Introduction to Markdown

Let's take an overview of **Markdown**, how it works, and what you can do with it.

What is Markdown?

Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents. Created by John Gruber in 2004, Markdown is now one of the world's most popular markup languages.

Markdown is different to a WYSIWYG editor. In those applications (Microsoft Word or Apple Pages) you click buttons to format words and phrases, and the changes are visible immediately. Markdown isn't like that. When you create a Markdown-formatted file, you add Markdown syntax to the text to indicate which words and phrases should look different.

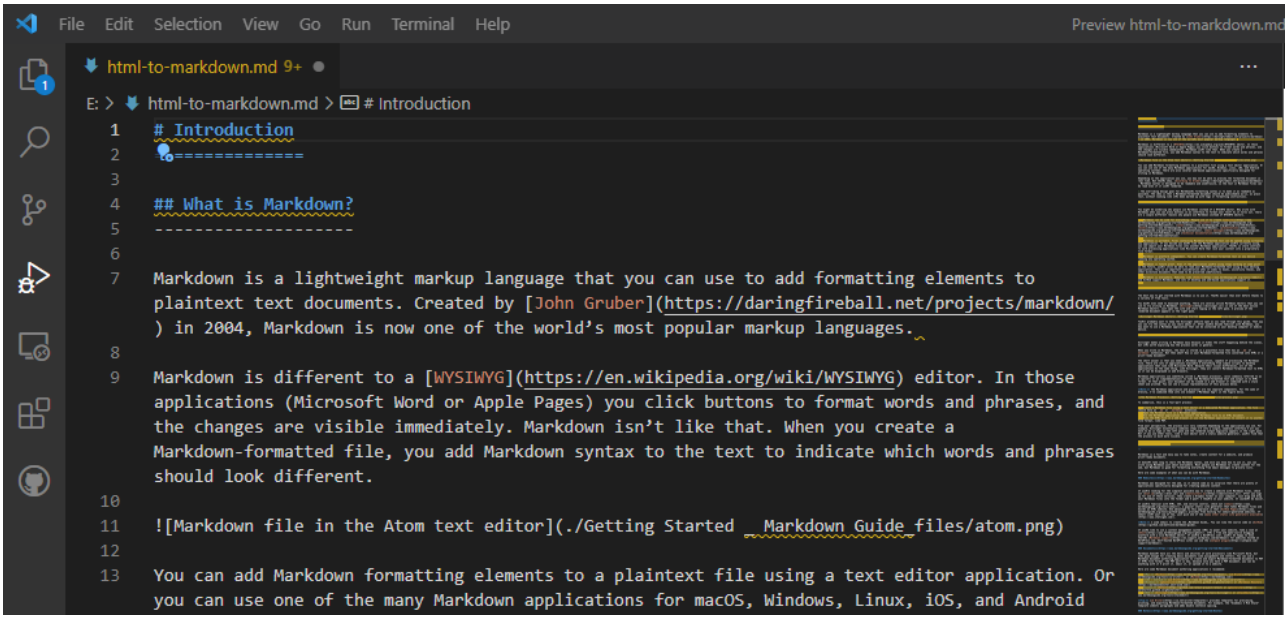


Figure 1. Markdown editingo in a plain tex processor (Visual Studio Code).

You can add Markdown formatting elements to a plaintext file using a text editor application. Or you can use one of the many Markdown applications for macOS, Windows, Linux, iOS, and Android operating systems. There are also several web-based applications specifically designed for writing in Markdown. According to Gruber, Markdown syntax is designed to be readable and unobtrusive, so the text in Markdown files can be read even if it isn't rendered.

> *The overriding design goal for Markdown's formatting syntax is to make it as readable as possible. The idea is that a Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions.*

Why Use Markdown?

You might be wondering why people use Markdown instead of a WYSIWYG editor. Why write with Markdown when you can press buttons in an interface to format your text? As it turns out, there are a couple different reasons why people use Markdown instead of WYSIWYG editors.

- Markdown can be used for everything.
    - To create websites
    - documents
    - notes
    - books
    - presentations
    - email messages,
    - and technical documentation.
- Markdown is portable. Files containing Markdown-formatted text can be opened using virtually any application.
- You can import your Markdown files into another Markdown application. As plain text, its not a proprietary file format.
- Markdown is platform independent. You can create Markdown-formatted text on any device running any operating system.
- Markdown is future proof. Even if the application you're using stops working at some point in the future, you'll still be able to read your Markdown-formatted text using a text editing application. This is an important consideration when it comes to books, university theses, and other milestone documents that need to be preserved indefinitely.
- Markdown is everywhere. Websites like Reddit and GitHub support Markdown, and lots of desktop and web-based applications support it.

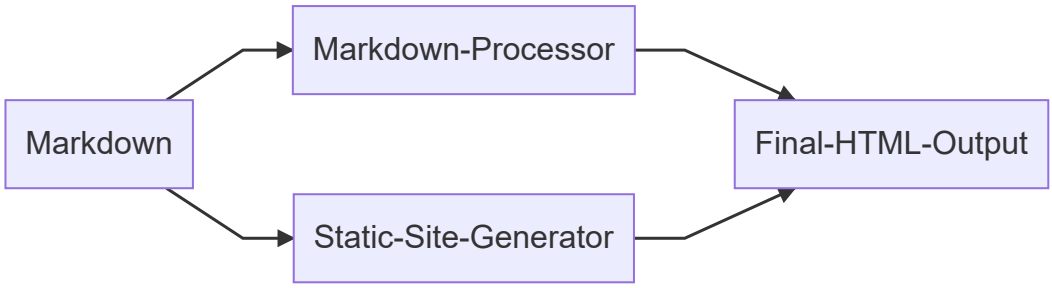
How does it work?

When you write in Markdown, the text is stored in a plaintext file that has an `.md` or `.markdown` extension. But then what? How is your Markdown-formatted file converted into HTML or a print-ready document?

The short answer is that you need a *Markdown application* capable of processing the Markdown file. There are lots of applications available (see section in this document from simple scripts to desktop applications that look like Microsoft Word. Despite their visual differences, all of the applications do the same thing. Like Stackedit, converts (and deploys) Markdown-formatted text to HTML so it can be displayed in web browsers.

Markdown applications use something called a *Markdown processor* (also commonly referred to as a "parser" or an "implementation") to take the Markdown-formatted text and output it to HTML format. At that point, your document can be viewed in a web browser or combined with a style sheet and printed. You can see a visual representation of this process below.

> ***Note:*** *The Markdown application and processor are two separate components. For the sake of brevity, I've combined them into one element ("Markdown App") in the figure below.*



To summarize, this is a four-part process:

1. Create a Markdown file using a text editor or a dedicated Markdown application. The file should have an `.md` or `.markdown` extension.
2. Open the Markdown file in a Markdown application.
3. Use the Markdown application to convert the Markdown file to an HTML document.
4. View the HTML file in a web browser or use the Markdown application to convert it to another file format, like PDF.

The process will vary somewhat depending on the application you use. For example, Dillinger essentially combines steps 1-3 into a single, seamless interface — all you have to do is type in the left pane and the rendered output magically appears in the right pane. But if you use other tools, like a text editor with a static website generator, you'll find that the process is much more visible.

What's Markdown Good For?

Markdown is a fast and easy way to take notes, create content for a website, and produce print-ready documents. It doesn't take long to learn the Markdown syntax, and once you know how to use it, you can write using Markdown just about everywhere. Most people use Markdown to create content for the web, but Markdown is good for formatting everything from email messages to grocery lists.

Here are some examples of what you can do with Markdown.

Websites

Markdown was designed for the web, so it should come as no surprise that there are plenty of applications specifically designed for creating website content.

If you're familiar with HTML, CSS, and version control, check out Hugo, a popular static site generator that takes Markdown files and builds an HTML website. One advantage to this approach is that GitHub Pages provides free hosting for hugo-generated websites. There are also many other static site generators available.

If you'd like to use a content management system (CMS) to power your website, take a look at Ghost. It's a free and open-source blogging platform with a nice Markdown editor. If you're a WordPress user, you'll be happy to know there's Markdown support for websites hosted on WordPress.com. Self-hosted WordPress sites can use the Jetpack plugin.

In the next link, a detailed list of static sites generators based on markdown are presented.

Our choice is Hugo but any of them is useful.

Documents

Markdown doesn't have all the bells and whistles of word processors like Microsoft Word, but it's good enough for creating basic documents like assignments and letters. You can use a Markdown document authoring application to create and export Markdown-formatted documents to PDF or HTML file format. The PDF part is key, because once you have a PDF document, you can do anything with it — print it, email it, or upload it to a website.

Here are some Markdown document authoring applications. Some of them are:

- **General Pourpose Editors:**

        - Atom
        - GNU Emacs
        - Remarkable
        - Haroopad
        - GitBook
        - ReText
        - UberWriter
        - Mark My Words
        - Vim-Instant-Markdown Plugin
        - Bracket-MarkdownPreview Plugin
        - SublimeText-Markdown Plugin
- **Mac:**
        - MacDown,
        - iA Writer
        - Marked
- **iOS / Android:
        - iA Writer
- **Windows:**
        - ghostwriter or
        - Markdown Monster
- **Linux:**
- ReText or ghostwriter
- **Web:**
        - Dillinger
        - StackEdit
- **Static Document Generators**
        - Docusaurus
        - Docsify
        - VuePress
        - Slate
        - Docute

Our choice is Typora not listed below because it keep for most of operating systems versions.

Notes

In nearly every way, Markdown is the ideal syntax for taking notes. Sadly, Evernote and OneNote, two of the most popular note applications, don't currently support Markdown. The good news is that several other note applications *do* support Markdown:

- Simplenote
- Notable
- Bear
- Boostnote
- MarkText
- Yosoro
- Uncolored
- Bootsnote

If you can't part with Evernote, check out Marxico, a subscription-based Markdown editor for Evernote, or use Markdown Here with the Evernote website. Otherwise Notable should be a good choice.

