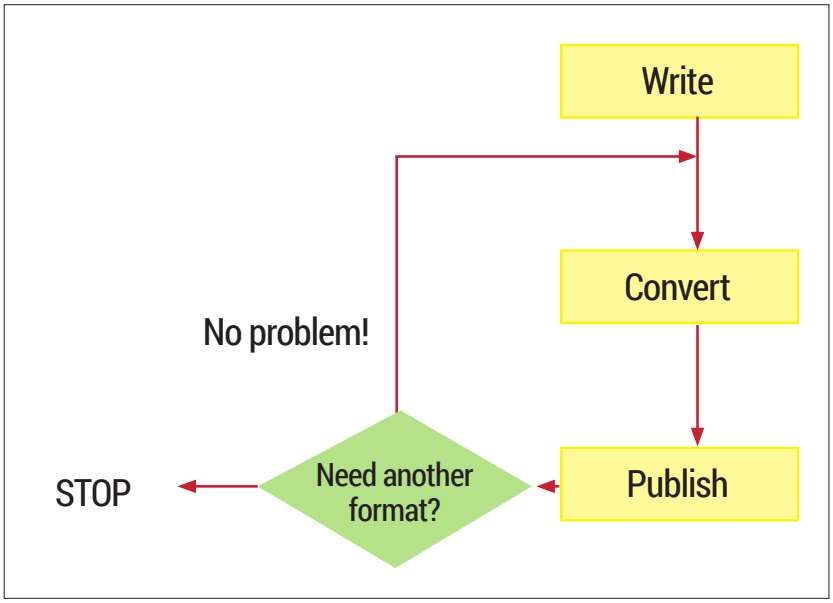
examples. Second, it's the software that converts that text to whatever target format its users need in any given moment.

The basic Markdown syntax, and all the programs written for it, were conceived for the web, to define and generate basic HTML. Very soon after the appearance of Markdown, however, other developers created syntax extensions and corresponding applications to make it possible to convert Markdown into ODF, Latex and more. Let's look at the basics first, however.

Markdown was designed to satisfy two non-negotiable criteria; efficiency and, above all, readability. If you ask your browser to show you the source of any web page, you will immediately remember that the "M" in HTML means just that: "Markup". The problem is that writing and reading raw HTML markup is tiring, error-prone and time consuming. By contrast, the main rule in Markdown is that any text formatted with it should be easily readable as-is, even by people who have never even heard of Markdown. As many advocates of this method say "the single biggest source of inspiration for Markdown's syntax is the format of plain text email".

To favour readability, all Markdown tags consist of punctuation characters, chosen to look as unintrusive as possible. The idea is to write Web-ready content faster, not to insert HTML tags faster. By deliberate choice, tags are defined only for that very small subset of HTML that corresponds to properties or operations applicable to raw text. For any other markup, from page backgrounds to navigation menus, you are supposed to use HTML (In practice, as we will see shortly, there are ways to not perform such operations manually in many cases).

In a Markdown file (the standard extension is **.md**) you can switch from Markdown to full HTML and back at any moment, without problems. The only restriction is for block-level elements – that is, practically every long block of HTML that corresponds to one object (eg a table, or one preformatted chunk of source code), and as such is enclosed by a matching pair of tags, like **<table>** and **</table>**. Inside **.md** files, such elements must be preceded and followed by blank lines, and their enclosing tags cannot be indented with tabs or spaces.



This is the power of the Markdown flow: once you have written the source, you can repeat the conversion and publishing steps as many times you want.

The Markdown flow

In the real world, a complete Markdown-based publishing flow will have at least these three phases:

1. Generate your text with all the required Markdown tags.
2. Run the converter that generates the HTML version (or other formats, if needed) of your work.
3. "Publish" that version, that is send it by email to whoever may need it, put it online and so on.

Can you see the efficiency, power and flexibility, in one word: the freedom embedded in a flow like this? No? Don't worry, here it is: there is no need to perform those operations all on the same computer, all by the same person, or manually, or all at once. As far as phase 1 is concerned, any text editor, on any operating system, can be used to write Markdown (or to update Markdown files that somebody else wrote 10 years ago). And if you're using a halfway decent editor, it will surely have a Markdown syntax highlighting mode. If you wanted to publish the log files from your server, your email archive or any other body of plain text, you could write a script that converts everything to Markdown, and make the whole flow automatic.

