**What is Markdown?**

An alternative way to style a text document for readability and portability.

**Introduction**

If you are looking for a simple way to create visually appealing text documents without using any fancy editors, check out [Markdown](https://en.wikipedia.org/wiki/Markdown). Invented by John Gruber and Aaron Swartz in 2004, Markdown provides a lightweight syntax to style any text document so that it can be converted to [HTML](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics) for viewing and publishing.

Differences between HTML and Markdown

Markdown translates to HTML, but Markdown is not a replacement for HTML. Markdown’s syntax can be converted to a small subset of HTML tags to do things like format text, add links, display images, and more. You can even incorporate HTML elements inside a Markdown document. To render Markdown in HTML, though, you would need a tool called a Markdown parser (more about this parser later on).

Benefits of Using Markdown

Why would you use Markdown? Compared to HTML, writing text in Markdown is much faster because the syntax is simpler. The [authors of Markdown](https://daringfireball.net/projects/markdown/) intended Markdown to be a language for writing styled text with syntax that is just as easy to read as it is to write.

Imagine that we are writing down some important text on a sheet of paper. If we wanted to emphasize the text we might underline it, twice even! Similarly, in Markdown, we could write:

Title of My Document  
====================

Notice that we have plain text that has a row of equal signs (=) to produce a first-level header, also known as <h1>. On a webpage, the heading will display as such:

Title of My Document

The amount of =s don’t matter so long as there’s at least one and it goes under the text.

For text that’s important, but not as important as a first-level heading, we might just underline it once. In code, we could use a row of hyphens (-), like so:

Sub-Title of My Document  
------------------------

to produce a second-level header, also known as <h2>:

Sub-Title of My Document

Both examples are valid Markdown syntax. It is intuitive and normal for headings to be in either format. Alternatively, you can format headings using the # character. We’ll show more examples of different headings, but for now, here’s an example of valid syntax for a <h1> heading:

# Title of My Document

Furthermore, a Markdown document without any HTML tags can be published as is because its syntax is already made for easy viewing. For example, the following Markdown document has:

styled a heading underlined with equal signs (=),

emphasized week in bold with double asterisks (\*\*),

bulleted a list by prefacing each item with a number and a period,

and marked several lines as separate quotes with an angle bracket (>) per quote.

My Todo List  
============  
   
At the end of this \*\*week\*\*, I plan to:  
   
1. Learn Markdown  
2. Write Markdown  
3. Share a Markdown note  
   
My favorite quote is:  
> If you didn't get it the first time  
> Do not despair  
> Try and try again  
> ~ Anonymous

The HTML equivalent to the above would be:

<h1>My Todo List</h1>  
<p>At the end of this <strong>week</strong>, I plan to:</p>  
<ol>  
  <li>Learn Markdown</li>  
  <li>Write Markdown</li>  
  <li>Share a Markdown note</li>  
</ol>  
<p>My favorite quote is:</p>  
<blockquote>  
  <p>  
    If you didn't get it the first time<br>  
    Do not despair<br>  
    Try and try again<br>  
    ~ Anonymous  
  </p>  
</blockquote>

When viewed on a Markdown parser such as [Stackedit.io](https://stackedit.io/app), you will see this:

or this, based on Codecademy’s parser:

My Todo List

At the end of this week, I plan to:

Learn Markdown

Write Markdown

Share a Markdown note

My favorite quote is:

If you didn’t get it the first time Do not despair Try and try again ~ Anonymous

While both parsers take the same Markdown input, they render slightly different HTML based on their internal configuration and CSS.

When to Use Markdown

After knowing the many benefits of Markdown, you might want to consider using Markdown if you ever find yourself in any of these scenarios:

The only editor available to you supports just plain text.

Time is of the essence and you can’t afford to learn how to use an unfamiliar rich-text editor.

You need to quickly outline your ideas in a structured but presentable manner.

You want your document to be portable so that it can convert to HTML, PDF, EPUB, and/or MOBI.

Markdown Document Extension

Contrary to popular belief, Markdown is not a document format. Therefore, it doesn’t require a strict file extension naming convention, such as .md. As the [official Markdown mailing list](https://superuser.com/questions/249436/file-extension-for-markdown-files) explains, Markdown isn’t meant to take over the format of a file. Any file extension that you would normally use to name your text document such as .txt or .text is appropriate. However, organizations such as [GitHub](https://guides.github.com/features/mastering-markdown/) have a preference to expect Markdown documents to appear with an .md or .markdown extension.

**Markdown Applications**

Since Markdown has gained a lot of popularity, you will find Markdown content being accepted in many applications.