Books

- If you try to self-publish your book, try Leanpub, which uses markdown-formatted files as source of electronic books.
- The GitBook style was borrowed from GitBook, a project launched by Friendcode, Inc. (https://www.gitbook.com) and dedicated to helping authors write books with Markdown.
- If you have used R Markdown before, you should be familiar with the Bootstrap style (http://getbootstrap.com), which is the default style of the HTML output of R Markdown. Like the Bootstrap style, the Tufte style is provided by an output format tufte_html_book(), which is also a special case of html_chapters() using tufte::tufte_html() as the base format.
- To create an EPUB book, you can use the `epub_book()` format. It has some options in common with `rmarkdown::html_document()`.
- MOBI e-books can be read on Amazon's Kindle devices. Pandoc does not support MOBI output natively, but you may use third-party tools to convert EPUB to MOBI. One possible tool is Calibre. Calibre is open-source and free, and supports conversion among many more formats. For example, you can convert HTML to EPUB, Word documents to MOBI, and so on. The function calibre() in bookdown is a wrapper function of the command-line utility ebook-convert in Calibre.
- The **knitr** package was designed based on the idea of "Literate Programming" (Knuth 1984), which allows you to intermingle program code with text in a source document. When **knitr** compiles a document, the program code (in code chunks) will be extracted and executed, and the program output will be displayed together with the original text in the output document.

Presentations

Believe it or not, you can generate presentations from Markdown-formatted files. Creating presentations in Markdown takes a little getting used to, but once you get the hang of it, it's a lot faster and easier than using an application like PowerPoint or Keynote.

- Landslide generates a slideshow using from markdown, ReST, or textile. It builds off of Google's html5slides template.
- Remark (GitHub project) is a popular browser-based Markdown slideshow tool
- Cleaver (GitHub project).
- If you use a Mac and would prefer to use an application check out:
    - Deckset
    - Marked.
- Marp is a very good alternative due to its integration with some editors like visual studio code, it is a work in progress, but it shows promise. Short for "Markdown Presentation Writer," Marp is an Electron app in which you craft slides using a simple two-pane editor: Write in Markdown in the left pane and you get a preview in the right pane.

  Marp supports GitHub Flavored Markdown. If you need a quick tutorial on using GitHub Flavored Markdown to write slides, check out the sample presentation. It's a bit more flexible than baseline Markdown.

Email

If you send a lot of email and you're tired of the formatting controls available on most email provider websites, you'll be happy to learn there's an easy way to write email messages using Markdown.

- Markdown Here (MDH) is a simple browser extension that can be installed in browsers such as Chrome, Firefox, Safari and Opera.
- R Markdown, with the **blastula** package can render an Rmd document to the email body and send the email.
-

Collaboration

Collaboration and team messaging applications are a popular way of communicating with coworkers and friends at work and home. These applications don't utilize all of Markdown's features, but the features they do provide are fairly useful. For example, the ability to bold and italicize text without using the WYSIWYG interface is pretty handy.

Some examples:

- Slack
- Discord
- Mattermost
- HackMD
- Collaborative MarkDown Editor
- Socrates.io
- StackEdit
- Draft
- Quiver
- FirePad

Technical Documentation

Markdown is a natural fit for technical documentation. Companies like GitHub are increasingly switching to Markdown for their documentation — check out their blog post about how they migrated their Markdown-formatted documentation to Jekyll. If you write documentation for a product or service, take a look at these handy tools:

- Read the Docs can generate a documentation website from your open source Markdown files. Just connect your GitHub repository to their service and push — Read the Docs does the rest. They also have a service for commercial entities.
- MkDocs is a fast and simple static site generator that's geared towards building project documentation. Documentation source files are written in Markdown and configured with a single YAML configuration file. MkDocs has several built in themes, including a port of the Read the Docs documentation theme for use with MkDocs. One of the newest themes is MkDocs Material.
- Docusaurus is a static site generator designed exclusively for creating documentation websites. It supports translations, search, and versioning.
- VuePress is a static site generator powered by Vue and optimized for writing technical documentation.
- Jekyll was mentioned earlier in the section on websites, but it's also a good option for generating a documentation website from Markdown files. If you go this route, be sure to check out the Jekyll documentation theme.
- You probably know pandoc as a magic wand for converting between various markup languages. What you might not know is that pandoc can take a file formatted with Markdown and create attractive HTML slides that work with the Slidy, Slideous, DZSlides, S5, and Reveal.js presentation frameworks. If you prefer LaTeX, you can also output PDF slides using the Beamer package. You'll need to use specific formatting for your slides, but you can add some variables to

control how they behave. You can also change the look and feel of your slides, add pauses between slides, and include speaker notes. Of course, you must have the supporting files for your preferred presentation framework installed on your computer. Pandoc spits out only the raw slide file.

- Hacker Slides is an application for Sandstorm and Sandstorm Oasis that mates Markdown and the Reveal.js slide framework. The slides are simple, but they can be visually striking.

  Craft your slide deck in a two-pane editor in your browser—type in Markdown on the left and see it rendered on the right. When you're ready to present, you can do it from within Sandstorm or get a link that you can share with others to present remotely.

Flavors of Markdown

One of the most confusing aspects of using Markdown is that practically every Markdown application implements a slightly different version of Markdown. These variants of Markdown are commonly referred to as *flavors*. It's your job to master whatever flavor of Markdown your application has implemented.

To wrap your head around the concept of Markdown flavors, it might help to think of them as language dialects. People in Ciudad Juárez speak Spanish just like the people in Barcelona, but there are substantial differences between the dialects used in both cities. The same is true for people using different Markdown applications.

Practically speaking, this means you never know exactly what a company means when they say they support "Markdown." Are they talking about only the basic syntax elements, or all of the basic and extended syntax elements combined, or some arbitrary combination of syntax elements? You won't know until you read the documentation or start using the application.

If you're just starting out, the best advice I can give you is to pick a Markdown application with good Markdown support. That'll go a long way towards maintaining the portability of your Markdown files. You might want to store and use your Markdown files in other applications, and to do that you need to start with an application that provides good support. You can use the tools directory to find an application that fits the bill.

Additional Resources

There are lots of resources you can use to learn Markdown. Here are some other introductory resources:

- John Gruber's Markdown documentation. The original guide written by the creator of Markdown.
- Markdown Tutorial. An open source website that allows you to try Markdown in your web browser.
- Awesome Markdown. A list of Markdown tools and learning resources.
- Typesetting Markdown. A multi-part series that describes an ecosystem for typesetting Markdown documents using pandoc and ConTeXt.
- La guía definitiva de markdown(spanish).

the next level.

Learn Markdown in 60 pages. Designed for both novices and experts, *The Markdown Guide* book is a comprehensive reference that has everything you need to get started and master Markdown syntax.

Get the Book

To learn more Markdown

At the GitHub repository and then enter your email address below to receive new Markdown tutorials via email. No spam!

Stay updated

Markdown Editors for Mac

Here are some of the best Markdown editors for Mac users:

1. Byword
2. Ulysses
3. MacDown

Markdown Editors for Windows

1. Caret
2. ghostwriter
3. Typora

Markdown Editors for Linux

1. Remarkable
2. Haroopad
3. UberWriter

Online Markdown Editors

1. HackMD

Markdown Cheat Sheet

The Markdown Cheat Sheet provides a quick overview of all the Markdown syntax elements. It can't cover every edge case, so if you need more information about any of these elements, refer to the reference guides for basic syntax and extended syntax.

Basic Syntax

These are the elements outlined in John Gruber's original design document. All Markdown applications support these elements.

Heading

H1

H2

H3

Bold

bold text

Italic

italicized text

Blockquote

> *blockquote*

Ordered List

1. First item
2. Second item
3. Third item

Unordered List

- First item
- Second item
- Third item

Code

`code`

Horizontal Rule

Link

[title]

Image

`[alternative text](PATH/image.jpg)`

Extended Syntax

These elements extend the basic syntax by adding additional features. Not all Markdown applications support these elements.

Table

SYNTAX	DESCRIPTION
Header	Title
Paragraph	Text

Fenced Code Block

```
{
  "firstName": "Anxo",
  "lastName": "Sánchez",
  "age": 60
}
```

Footnote

Here's a sentence with a footnote. [1]

Heading ID

My Great Heading {#custom-id}

Definition List

```
bar
: definition
foo
: definition
```

Strikethrough

`~~The world is flat.~~`

Task List

```
Write the press release

- [ ] Update the website
- [ ] Contact the media
```

Basic Syntax

The Markdown elements outlined in John Gruber's design document. Nearly all Markdown applications support the basic syntax outlined in John Gruber's original design document. There are minor variations and discrepancies between Markdown processors, those are noted inline wherever possible.

Headings

To create a heading, add number signs (`#`) in front of a word or phrase. The number of number signs you use should correspond to the heading level. For example, to create a heading level three (`<h3>`), use three number signs (e.g., `### My Header`).

MARKDOWN	HTML	RENDERED OUTPUT
# Heading level 1	`<h1>Heading level 1</h1>`	Heading level 1
## Heading level 2	`<h2>Heading level 2</h2>`	Heading level 2
### Heading level 3	`<h3>Heading level 3</h3>`	Heading level 3
#### Heading level 4	`<h4>Heading level 4</h4>`	Heading level 4
##### Heading level 5	`<h5>Heading level 5</h5>`	Heading level 5
###### Heading level 6	`<h6>Heading level 6</h6>`	Heading level 6

Alternate Syntax

Alternatively, on the line below the text, add any number of `==` characters for heading level 1 or `--` characters for heading level 2.

MARKDOWN	HTML	RENDERED OUTPUT
Heading level 1 ==========	`<h1>Heading level 1</h1>`	Heading level 1
Heading level 2 ----------	`<h2>Heading level 2</h2>`	Heading level 2

Heading Best Practices

Markdown applications don't agree on how to handle a missing space between the number signs (#) and the heading name. For compatibility, always put a space between the number signs and the heading name.

✔ DO THIS	✕ DON'T DO THIS
`#` This is a Heading	`#`This is a Heading

Paragraphs

To create paragraphs, use a blank line to separate one or more lines of text.

MARKDOWN	HTML	RENDERED OUTPUT
I really like using Markdown. I think I'll use it to format all of my documents from now on.	`<p>`I really like using Markdown.`</p>` `<p>`I think I'll use it to format all of my documents from now on.`</p>`	I really like using Markdown. I think I'll use it to format all of my documents from now on.

Paragraph Best Practices

Unless the paragraph is in a list, don’t indent paragraphs with spaces or tabs.

✔ DO THIS	✕ DON'T DO THIS
Don't put tabs or spaces in front of your paragraphs.	This can result in unexpected formatting problems.
Keep lines left-aligned like this.	Don't add tabs or spaces in front of paragraphs.

Line Breaks

To create a line break (`<br>`), end a line with two or more spaces, and then type return.

MARKDOWN	HTML	RENDERED OUTPUT

MARKDOWN	HTML	RENDERED OUTPUT
This is the first line. And this is the second line.	`<p>`This is the first line. And this is the second line. `</p>`	This is the first line. And this is the second line.