Phases 2 and 3 can be very easily delegated to a **cron** job, and often should be. There is also no problem if years pass between one phase and the next. Everything you produce with a system like Markdown is, by definition, reusable in ways that you may not even know that you'll need one day. For example, suppose next year some blog invites you to republish stuff you wrote in 2008. If that stuff is in Markdown format, and the webmaster is willing to

LV PRO TIP

Even if you decide to not use Pandoc to generate the output formats of your Markdown documents, study and try it a few times. If nothing else, it may help you to convert all your text files to Markdown, even if they are in Microsoft or OpenDocument formats.

Documentation

Cheatsheets? Yes, a good, detailed cheatsheet is all you will need to get started with Markdown after reading this tutorial. Don't ask us for one, however. Search for them online and you'll find plenty, in all possible formats from Markdown (of course!) to desktop wallpapers. Once you have learned Markdown, study the practical YAML tutorial at **http://rhnh.net/2011/01/31/yaml-tutorial**.

Before that, however, we suggest you read the post **www.terminally-incoherent.com/blog/2012/05/25/Markdown-for-muggles**, which is a funny encouragement to use Markdown. Another article to read before starting is 'Thoughts on Markdown' (**www.leancrew.com/all-this/2010/10/thoughts-on-markdown**). On a more technical level, other useful resources are the man page 'pandoc_markdown', which explains the differences between basic Markdown and its Pandoc superset, and the lists at **https://github.com/jgm/pandoc/wiki/Pandoc-vs-Multimarkdown**. That wiki page thoroughly compares the two extended converters feature by feature, starting with the input and output formats they support.

install the *Markdown QuickTags* plugin for *WordPress*, all you'll need to do is copy and paste your Markdown sources in the *WordPress* form. Please stand back one moment in silence with us, to appreciate the awesomeness of it all!

A practical example

Here's a simple Markdown source file snippet that mixes both HTML code (in this case for a navigation menu, not the actual content!) and Markdown:

```
<!-- Navigational markup -->
<ul class="nav">
<li><a href="./">Home</a></li>
<li><a href="contact/">About</a></li>
<li><a href="contact/">Contact</a></li>
</ul>

"...MultiMarkdown provides an easy way to share formatting between all of my devices. It's easy to learn (even for us mortals) and immediately useful."
> --- David Sparks, [MacSparky.com](http://MacSparky.com/)

## Why Markdown and MultiMarkdown?
Because life is too short to waste it *formatting* text, instead of just **writing it**.
```

The image below shows the full HTML version generated by any Markdown converter, and the image on the facing page shows the way it looks inside a web browser. See what we meant? Even without detailed explanations, or having a cheatsheet handy, both the original plain text and what is eventually shown by a browser are much more readable than the HTML.

Images and hyperlinks

One image is worth a thousand words, or so they say, but how do we handle them in plain text files? The answer is "With a little bit of care". You can tell your your Markdown converter to insert in the target file the HTML code that displays an image using this syntax:

```
![Here you should see an image](/path/to/img.jpg "This is the image title")
```

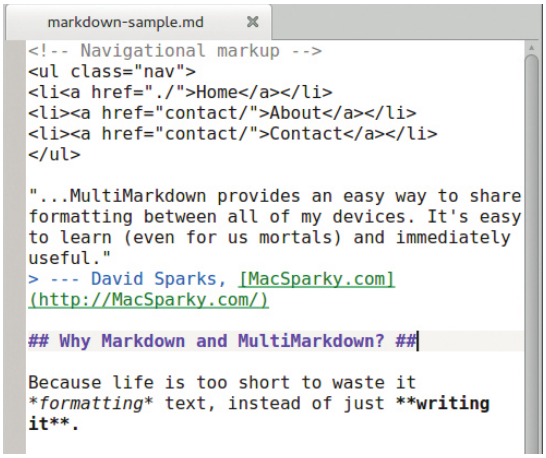
This combines three strings: alternative text for textual browsers, a path to the image, and the image