Website publishers that accept Markdown content include [Wordpress.com](https://wordpress.com/support/can-i-use-markdown-on-wordpress-com), [Ghost](https://www.markdownguide.org/tools/ghost/) and [BeakerBrowser](https://beakerbrowser.com/docs/guides/create-a-markdown-site" \t "_blank). Tools such as [Jekyll](https://www.markdownguide.org/tools/jekyll/), [Docusaurus](https://www.markdownguide.org/tools/docusaurus/" \t "_blank) and [MkDocs](https://www.markdownguide.org/tools/mkdocs/" \t "_blank) can convert Markdown documents into a static website geared for technical documentation. Take a look at this [page](https://daringfireball.net/projects/markdown/basics) rendered in HTML, and its text [source](https://daringfireball.net/projects/markdown/basics.text) styled in Markdown.

Book publishers such as [LeanHub](http://help.leanpub.com/en/articles/2941344-leanpub-flavoured-markdown-vs-markua-for-writing-in-plain-text" \t "_blank) accepts Markdown manuscripts and convert them to books for publication.

Slide-show presentations such as [Remark](https://remarkjs.com/) and [Cleaver](https://jdan.github.io/cleaver/#2) can convert Markdown slides into HTML for web viewing.

**The Markdown Parser**

A Markdown parser is software written to parse the Markdown syntax in a text document and convert it to HTML syntax. The original Markdown parser was written in Perl, but you can find parser [implementations](https://github.com/markdown/markdown.github.com/wiki/Implementations) today in almost any programming language. Regardless, a basic Markdown parser should support the [core Markdown syntax](https://daringfireball.net/projects/markdown/basics): paragraphs, headers, blockquotes, phrase emphasis, lists, code, images, and links.

There are Markdown parsers that are freely available on the Web: [StackEdit.io](https://stackedit.io/app), [Dillinger](http://dillinger.io/), [Parse](https://parsedown.org/demo) and [Markdown to HTML Converter](https://markdowntohtml.com/). In addition to parsing and rendering, both Parse and Markdown to HTML Converter also convert the Markdown document to raw HTML so that you can copy and paste the HTML to be used elsewhere.

**Markdown Tutorial**

Let’s learn a little more about the Markdown syntax. As mentioned above, you can simulate a <h1> HTML tag with a # Markdown symbol. Since there are six heading tags for HTML, from <h1> through <h6>, you can simulate this with # through ###### in Markdown. For example this Markdown syntax:

# This is a H1 heading  
## This is a H2 heading  
### This is a H3 heading  
#### This is a H4 heading  
##### This is a H5 heading  
###### This is a H6 heading

will render this:

This is a H1 heading

This is a H2 heading

This is a H3 heading

This is a H4 heading

This is a H5 heading

This is a H6 heading

In addition to numbered lists, you can style a bulleted list too. There are three different symbols you can use to mark a line item as a bullet: asterisk (\*), plus sign (+), or hyphen (-).

For example:

\* Markdown  
+ HTML  
- XML

produces this:

Markdown

HTML

XML

For best practice, it is recommended to use the same marker throughout your list instead of mixing them like above.

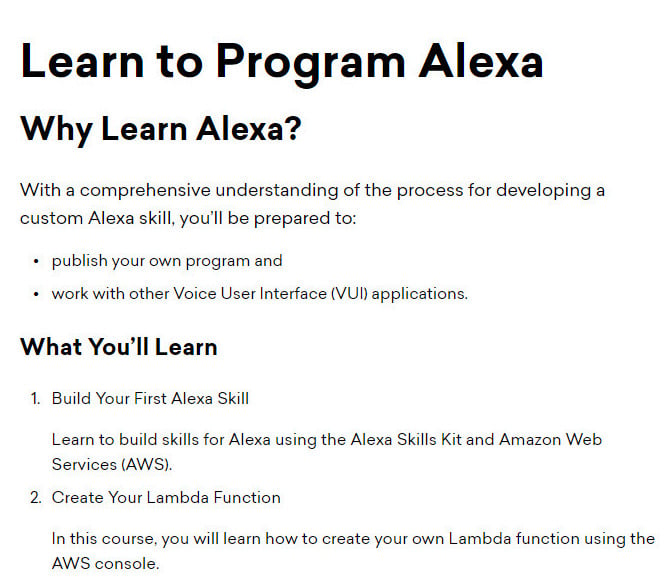
The core Markdown syntax does not include nested lists (list within another list), but it allows adding paragraphs between list items. To do so, you need to add a blank line after a list item and indent 4 spaces or 1 tab before starting a paragraph. Some parsers are lenient and do not enforce 4 spaces but there should be some spacing.

\* Markdown  
   
   Markdown is a lightweight markup language for styling text.  
   
\* HTML  
   
   HTML stands for HyperText Markup Language.

For more examples of basic syntax, check out [CommonMark’s reference guide](https://commonmark.org/help/" \t "_blank).

Fill in the blank

Let’s practice styling a page that contains a truncated description of one of Codecademy’s courses, [Learn to Program Alexa](https://www.codecademy.com/learn/learn-alexa). Fill in the code so that your output should look like this:



# Learn to Program Alexa  
   
## Why Learn Alexa?  
   
With a comprehensive understanding of the process for developing a custom Alexa skill, you’ll be prepared to:  
+ publish your own program and   
+ work with other Voice User Interface (VUI) applications.  
   