Line Break Best Practices

You can use two or more spaces (commonly referred to as “trailing whitespace”) for line breaks in nearly every Markdown application, but it’s controversial. It’s hard to see trailing whitespace in an editor, and many people accidentally or intentionally put two spaces after every sentence. For this reason, you may want to use something other than trailing whitespace for line breaks. Fortunately, there is another option supported by nearly every Markdown application: the `<br>` HTML tag. For compatibility, use trailing white space or the `<br>` HTML tag at the end of the line.

There are two other options not recommended using. CommonMark and a few other lightweight markup languages let you type a backslash (`\`) at the end of the line, but not all Markdown applications support this, so it isn't a great option from a compatibility perspective. And at least a couple lightweight markup languages don't require anything at the end of the line — just type return and they'll create a line break.

☑ DO THIS	✕ DON'T DO THIS
First line with two spaces after. And the next line.	First line with a backslash after.`\` And the next line.
First line with the HTML tag after.` ` And the next line.	First line with nothing after. And the next line.

Emphasis

You can add emphasis by making text bold or italic.

Bold

To bold text, add two asterisks or underscores before and after a word or phrase. To bold the middle of a word for emphasis, add two asterisks without spaces around the letters.

MARKDOWN	HTML	RENDERED OUTPUT

MARKDOWN	HTML	RENDERED OUTPUT
I just like `**`bold text`**`.	I just like `<strong>`bold text`</strong>`.	I just love **bold text**.
I just like `__`bold text`__`.	I just like `<strong>`bold text`</strong>`.	I just love **bold text**.
Love`**`is`**`bold	Love`<strong>`is`</strong>`bold	Love**is**bold

Bold Best Practices

Markdown applications don't agree on how to handle underscores in the middle of a word. For compatibility, use asterisks to bold the middle of a word for emphasis.

✔ DO THIS	✕ DON'T DO THIS
`Love**is**bold`	`Love__is__bold`

Italic

To italicize text, add one asterisk or underscore before and after a word or phrase. To italicize the middle of a word for emphasis, add one asterisk without spaces around the letters.

MARKDOWN	HTML	RENDERED OUTPUT
Italicized text is the `*`Equation of State`*`	Italicized text is the `<em>`Equation of State`</em>`.	Italicized text is the *Equation of State*
Italicized text is the `_`Equation of State`_`.	Italicized text is the `<em>`Equation of State`</em>`.	Italicized text is the *Equation of State*.
Equation `*`of`*` State	Equation `<em>`of`</em>` State	Equation *of* State

Italic Best Practices

Markdown applications don't agree on how to handle underscores in the middle of a word. For compatibility, use asterisks to italicize the middle of a word for emphasis.

✔ DO THIS	✕ DON'T DO THIS

✅ DO THIS	❌ DON'T DO THIS
Equation`*`of`*`State	Equation_of_State

Bold and Italic

To emphasize text with bold and italics at the same time, add three asterisks or underscores before and after a word or phrase. To bold and italicize the middle of a word for emphasis, add three asterisks without spaces around the letters.

MARKDOWN	HTML	RENDERED OUTPUT
This text is `***`really important`***`.	This text is `<strong><em>`really important`</em></strong>`.	This text is **really important**.
This text is `___`really important`___`.	This text is `<strong><em>`rreally important`</em></strong>`.	This text is **really important**.
This text is `__*`really important`*__`.	This text is `<strong><em>`really important`</em></strong>`.	This text is **really important**.
This text is `**_`really important`_**`.	This text is `<strong><em>`really important`</em></strong>`.	This text is **really important**.
This is really`***`very`***`important text.	This is really`<strong> <em>`very`</em></strong>`important text.	This is really**very**important text.

Bold and Italic Best Practices

Markdown applications don't agree on how to handle underscores in the middle of a word. For compatibility, use asterisks to bold and italicize the middle of a word for emphasis.

✅ DO THIS	❌ DON'T DO THIS
This is really`***`very`***`important text.	This is really`___`very`___`important text.

Blockquotes

To create a blockquote, add a `>` in front of a paragraph.

```
> Esse metuenda quicquam mulcebunt legunt tu tamen.
```

The rendered output looks like this:

> *Esse metuenda quicquam mulcebunt legunt tu tamen.*

Blockquotes with Multiple Paragraphs

Blockquotes can contain multiple paragraphs. Add a `>` on the blank lines between the paragraphs.

```
> Dorothy followed her through many of the beautiful rooms in her
castle.
>
> Lorem markdownum spumantiaque esse messes, suam ingens de Ithaco
spumis tuorum,
> submisit quae trahit in soceri?. Et illi coloribus Numidasque hoc
tu ponit
> medullas.
```

The rendered output looks like this:

> *Dorothy followed her through many of the beautiful rooms in her castle.*
>
> *Lorem markdownum spumantiaque esse messes, suam ingens de Ithaco spumis tuorum, submisit quae trahit in soceri?. Et illi coloribus Numidasque hoc tu ponit medullas.*

Nested Blockquotes

Blockquotes can be nested. Add a `>>` in front of the paragraph you want to nest.

```
> Et requies forsitan manus peragentem leves dat.
>
>> Omnes inde saepe corpore Meleagron eadem Apollineae.
```

The rendered output looks like this:

> *Et requies forsitan manus peragentem leves dat.*
>
> > *Omnes inde saepe corpore Meleagron eadem Apollineae.*

Blockquotes with Other Elements

Blockquotes can contain other Markdown formatted elements. Not all elements can be used — you'll need to experiment to see which ones work.

```
#### Lorem markdownum, depositoque lucoque dentibus iuvenale, sede
flammis, memores.
> Incana ture quae hunc, volvitur artem.
> Porrexit alii; ora undis aestu, o positaeque forte constitit
aurum, mortales
> leves ipse. Tori praesuta, robora.

- Apertas ista umbra semina: fronti si patrio Othrys quos.
- Apertum Phlegyis neque redeunt mutavit praestanti, oculos
agitante et petitur remansit
 equosque quae *tempora ut fuerit* __non carmine__.
```

The rendered output looks like this:

> ***Lorem markdownum, depositoque lucoque dentibus iuvenale, sede flammis, memores.***
>
> *Incana ture quae hunc, volvitur artem.*
> *Porrexit alii; ora undis aestu, o positaeque forte constitit aurum, mortales leves ipse. Tori praesuta, robora.*
>
> - *Apertas ista umbra semina: fronti si patrio Othrys quos.*
> - *Apertum Phlegyis neque redeunt mutavit praestanti, oculos agitante et petitur remansit*
>   *equosque quae tempora ut fuerit* ***non carmine****.*

Lists

You can organize items into ordered and unordered lists.

Ordered Lists

To create an ordered list, add line items with numbers followed by periods. The numbers don't have to be in numerical order, but the list should start with the number one.

MARKDOWN	HTML	RENDERED OUTPUT

MARKDOWN	HTML	RENDERED OUTPUT
1. First item 2. Second item 3. Third item 4. Fourth item	1. First item 2. Second item 3. Third item 4. Fourth item	1. First item 2. Second item 3. Third item 4. Fourth item
1. First item 1. Second item 1. Third item 1. Fourth item	1. First item 2. Second item 3. Third item 4. Fourth item	1. First item 1. Second item 1. Third item 1. Fourth item
1. First item 8. Second item 3. Third item 5. Fourth item	1. First item 2. Second item 3. Third item 4. Fourth item	1. First item 8. Second item 3. Third item 5. Fourth item

`

MARKDOWN	HTML	RENDERED OUTPUT
1. First item 2. Second item 3. Third item 1. Indented item 2. Indented item 4. Fourth item	1. First item 2. Second item 3. Third item a. Indented item b. Indented item 4. Fourth item `	1. First item 2. Second item 3. Third item 1. Indented item 2. Indented item 4. Fourth item

Ordered List Best Practices

CommonMark and a few other lightweight markup languages let you use a parenthesis (`)`) as a delimiter (e.g., `1) First item`), but not all Markdown applications support this, so it isn't a great option from a compatibility perspective. For compatibility, use periods only.

✅ DO THIS	❌ DON'T DO THIS
1. First item 2. Second item	1) First item 2) Second item

Unordered Lists

To create an unordered list, add dashes (`-`), asterisks (`*`), or plus signs (`+`) in front of line items. Indent one or more items to create a nested list. `&#9658` U+00BA

MARKDOWN	HTML	RENDERED OUTPUT

MARKDOWN	HTML	RENDERED OUTPUT
- First item - Second item - Third item - Fourth item	• First item • Second item • Third item • Fourth item	* First item * Second item * Third item * Fourth item
First item Second item Third item Fourth item	• First item • Second item • Third item • Fourth item	* First item * Second item * Third item * Fourth item
+ First item + Second item + Third item + Fourth item	• First item • Second item • Third item • Fourth item	* First item * Second item * Third item * Fourth item

`

MARKDOWN	HTML	RENDERED OUTPUT
- First item - Second item - Third item - Indented item - Indented item - Fourth item	• First item • Second item • Third item • Indented item • Indented item • Fourth item	* First item * Second item * Third item * Intended item * Intended item * Fourth item

Unordered List Best Practices

Markdown applications don't agree on how to handle different delimiters in the same list. For compatibility, don't mix and match delimiters in the same list — pick one and stick with it.

☑ DO THIS	✕ DON'T DO THIS
- First item- Second item - Third item - Fourth item	+ First item * Second item - Third item + Fourth item

Adding Elements in Lists

To add another element in a list while preserving the continuity of the list, indent the element four spaces or one tab, as shown in the following examples.

Paragraphs

```
*   This is the first list item.
*   Here's the second list item.

    I need to add another paragraph below the second list item.

*   And here's the third list item.
```

The rendered output looks like this:

- This is the first list item.
- Here's the second list item.

  I need to add another paragraph below the second list item.
- And here's the third list item.

Blockquotes

```
*   This is the first list item.
*   Here's the second list item.

    > A blockquote would look great below the second list item.

*   And here's the third list item.
```

The rendered output looks like this:

- This is the first list item.
- Here's the second list item.

  > *A blockquote would look great below the second list item.*
- And here's the third list item.