The HTML code generated converting the Markdown source of our snippet.

```
<!-- Navigational markup -->

<ul class="nav">
<li><a href="./">Home</a></li>
<li><a href="contact/">About</a></li>
<li><a href="contact/">Contact</a></li>
</ul>

<p>"...MultiMarkdown provides an easy way to share formatting between all
of my devices. It's easy to learn (even for us mortals) and immediately
useful."</p>

<blockquote>
  <p>--- David Sparks, <a href="http://MacSparky.com/">MacSparky.com</a>
</p>
</blockquote>

<h2>Why Markdown and MultiMarkdown?</h2>

<p>Because life is too short to waste it <em>formatting</em> text, instead
of just <strong>writing it</strong>.</p>
```

LV PRO TIP

Whatever Markdown converter you choose, you shouldn't use it directly at the prompt. Instead, write a shell script that will call it automatically and save a log file somewhere. This will greatly reduce the possibility of mistakes, and make you work even faster.



This is what the Markdown syntax looks like in a text editor. Marked text is coloured to make it even more readable, while embedded HTML code is left as is.

title. The problem, of course, is that this will generate enough HTML to display your image, but not enough to align, frame and size it just as you wish. There are two solutions here. One is to not use Markdown for images, but actual HTML, with all the options you may need:

```
<img src="/path/to/img.jpg" width="95%" alt="Here you should see an image"
```

A much smarter way to go is to write plain Markdown, and then pre-process your sources with an extra script that replaces all the Markdown image tags with equivalent ones in HTML, all with the same settings, CSS classes and so on.

As far as hyperlinks are concerned, the syntax itself is very simple:

```
This is [an example](http://example.com/ "Title") inline link.
This is [an example][id] reference-style link.
```

The first line shows how to define links to a generic web page: you put the URL inline between the string that will become clickable text (the part in square brackets) and the title that will appear when a user puts the mouse pointer over the link. Reference-style links are much more interesting, at least for frequently used URLs. The second square bracket is a pointer to the complete definition of the URL that goes there, which will have this format:

```
[id]: http://example.com/ "Optional Title Here"
```

and can be anywhere in the Markdown file! This means that you can write your own list of frequently used URL aliases, all in one file, and then append it with a script to all the Markdown sources that need any of those URLs!

Templates and metadata with YAML

Markdown is good, and powerful. Adding metadata to it makes it much more powerful. How? A very common way to do it is by using YAML (which is another of those annoying recursive acronyms – this one stands for "YAML Ain't Markup Language"). This is an open standard, designed to store generic, structured textual data just like XML, but in a way

that's much more easily readable and editable by humans. YAML metadata is usually placed in a separate frontmatter, at the very beginning of a Markdown file, between two lines containing three dashes each:

```
Title: My first Markdown/YAML post
date : 2014-08-01
categories: Free Software, Blogging, Open Standards
```

Please note that this is a very trivial example of YAML. As simple as it looks, YAML can store many types of data, from lists and abstracts to associative arrays. There are plenty of free Software parsers that can process it, or generate YAML frontmatter from many sources. Used together, YAML and Markdown can handle large amounts of text in a way that is very easy to use, but also very powerful.

Markdown editors and converters

As we already said, any halfway decent editor will make your Markdown source files even more readable, thanks to syntax highlighting. When it comes to conversion, there are many choices. You can even generate HTML from Markdown sources when working on somebody else's computer, by using the official online tool, called *Dingus* (**https://daringfireball.net/projects/markdown/dingus**) or one called *Dillinger* (**http://dillinger.io**), which you can even install on your own server.

On a Linux desktop, you can use the original converter, a Perl script called **markdown.pl**, or more advanced tools like Pandoc (**http://johnmacfarlane.net/pandoc**) or Multiple Markdown (MMD, **http://fletcherpenney.net/multimarkdown**). They are all command line tools, well suited for automation, and relatively simple to use. Basically, you pass them the input file (or the Standard Input) and the name of the output file in which they should save the result. The differences are in the number of input and output formats supported, and in the set of Markdown tags they recognise and can process. Therefore, there is no "best converter". You must figure out by yourself which one best matches your taste or, more importantly, the type of documents you must write. The original converter, for example, only accepts basic Markdown and outputs HTML. Pandoc, instead, is a generic tool that can also be used to convert to the Markdown format Web pages or many other documents.