Code Blocks

Code blocks are normally indented four spaces or one tab. When they're in a list, indent them eight spaces or two tabs.

```
'''


def greet(name):
    """
    This function greets to
    the person passed in as
    a parameter
    """
    print("Hello, " + name + ". Good morning!")

greet('Paul')

'''
```

The rendered output looks like this:

```
def greet(name):    """     This function greets to     the person passed in as     a parameter     """ print("Hello, " + name + ". Good morning!")greet('Paul')`
```

Images

```
1. Markdown code for presenting the University of Vigo Logo svg file.2. Open the file containing the Linux mascot or tyhe url (right button copy image link   in the web browser).3. Markdown code to inclde the image.   ![University of Vigo Logo](https://www.uvigo.gal/themes/custom/uvigo/logo.svg)
```

The rendered output looks like this:

1. Markdown code for presenting the University of Vigo Logo svg file.
2. Open the file containing the Linux mascot or tyhe url (right button copy image link
   in the web browser).

3. Markdown code to inclde the image.

Lists

You can nest an unordered list in an ordered list, or vice versa.

```
1. First item2. Second item3. Third item    - Indented item    - Indented item4. Fourth item
```

The rendered output looks like this:

1. First item
2. Second item
3. Third item
    - Indented item
    - Indented item
4. Fourth item

Code

To denote a word or phrase as code, enclose it in backticks (```).

MARKDOWN	HTML	RENDERED OUTPUT
At the command prompt, type ` nano`	At the command prompt, type <code>nano</code>.	At the command prompt, type nano.

Escaping backticks

If the word or phrase you want to denote as code includes one or more backticks, you can escape it by enclosing the word or phrase in double backticks (``).

MARKDOWN	HTML	RENDERED OUTPUT
`Use ``code`` in your Markdown file.`	<code>Use `code` in your Markdown file.</code>	Use `code` in your Markdown file.

Code Blocks

To create code blocks, indent every line of the block by at least four spaces or one tab.

```
<html>

    <head>

  </head>

</html>
```

The rendered output looks like this:

```
<html>  <head>  </head></html>
```

Note: To create code blocks without indenting lines, use fenced code blocks.

```
def say_hello():
 print("Hello from a function")
```

The rendered output looks like this:

```
def say_hello(): print("Hello from a function")
```

Horizontal Rules

To create a horizontal rule, use three or more asterisks (`***`), dashes (`---`), or underscores (`___`) on a line by themselves.

```
***---_____________________
```

The rendered output of all three looks identical:

Horizontal Rule Best Practices

For compatibility, put blank lines before and after horizontal rules.

☑ DO THIS	✕ DON'T DO THIS
Try to put a blank line before...	Without blank lines, this would be a heading.
...and after a horizontal rule.	Don't do this!

Links

To create a link, enclose the link text in brackets (e.g., [Universidade de Vigo]) and then follow it immediately with the URL in parentheses (e.g., (https://uvigo.gal)).

```
My favorite University is [Universidade de Vigo]
(https://uvigo.gal).
```

The rendered output looks like this:

My favorite University is Universidade de Vigo.

Adding Titles

You can optionally add a title for a link. This will appear as a tooltip when the user hovers over the link. To add a title, enclose it in parentheses after the URL.

```
My favorite search engine is [Duck Duck Go](https://duckduckgo.com
"The best search engine for privacy").
```

The rendered output looks like this:

My favorite search engine is Duck Duck Go.

URLs and Email Addresses

To quickly turn a URL or email address into a link, enclose it in angle brackets.

```
<informacion@uvigo.es>
```

The rendered output looks like this:

http://eqea.uvigo.gal
Informació Universidade de Vigo

Formatting Links

To emphasize links, add asterisks before and after the brackets and parentheses.

To denote links as code, add backticks in the brackets.

```
I love supporting the **[Celta de Vigo](https://rccelta.es)**.See
the section on [`code`](#code).
```

The rendered output looks like this:

I love supporting the **Celta de Vigo**.
See the section on `code`.

Reference-style Links

Reference-style links are a special kind of link that make URLs easier to display and read in Markdown. Reference-style links are constructed in two parts: the part you keep inline with your text and the part you store somewhere else in the file to keep the text easy to read.

Formatting the First Part of the Link

The first part of a reference-style link is formatted with two sets of brackets. The first set of brackets surrounds the text that should appear linked. The second set of brackets displays a label used to point to the link you're storing elsewhere in your document.

Although not required, you can include a space between the first and second set of brackets. The label in the second set of brackets is not case sensitive and can include letters, numbers, spaces, or punctuation.

This means the following example formats are roughly equivalent for the first part of the link:

- `[A. Sánchez][1]`

- [A. Sánchez] [1]

Formatting the Second Part of the Link

The second part of a reference-style link is formatted with the following attributes:

1. The label, in brackets, followed immediately by a colon and at least one space (e.g., `[label]:`).
2. The URL for the link, which you can optionally enclose in angle brackets.
3. The optional title for the link, which you can enclose in double quotes, single quotes, or parentheses.

This means the following example formats are all roughly equivalent for the second part of the link:

- `[1]: https://https://en.wikipedia.org/wiki/Vigo`
- `[1]: https://https://en.wikipedia.org/wiki/Vigo "Vigo City"`
- `[1]: https://en.wikipedia.org/wiki/Vigo 'Vigo City'`
- `[1]: https://en.wikipedia.org/wiki/Vigo (Vigo City)`
- `[1]: <https://en.wikipedia.org/wiki/Vigo> "Vigo City"`
- `[1]: <https://en.wikipedia.org/wiki/Vigo> 'Vigo City'`
- `[1]: <https://en.wikipedia.org/wiki/Vigo> (Vigo City)`

You can place this second part of the link anywhere in your Markdown document. Some people place them immediately after the paragraph in which they appear while other people place them at the end of the document (like endnotes or footnotes).

An Example Putting the Parts Together

Say you add a URL as a standard URL link to a paragraph and it looks like this in Markdown:

Vigo (/ˈviːgoʊ/, Galician: [ˈbiɣʊ], locally [ˈbiðʊ], Spanish: [ˈbiɣo]) is an industrial city and municipality in the province of Pontevedra, within the autonomous community of Galicia, Spain. Located in the northwest of the Iberian Peninsula, it sits on the southern shore of an inlet of the Atlantic Ocean, the Ria de Vigo, the southernmost of the so-called Rías Baixas.
[Vigo City at Wikipedia](httpshttps://en.wikipedia.org/wiki/Vigo (Vigo City).

Though it may point to interesting additional information, the URL as displayed really doesn't add much to the existing raw text other than making it harder to read. To fix that, you could format the URL like this instead:

```
In a hole in the ground there lived a hobbit. Not a nasty, dirty,
wet hole, filled with the endsof worms and an oozy smell, nor yet a
dry, bare, sandy hole with nothing in it to sit down on or toeat:
it was a [hobbit-hole][1], and that means comfort.[1]:
<https://en.wikipedia.org/wiki/Hobbit#Lifestyle> "Hobbit
lifestyles"
```

In both instances above, the rendered output would be identical:

> *In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.*

and the HTML for the link would be:

```
<a href="https://en.wikipedia.org/wiki/Hobbit#Lifestyle"
title="Hobbit lifestyles">hobbit-hole</a>
```

Link Best Practices

Markdown applications don't agree on how to handle spaces in the middle of a URL. For compatibility, try to URL encode any spaces with `%20`.

☑ DO THIS	✕ DON'T DO THIS
`[link]` `(https://www.example.com/my%20great%20page)`	`[link]` `(https://www.example.com/my great page)`

Images

To add an image, add an exclamation mark (`!`), followed by alt text in brackets, and the path or URL to the image asset in parentheses. You can optionally add a title after the URL in the parentheses.

```
![Philadelphia's Magic Gardens. This place was so cool!]
(/assets/images/philly-magic-gardens.jpg "Philadelphia's Magic
Gardens")
```

The rendered output looks like this:

Linking Images

To add a link to an image, enclose the Markdown for the image in brackets, and then add the link in parentheses.

```
![Catedral de Santiago de Compostela agosto 2018 (cropped).jpg]
(https://upload.wikimedia.org/wikipedia/commons/thumb/a/a3/Catedral
_de_Santiago_de_Compostela_agosto_2018_%28cropped%29.jpg/1280px-
Catedral_de_Santiago_de_Compostela_agosto_2018_%28cropped%29.jpg)
```

The rendered output looks like this:

Escaping Characters

To display a literal character that would otherwise be used to format text in a Markdown document, add a backslash (\) in front of the character.

```
\* Without the backslash, this would be a bullet in an unordered
list.
```

The rendered output looks like this:

* Without the backslash, this would be a bullet in an unordered list.

Characters You Can Escape

You can use a backslash to escape the following characters.

CHARACTER	NAME
\	backslash
`	backtick (see also escaping backticks in code)
*	asterisk
_	underscore
{ }	curly braces
[ ]	brackets
< >	angle brackets
( )	parentheses
#	pound sign
+	plus sign
-	minus sign (hyphen)
.	dot
!	exclamation mark
|	pipe (see also escaping pipe in tables)

HTML

Many Markdown applications allow you to use HTML tags in Markdown-formatted text. This is helpful if you prefer certain HTML tags to Markdown syntax. For example, some people find it easier to use HTML tags for images. Using HTML is also helpful when you need to change the attributes of an element, like specifying the color of text or changing the width of an image.

To use HTML, place the tags in the text of your Markdown-formatted file.

```
This **word** is bold. This <em>word</em> is italic.
```

The rendered output looks like this:

This **word** is bold. This *word* is italic.

HTML Best Practices

For security reasons, not all Markdown applications support HTML in Markdown documents. When in doubt, check your Markdown application's documentation. Some applications support only a subset of HTML tags.

Use blank lines to separate block-level HTML elements like `<div>`, `<table>`, `<pre>`, and `<p>` from the surrounding content. Try not to indent the tags with tabs or spaces — that can interfere with the formatting.

You can't use Markdown syntax inside block-level HTML tags. For example, `<p>italic and **bold**</p>` won't work.

Markdown dialects

One of the problems (or advantages) of markdown is the number of dialects generated last years. THere are several things you cant, and cannot do depending on the markdown dialect you choose. In the next table you can see a selection of dialects and features of some of the most popular markdown variations that exist now (see https://github.com/rhythmus/markdown-resources/blob/master/markdown-dialects.yml) or the wikipedia markdown article.

MARKDOWN DIALECT	SUPPORTED FEATURES
Original (markdown)	headers (six levels) headers (six levels) paragraphs unordered lists ordered lists nested lists (?) blockquotes block code inline code inline format emphasize inline format strong auto links inline links referenced links inline images block level images referenced images horizontal rule mplementations: Perl

MARKDOWN DIALECT	SUPPORTED FEATURES
CommonMark	*plain-markdown implementations: Javascript, C
Madoko	*plain-markdown # plus some *may* more… implementations: Koka
GitHub Flavored Markdown (GFM)	*plain-markdown fenced code blocks syntax highlighting task Lists emoji autocomplete
Stack Overflow Markdown flavor	Sama as Github?
Pandoc Markdown	implementations: Haskell
MultiMarkdown	*plain-markdown tables citations and bibliography (using BibTeX) math automatic cross-referencing ability smart typography (multiple languages) image attributes captions (tables and images) definition lists glossary entries (LaTeX only) document metadata (title, author, etc.) footnotes implementations: C
Markdown Extra	*plain-markdown inline html Markdown inside html blocksheader ID attributefenced code blockstables definition lists footnotes abbreviations emphasis backslash escapes<br/
Maruku	implementations: Ruby

MARKDOWN DIALECT	SUPPORTED FEATURES
Strictdown	*plain-markdown # (with a few differences) table of contents simple code blocks fenced code blocks definition lists tables deletions literal text code snippets footnotes abbreviations anchors and inline references advanced reference links block divider sub-indented elements macros

A nighmare isn't it?.

Most of the complains the basic syntax of markdown original. The advanced features can be enabled by using a lightweight markup language that builds upon the basic Markdown syntax, or by adding an extension to a compatible Markdown processor.

Markdown software

Not all Markdown applications support extended syntax elements. You'll need to check whether or not the lightweight markup language your application is using supports the extended syntax elements you want to use. If it doesn't, it may still be possible to enable extensions in your Markdown processor. For example,Typora is an excellent processor for Github flavor but, if you are a high math syntax use, probablily you may use Jupyter Lab.

There are several lightweight markup languages that are *supersets* of Markdown. They include Gruber's basic syntax and build upon it by adding additional elements like tables, code blocks, syntax highlighting, URL auto-linking, and footnotes. Many of the most popular Markdown applications use one of the following lightweight markup languages:

- CommonMark
- GitHub Flavored Markdown (GFM)
- Markdown Extra
- MultiMarkdown

- R Markdown

Tables

To add a table, use three or more hyphens (`---`) to create each column's header, and use pipes (`|`) to separate each column. You can optionally add pipes on either end of the table.

```
| Syntax      | Description |
| ----------- | ----------- |
| Header      | Title       |
| Paragraph   | Text        |
```

The rendered output looks like this:

SYNTAX	DESCRIPTION
Header	Title
Paragraph	Text

Cell widths can vary, as shown below. The rendered output will look the same.

```
| Syntax | Description |
| --- | ----------- |
| Header | Title |
| Paragraph | Text |
```

Tip: Creating tables with hyphens and pipes can be tedious. To speed up the process, try using the Markdown Tables Generator. Build a table using the graphical interface, and then copy the generated Markdown-formatted text into your file.

Alignment

You can align text in the columns to the left, right, or center by adding a colon (`:`) to the left, right, or on both side of the hyphens within the header row.

```
| Syntax      | Description | Test Text     |
| :---        |    :----:   |          ---: |
| Header      | Title       | Here's this   |
| Paragraph   | Text        | And more      |
```

The rendered output looks like this:

SYNTAX	DESCRIPTION	TEST TEXT
Header	Title	Here's this
Paragraph	Text	And more

Formatting Text in Tables

You can format the text within tables. For example, you can add links, code (words or phrases in backticks (```) only, not code blocks), and emphasis.

You can't add headings, blockquotes, lists, horizontal rules, images, or HTML tags.

Escaping Pipe Characters in Tables

You can display a pipe (`|`) character in a table by using its HTML character code (`|`).

Fenced Code Blocks

The basic Markdown syntax allows you to create code blocks by indenting lines by four spaces or one tab. If you find that inconvenient, try using fenced code blocks. Depending on your Markdown processor or editor, you'll use three backticks (```) or three tildes (~~~`) on the lines before and after the code block. The best part? You don't have to indent any lines!

```
```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25
}
```
```

The rendered output looks like this:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25
}
```

Tip: Need to display backticks inside a code block? See this section to learn how to escape them.

Syntax Highlighting

Many Markdown processors support syntax highlighting for fenced code blocks. This feature allows you to add color highlighting for whatever language your code was written in. To add syntax highlighting, specify a language next to the backticks before the fenced code block.

````
```json
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25
}
```
````

The rendered output looks like this:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25
}
```

Footnotes

Footnotes allow you to add notes and references without cluttering the body of the document. When you create a footnote, a superscript number with a link appears where you added the footnote reference. Readers can click the link to jump to the content of the footnote at the bottom of the page.

To create a footnote reference, add a caret and an identifier inside brackets (`[^1]`). Identifiers can be numbers or words, but they can't contain spaces or tabs. Identifiers only correlate the footnote reference with the footnote itself — in the output, footnotes are numbered sequentially.

Add the footnote using another caret and number inside brackets with a colon and text (`[^1]: My footnote.`). You don't have to put footnotes at the end of the document. You can put them anywhere except inside other elements like lists, block quotes, and tables.

```
Here's a simple footnote,[^1] and here's a longer one.[^bignote]

[^1]: This is the first footnote.

[^bignote]: Here's one with multiple paragraphs and code.

    Indent paragraphs to include them in the footnote.

    `{ my code }`

    Add as many paragraphs as you like.
```

The rendered output looks like this:

Here's a simple footnote,[1] and here's a longer one.[2]

1. This is the first footnote. ↩
2. Here's one with multiple paragraphs and code.

   Indent paragraphs to include them in the footnote.

   `{ my code }`

   Add as many paragraphs as you like. ↩

Heading IDs

Many Markdown processors support custom IDs for headings — some Markdown processors automatically add them. Adding custom IDs allows you to link directly to headings and modify them with CSS. To add a custom heading ID, enclose the custom ID in curly braces on the same line as the heading.

```
### My Great Heading {#custom-id}
```

The HTML looks like this:

```
<h3 id="custom-id">My Great Heading</h3>
```

Linking to Heading IDs

You can link to headings with custom IDs in the file by creating a standard link with a number sign (`#`) followed by the custom heading ID.

MARKDOWN	HTML	RENDERED OUTPUT
`[Heading IDs](#heading-ids)`	`<a href="#heading-ids">Heading IDs</a>`	Heading IDs

Other websites can link to the heading by adding the custom heading ID to the full URL of the webpage (e.g, `[Heading IDs](https://www.markdownguide.org/extended-syntax#heading-ids)`).

Definition Lists

Some Markdown processors allow you to create *definition lists* of terms and their corresponding definitions. To create a definition list, type the term on the first line. On the next line, type a colon followed by a space and the definition.

```
First Term: This is the definition of the first term.Second Term: This is one definition of the second term.: This is another definition of the second term.
```

The HTML looks like this:

```
<dl>  <dt>First Term</dt>  <dd>This is the definition of the first term.</dd>  <dt>Second Term</dt>  <dd>This is one definition of the second term. </dd>  <dd>This is another definition of the second term.</dd></dl>
```

The rendered output looks like this:

- First Term

  This is the definition of the first term.

- Second Term

  This is one definition of the second term.

  This is another definition of the second term.

Strikethrough

You can strikethrough words by putting a horizontal line through the center of them. The result looks ~~like this~~. This feature allows you to indicate that certain words are a mistake not meant for inclusion in the document. To strikethrough words, use two tilde symbols (`~~`) before and after the words.

```
~~The world is flat.~~ We now know that the world is round.
```

The rendered output looks like this:

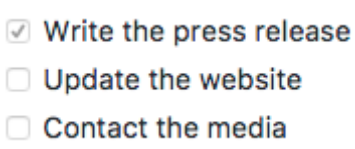
~~The world is flat.~~ We now know that the world is round.

Task Lists

Task lists allow you to create a list of items with checkboxes. In Markdown applications that support task lists, checkboxes will be displayed next to the content. To create a task list, add dashes (`-`) and brackets with a space (`[ ]`) in front of task list items. To select a checkbox, add an `x` in between the brackets (`[x]`).

```
- [x] Write the press release- [ ] Update the website- [ ] Contact
the media
```

The rendered output looks like this:



Emoji

There are two ways to add emoji to Markdown files: copy and paste the emoji into your Markdown-formatted text, or type *emoji shortcodes*.

Copying and Pasting Emoji

In most cases, you can simply copy an emoji from a source like Emojipedia and paste it into your document. Many Markdown applications will automatically display the emoji in the Markdown-formatted text. The HTML and PDF files you export from your Markdown application should display the emoji.

Tip: If you're using a static site generator, make sure you encode HTML pages as UTF-8.

Using Emoji Shortcodes

Some Markdown applications allow you to insert emoji by typing emoji shortcodes. These begin and end with a colon and include the name of an emoji.

```
Gone camping! :tent: Be back soon.That is so funny! :joy:
```

The rendered output looks like this:

Gone camping! ⛺ Be back soon.

That is so funny! 😂

Note: You can use this list of emoji shortcodes, but keep in mind that emoji shortcodes vary from application to application. Refer to your Markdown application's documentation for more information.

Automatic URL Linking

Many Markdown processors automatically turn URLs into links. That means if you type http://www.example.com, your Markdown processor will automatically turn it into a link even though you haven't used brackets.

```
http://www.example.com
```

The rendered output looks like this:

http://www.example.com

Disabling Automatic URL Linking

If you don't want a URL to be automatically linked, you can remove the link by denoting the URL as code with backticks.

```
`http://www.example.com`
```

The rendered output looks like this:

```
http://www.example.com
```

Scientific articles

This is the subject of a separate tutorial.

Organize Writing with Markdown

If you use yor computer to write blog articles, documentation for work, emails, books that will never be published. But all of this writing on the computer presents some problems. For one, how do you keep all those documents organized? Should you write them all in Microsoft Word? What about notebook apps likes Evernote or OneNote?. How should we organize, version, and format our written documents so they will be the most useful for us in the long run?

Formatting

The main problem creating teaching documents is that you often don't know for sure what final form your writing will end up in. For example, teachers often solve probles in the classroom and make presentations with nice diagrams which, lately, could be used in a lecture document, in pdf. But in many cases, take that diagram to the original pdf document its is not easy. In the case of writting blogs or creating moodle questionaries or so, should be again useful that diagram but, in too many case we don't do that beacuse of we don't remeber where was it, or need to chase fonsts and sizes, etc. Most of us use Microsoft Powerpoint to make slides, Microsoft Word to make documents (which after editing save as PDF format), Inkscape for Illustrations ando so. It also depends of the first use the creation software, for example when you write a blog you uses wordpress or blogger to make a draft, but if you need a presentation you normally use Powerpoint as the first program. After that the format conversion is so ugly that you'll probably make

the diiagram again before try to reuse it. You realize that what you need is to separate content and formatting.