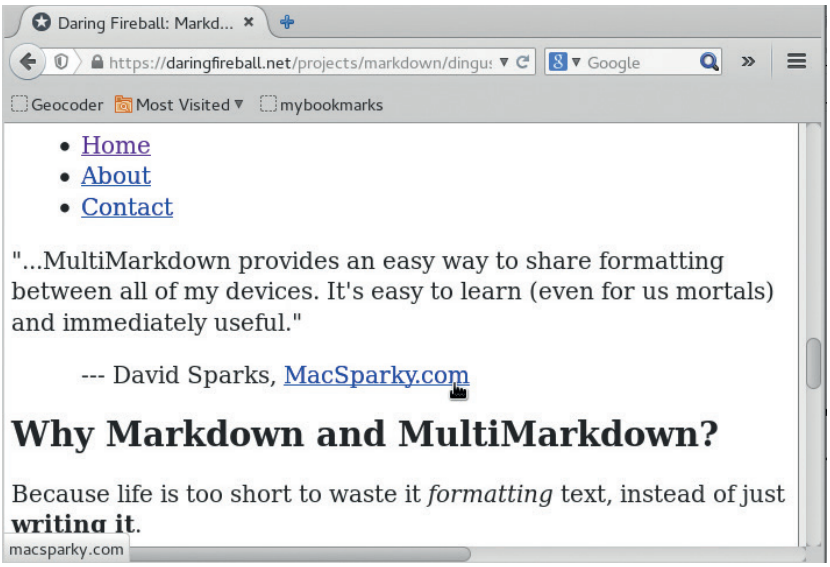
Pandoc defines extra Markdown tags to handle, among other things footnotes, tables, flexible ordered lists, automatic tables of contents, embedded Latex formulas, citations, and markdown inside HTML block elements. When you run the converter, multiple input files are concatenated automatically. Pandoc can even accept URLs as input files! Output goes to **stdout** by default, except for complex formats like OpenDocument or ePUB. In this way, it's also possible to generate PDF files directly from Markdown.

The other most useful features of Pandoc are the command line options that tell it to place the content of external files, for example SEO keywords, in the header of the HTML output, or at the end of a page.

MMD is another pair of Markdown syntax superset and associated converter. It is optimised for handling, among other things, tables, footnotes, citations, internal cross-references and equations. Cross-references, for example, work in this way:

```
[This string will point to an internal link][mylink]
## This is where I will end up when clicking on the string above
[mylink]
```

MMD can also convert Markdown sources to Latex, which is the basis for professional-quality PDFs, with its auxiliary tool **mmd2tex**. Other, extremely important output formats supported by MMD are OpenDocument and OPML, the standard used in the *Fargo 2* blogging platform.



The final result: fully standard code that any browser will render without problems, with a structure perfectly matching the original Markdown file.

Flash introduction to Markdown syntax

Warning! This is very far from being a complete description of the Markdown syntax. We only want to whet your appetite by showing how easy it is to create structured, yet highly readable plain text using Markdown. Besides, even if we had enough space, it would make no sense to give you a complete syntax primer here. The whole format is so simple, and already completely documented in countless cheatsheets that it would make no sense to copy it here:

```
*Single asterisks (or underscores) enclose italic text*
**Couple of asterisks, instead, mean "bold!"**
```

Headers can be marked in two ways. The simplest is this:

```
# Level 1 header #
## Level 2 header ##
```

Numbered lists:

```
1. Foo
2. Bar
```

Unordered lists:

```
* first list item
* another list item
```

```
> Block quotes work just like in email.
> Put a ">" sign at the beginning of
> each of their lines.
```

LV PRO TIP

Markdown is great and may be a good reason to change text editor, if your favourite one doesn't highlight its tags properly. We suggest you try multiplatform editors if you haven't already, as they allow you always to work in the same way!

Marco Fioretti is a Free Software and open data campaigner who has advocated FOSS all over the world.