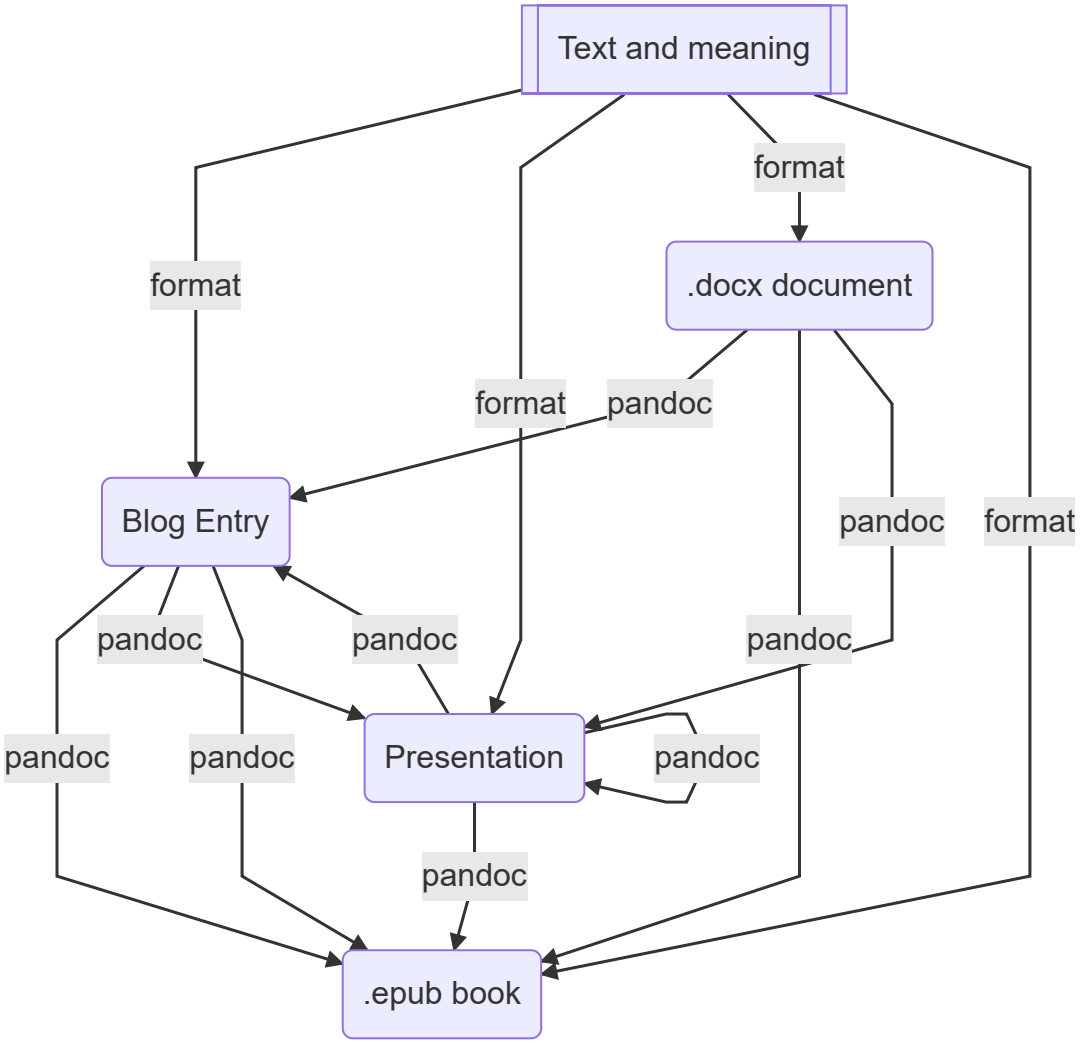
The problem goes worst when you consider that most writing doesn't you make is based in your past job. We all have frequently adapted lectures notes into presentations, laboratory annotations into articles (think about bibliography references), and papers into PowerPoint presentations. The proccess of that adaptation from one format to another is an arduous work, removing all of the formatting and then reformatting it for a different document type one time and another. But, in my opinion the worst thing is when I question myself about where is or which is the master copy of my really needed document?. Markdown is a better way to do the things.

The Way of Working

The best way to separate contetn and format is to use plain text as default format. The advantages when you are:

- the text can be written in almost any (old or new) computer with a simple text editor.
- You don't waste your time fomatting (which in Word is represents a lot of time choosing font, adding headings, and selecting margins an styles.
- You are more focussed in the meaning of the text than in its format or appearance.
- You can eliminate the clutter of having thousands of formatted Word documents dusting your computer desktop
- You can remove the decision of having to decide the form of the document before writing it, which can promote the creation of a "master documents" repository (format-agnostic)

The documetnformatting should be the very last step of the process, not the first. One of the final stages in publishing a book is typesetting. The manuscript needs to go through several rounds of editing, revisions, and rewriting before they even consider what font it will use and the style of the chapter headings. The same thing should be for our personal writing of anu kind of document: content first, formatting last, like can be seen in this flux diagram:



By separating content from formatting we turn writing into a two-stage process. This gives us greater agility in how we deploy that written content later on.

Implementation

We are telling you a posibble markdown authoring perpective, but we higly recommennd you to investigate abroad the vaste markdown application ecosystem to choose yoyr preffeared configuration.

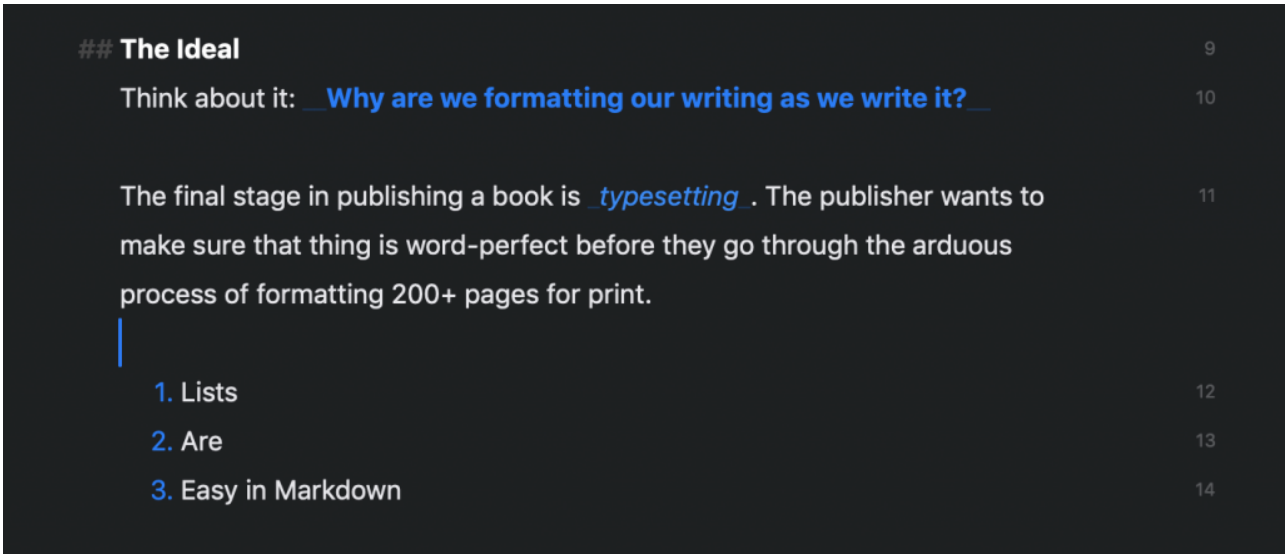
As we said before, Markdown is a formatting syntax which allows you to write documents in plain-text and later render them with basic formatting. From a Markdown document, you can output your writing as HTML, PDF, DOCX, etc.

We can use to edit markdown any plain tex editor, like Notepad, or a specialized software (mainly with widgets to do the live more easy).

If you are used to write programming code (like me) you robably want to use your every-day editor, in my case visual studio code

Begin Your Writing in Markdown

What Markdown adds to your plain-text writing is some simple syntax to indicate what different elements in your document are. For example, you can mark something as a level-one heading by using a hashtag. Level two headings begin with two hashtags and so forth. Bullet lists just need to begin with a hyphen or asterisk. To see all of the options check out this comprehensive Markdown cheatsheet.

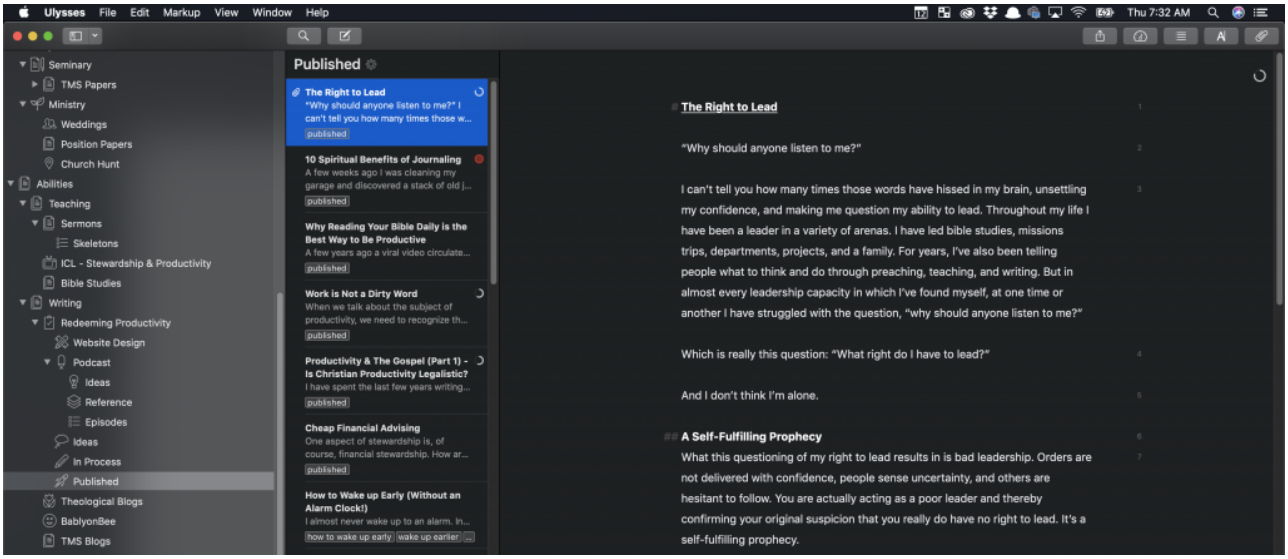


Here's a screenshot of an early draft of this article done in Markdown using Ulysses.

When you finally render your document, the Markdown app formats those headings, lists, and boldface according to whatever rules you've asked it to apply in the target document type. More on rendering Markdown in point three. But before we render, we need to get organized. Markdown can help with this too.

2 – Use a Markdown Editor to Organize

Since Markdown documents are plain-text you could just store them in folders on your computer. But I much prefer to use a Markdown editor to organize my documents.

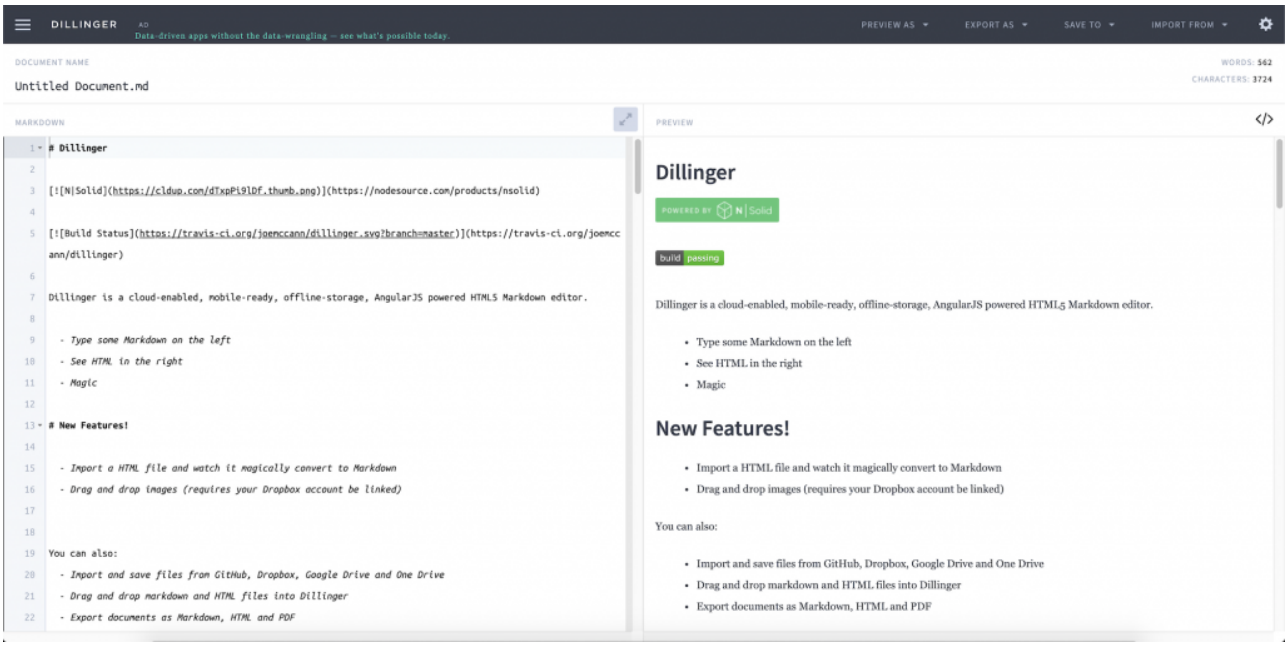


This is Ulysses. The left column contains all the folders and subfolders I use to organize my writing and ideas.

I use Ulysses for this since I like its infinite folder structure. But there are tons of other Markdown editors which also have organizational abilities. And most of these sync across your devices. So, you can jot stuff down from your phone.

- Drafts is a great lightweight option.
- iA Writer works in Windows, Mac, iOS, and Android.
- Bear is a beautiful option for the Apple ecosystem
- And Caret is another good cross-platform option.

There are also lots of online Markdown editors like Dillinger. Or HackMD which allow you to work on Markdown documents collaboratively *a la* Google Docs.



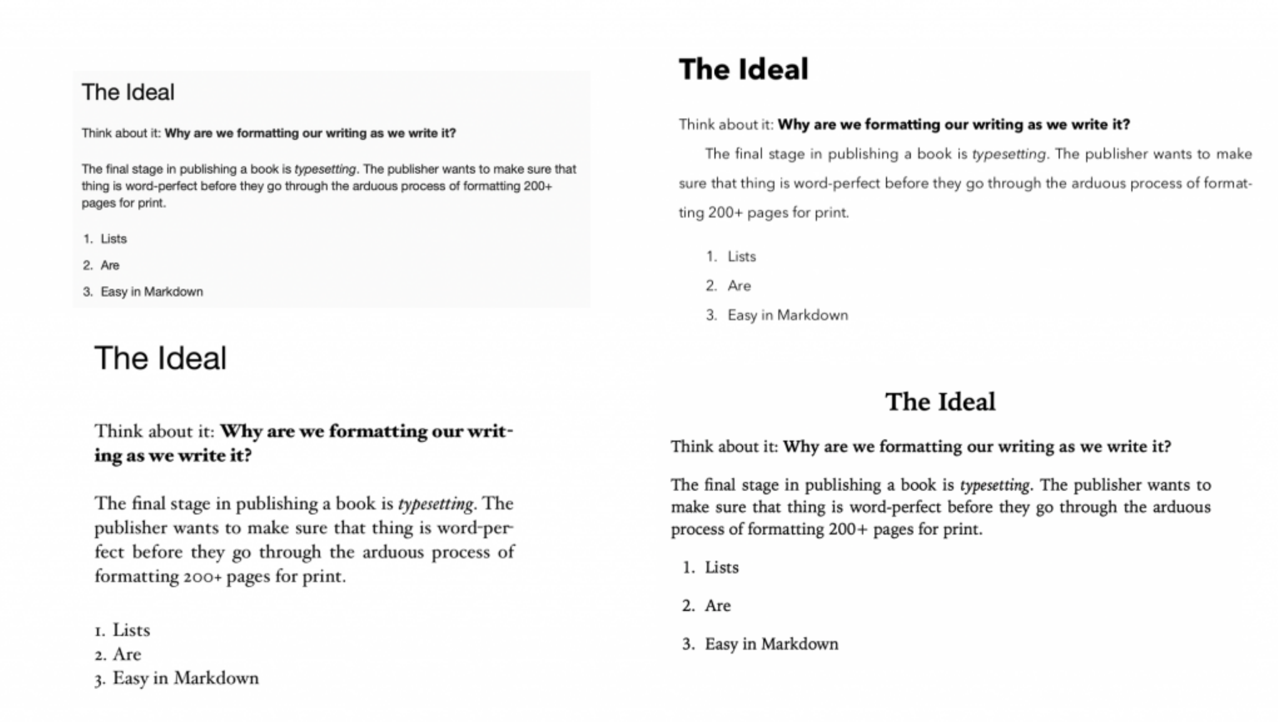
is a free online Markdown editor you can use to output Markdown in a variety of formats. It also lets you preview the results, so it's a good way to learn Markdown.

Another benefit of organizing your writing this way is that your Markdown editor can act like a notebook, similar to how people use Evernote or Microsoft OneNote. For me, that means when I have an idea, I just open up Ulysses and start typing. And if I want to find an old written document, I know it's somewhere in Ulysses. Using a Markdown editor keeps your writing organized and efficient.

So, now you're writing your master documents in Markdown and you're organizing them with a Markdown editor, but what do you do when it's time to output that writing to a specific format and apply styling?

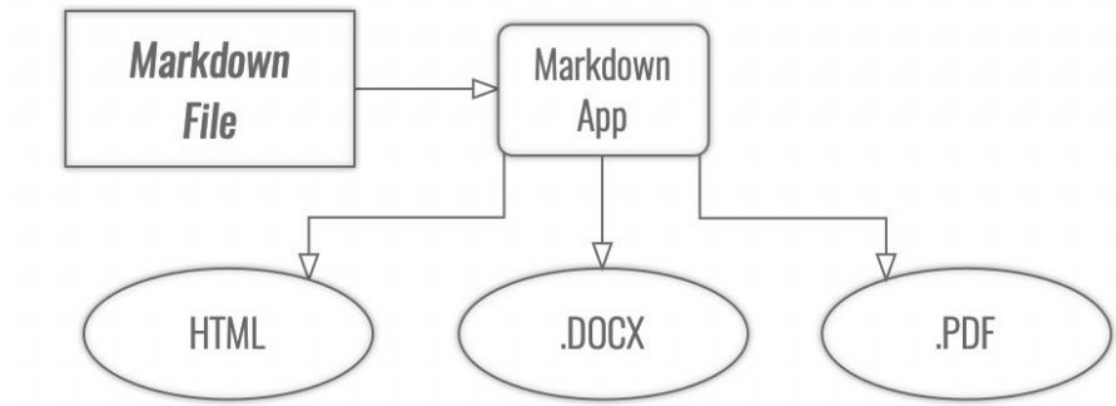
3 – Render the Markdown and Finish Formatting

The final step with any Markdown document is rendering it to its final format. Often for me, that means publishing my blog articles as HTML. The Markdown editor does all the work of converting my headings, links, and images to HTML formatting and sticking that into a post on WordPress.



The is the example from above rendered to four different formats and document types using temples I've setup. You can get pretty fancy with this. For example, in seminary, I used an academic papers template to render my markdown as a .docx file with the correct font, heading styles, and even footnotes.

But if I wanted to use that blog post as a handout for a Bible study, I go back to my Markdown document in my markdown editor and I output it as a .docx file. It does all the simple formatting for me and any additional formatting I complete in Microsoft Word.



Markdown files are plain text, but you'll need a Markdown editor to render those files to different formats.

Conclusion

That's the beauty of Markdown, it gives you a better way to organize, version, and format your written documents.

Another side benefit of being familiar with markdown is that even though it's been around since 2004, it keeps showing up in new places. WordPress, for example, which I use to host this blog, recently added the ability to use markdown in the text editor. All of Atlassian's products use it natively and GitHub has used it for years on their platform. Don't be surprised if you see it show up in even more places.

Because of its simplicity and benefits, Markdown isn't going anywhere any time soon. So, why not learn it and take advantage of those benefits now?

Markdown Here Cheatsheet

This is a Github-flavored Markdown info page cheat sheet. This is specifically version of Github-flavored Markdown. You can play around with Markdown on our [live demo page(http://www.markdown-here.com/livedemo.html).

Table of Contents

Headers

```
# H1
## H2
### H3
#### H4
##### H5
###### H6

Alternatively, for H1 and H2, an underline-ish style:

Alt-H1
======

Alt-H2
------
```

H1

H2

H3

H4

H5

H6

Alternatively, for H1 and H2, an underline-ish style:

Alt-H1

Alt-H2

Emphasis

```
Emphasis, aka italics, with *asterisks* or _underscores_.

Strong emphasis, aka bold, with **asterisks** or __underscores__.

Combined emphasis with **asterisks and _underscores_**.

Strikethrough uses two tildes. ~~Scratch this.~~
```

Emphasis, aka italics, with *asterisks* or *underscores*.

Strong emphasis, aka bold, with **asterisks** or **underscores**.

Combined emphasis with **asterisks and *underscores***.

Strikethrough uses two tildes. ~~Scratch this.~~

Lists

```
1. First ordered list item
2. Another item
  * Unordered sub-list.
1. Actual numbers don't matter, just that it's a number
  1. Ordered sub-list
4. And another item.

   Some text that should be aligned with the above item.

* Unordered list can use asterisks
- Or minuses
+ Or pluses
```

1. First ordered list item
2. Another item

- Unordered sub-list.

1. Actual numbers don't matter, just that it's a number
2. Ordered sub-list
3. And another item.

   Some text that should be aligned with the above item.

- Unordered list can use asterisks
- Or minuses
- Or pluses

Links

There are two ways to create links.

```
[I'm an inline-style link](https://www.google.com)
```

```
[I'm a reference-style link][Arbitrary case-insensitive reference
text]

[You can use numbers for reference-style link definitions][1]

Or leave it empty and use the [link text itself]

URLs and URLs in angle brackets will automatically get turned into
links.
http://www.example.com or <http://www.example.com> and sometimes
example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

[arbitrary case-insensitive reference text]:
https://www.mozilla.org
[1]: http://slashdot.org
[link text itself]: http://www.reddit.com
```

`[I'm an inline-style link](https://www.uvigo.gal/)`

`[I'm a reference-style link](https://www.uvigo.gal/)`

`[You can use numbers for reference-style link definitions]`
`(http://uvigo.gal/)`

Or leave it empty and use the `[link text itself](http://www.uvigo.gal/)`

URLs and URLs in angle brackets will automatically get turned into links.
`[http://www.uvigo.gal](http://www.uvigo.gal/)` or `[http://www.uvigo.gal]`
`(http://www.uvigo.gal/)` and sometimes `uvigo.gal (but not on Github, for example).

Some text to show that the reference links can follow later.

Images

```
Here's our logo (hover to see the title text):

Inline-style:
![alt text](https://github.com/adam-p/markdown-
here/raw/master/src/common/images/icon48.png "Logo Title Text 1")

Reference-style:
![alt text][logo]

[logo]: https://github.com/adam-p/markdown-
here/raw/master/src/common/images/icon48.png "Logo Title Text 2"
```

Here's University of Vigo logo):

Inline-style: Universidade Vigo

Reference-style:

Universidade Vigo

Code and Syntax Highlighting

Code blocks are part of the Markdown spec, but syntax highlighting isn't. However, many renderers -- like Github's and *Markdown Here* -- support syntax highlighting. *Markdown Here* supports highlighting for dozens of languages (and not-really-languages, like diffs and HTTP headers); to see the complete list, and how to write the language names, see the highlight.js demo page.

```
Inline `code` has `back-ticks around` it.
```

Inline `code` has `back-ticks around` it.

Blocks of code are either fenced by lines with three back-ticks ```, or are indented with four spaces. I recommend only using the fenced code blocks -- they're easier and only they support syntax highlighting.

````
```javascript
var s = "JavaScript syntax highlighting";
alert(s);
```


```python
s = "Python syntax highlighting"
print s
```


```
No language indicated, so no syntax highlighting.
But let's throw in a <b>tag</b>.
```
var s = "JavaScript syntax highlighting";
alert(s);
s = "Python syntax highlighting"
print s
No language indicated, so no syntax highlighting in Markdown Here
(varies on Github).
But let's throw in a <b>tag</b>.
````

Again, to see what languages are available for highlighting, and how to write those language names, see the highlight.js demo page.

Tables

Tables aren't part of the core Markdown spec, but they are part of GFM and *Markdown Here* supports them. They are an easy way of adding tables to your email -- a task that would otherwise require copy-pasting from another application.

```
Colons can be used to align columns.

| Tables        | Are           | Cool  |
| ------------- |:-------------:| -----:|
| col 3 is      | right-aligned | $1600 |
```

```
| col 2 is      | centered      |   $12 |
| zebra stripes | are neat      |    $1 |

The outer pipes (|) are optional, and you don't need to make the
raw Markdown line up prettily. You can also use inline Markdown.

Markdown | Less | Pretty
--- | --- | ---
*Still* | `renders` | **nicely**
1 | 2 | 3
```

Colons can be used to align columns.

TABLES	ARE	COOL
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline Markdown.

MARKDOWN	LESS	PRETTY
Still	`renders`	**nicely**
1	2	3

Blockquotes

```
> Blockquotes are very handy in email to emulate reply text.> This
line is part of the same quote.Quote break.> This is a very long
line that will still be quoted properly when it wraps. Oh boy let's
keep writing to make sure this is long enough to actually wrap for
everyone. Oh, you can *put* **Markdown** into a blockquote.
```

> *Blockquotes are very handy in email to emulate reply text. This line is part of the same quote.*

Quote break.

> *This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can put **Markdown** into a blockquote.*

Inline HTML

You can also use raw HTML in your Markdown, and it'll mostly work pretty well.

```
<dl>  <dt>Definition list</dt>  <dd>Is something people use sometimes.</dd>  <dt>Markdown in HTML</dt>  <dd>Does *not* work **very** well. Use HTML <em>tags</em>.</dd></dl>
```

- Definition list

  Is something people use sometimes.
- Markdown in HTML

  Does *not* work **very** well. Use HTML *tags*.

Horizontal Rule

```
Three or more...---Hyphens***Asterisks___Underscores
```

Three or more...

Hyphens

Asterisks

Underscores

Line Breaks

My basic recommendation for learning how line breaks work is to experiment and discover -- hit once (i.e., insert one newline), then hit it twice (i.e., insert two newlines), see what happens. You'll soon learn to get what you want. "Markdown Toggle" is your friend.

Here are some things to try out:

```
Here's a line for us to start with.This line is separated from the
one above by two newlines, so it will be a *separate
paragraph*.This line is also a separate paragraph, but...This line
is only separated by a single newline, so it's a separate line in
the *same paragraph*.
```

Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a *separate paragraph*.

This line is also begins a separate paragraph, but...
This line is only separated by a single newline, so it's a separate line in the *same paragraph*.

(Technical note: *Markdown Here* uses GFM line breaks, so there's no need to use MD's two-space line breaks.)

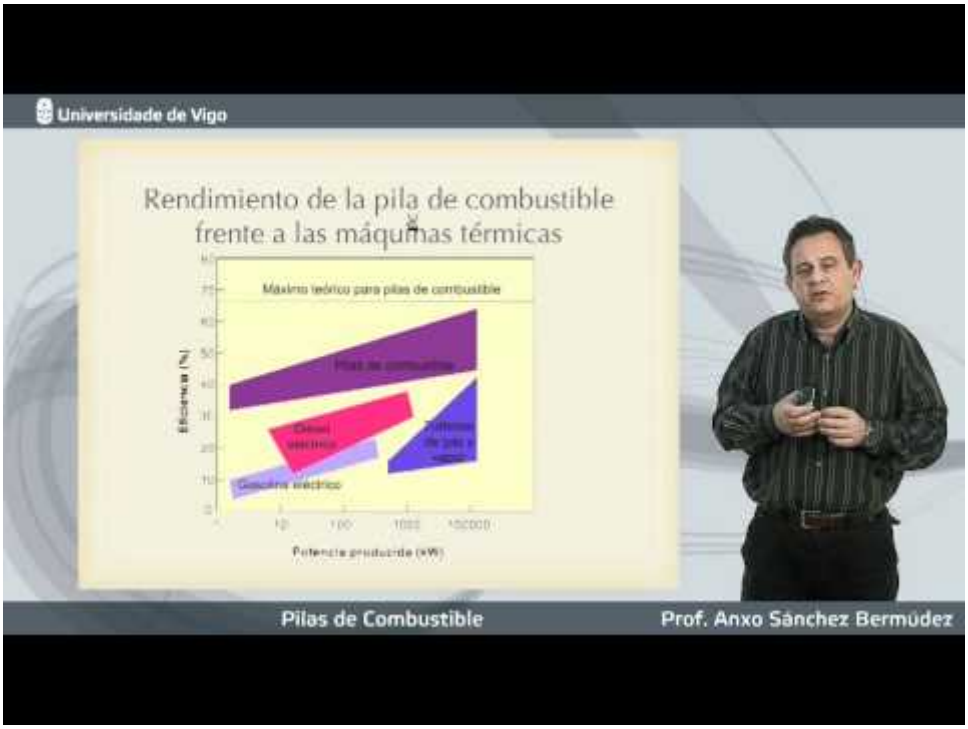
Videos

They can't be added directly but you can add an image with a link to the video like this:

Youtube video:

```
[![IMAGE ALT](http://img.youtube.com/vi/I6EKAH0b69s/0.jpg)]
(http://www.youtube.com/watch?v=I6EKAH0b69s "Pilas de combustible")
```

In pure Markdown, but losing the image sizing and border:



Or, you can use `html` syntax and embed video in a frame:

Video in a frame:

```
<iframe src='https://tv.uvigo.es/iframe/5b5b67b28f4208a36b9bf647'
id='pumukitiframe' frameborder='0' border='0' width='560'
height='315' allowfullscreen></iframe>
```

The result is:

TeX Mathematical Formulae

A full description of TeX math symbols is beyond the scope of this cheatsheet. Here's a good reference, and you can try stuff out on CodeCogs. You can also play with formulae in the Markdown Here options page.

Here are some examples to try out:

```
g = \int_a^b f(x)dx \label{eq:pythagoras}
```

The beginning and ending dollar signs (`$$`) are the delimiters for the TeX markup.

The result is:

$$g = \int_a^b f(x)dx \tag{1}$$

This (1) an example of equation reference.

---

1. This is the footnote. ↩