### What You'll Learn  
   
1. Build Your First Alexa Skill  
   
    Learn to build skills for Alexa using the Alexa Skills Kit and Amazon Web Services (AWS).  
    
2. Create Your Lambda Function  
   
    In this course, you will learn how to create your own Lambda function using the AWS console.

👏

You got it!

**The Markdown Flavors**

Because the core Markdown language supports only a subset of HTML features, many independent developers have extended the Markdown syntax to incorporate more HTML compatibilities and customize it for their own organizations. Here are a few popular flavors of Markdown:

[CommonMark](https://commonmark.org/) is a body of special-interest developers who work side-by-side on a proposal to standardize the Markdown syntax and offer extensive test suites to validate Markdown implementations against this specification. This standard has been used by other developers to base their code upon.

GitHub Flavored Markup, or [GFM](https://github.github.com/gfm/) is GitHub’s expanded dialect of Markdown based on CommonMark and is used throughout the GitHub platform by its active community.

[Trello](https://help.trello.com/article/821-using-markdown-in-trello), a popular collaborative tool that organizes and tracks information through virtual boards and cards, implements a custom version of Markdown as well.

Conclusion

You’ve learned a lot about Markdown, specifically:

What Markdown is and how it differs from HTML

How you can benefit from Markdown

Where and when to apply Markdown in various scenarios

How to write Markdown to style your text and make it presentable

What a Markdown parser is and where to locate one

Where to find other flavors of Markdown that are used in industry

**Cheatsheet**

<https://commonmark.org/help/>

**Github Flavored Markdown (GFM)**

<https://github.github.com/gfm/>

**Common Mark**

<https://commonmark.org/>

**How To Write a Good README for Your Project**

This article explains the purpose, conventional structure, and proper formatting of a README file, and best practices to follow when writing one.

**What Is a README File?**

You may have noticed that when a GitHub repo is initialized, you see a prompt to create a README.md file immediately. As implied in its name, a [README](https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-readmes) file is a text file that is meant to be read as soon as someone views the repository. This file contains text to introduce, explain, and share the information required to understand what the project is about.

Since a README file is often the first thing a visitor sees, README’S should tell the viewer how to install and use the project. There is no one good way to structure a README but there is only one bad way: not to include a README at all.

**Complete the sentence below about the function of a README file.**

A README file is the first thing a user sees on your GitHub repository. It provides a/an overview of what your project is about.

👏 You got it!

**Conventions of a Good README File**

Your README file should be as good as your project itself.

Make your project stand out look professional by at least including the following elements in your README:

**Project Title**: the name of your project

**Description**: This is an extremely important component of the README. You should describe the main purpose of your project. Answer questions like “why did you build this project?” and “what problem(s) does it solve?”. It also helps to include your motivations for the project and what you learned from it.

**Features**: If your project has multiple features, list them here. Don’t be afraid to brag if your project has unique features that make it stand out. You can even add screenshots and gifs to show off the features.

**How to use**: Here, you should write step-by-step instructions on how to install and use your project. Any software or package requirements should also be listed here.

**Technologies**: List all the technologies and/or frameworks you used and what purpose they serve in your project.

**Collaborators**: If others have contributed to your project in any way, it is important to give them credit for their work. Write your team members’ or collaborators’ names here along with a link to their GitHub profile.

**License**: It’s also important to list a license on your README so other developers can understand what they can and cannot do with your project. You can use [this guide](https://choosealicense.com/) to help you choose a license.

Keep **README’s** brief but detailed. **README** should always be up-to-date and self-explanatory. If you have spent a lot of time on your project, you should also spend a good chunk of time on the README. It can help your future self as well when you step away for a while and need to get reacquainted with your project. Not to mention it’ll leave a positive impression on future interviewers who look at your GitHub profile.

**Using Markdown to Format README’s**

Keep in mind that nobody wants to read a long block of unformatted text bloated with information. That is why a README file usually has the .md extension. Formatting README files with Markdown can give it flair and make it interesting to read.

Multiple choice

What file extension is typically used for a README file on GitHub?

.txt

.html

(Selected)Correct:

.md

👏

You got it! Markdown files are automatically rendered on GitHub so they are the preferred file type for a README.

**Use Headers**

Every title or section (usage, license, etc.) in a README.md should be formatted as a header. Using headers and adding some HTML, we can achieve stuff like this:

**📕 Codecademy Docs**

Documentation for popular programming languages and frameworks.  
Built by the community. Maintained by Codecademy.

**What is Docs?**

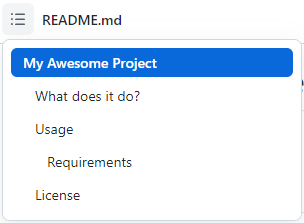
[Codecademy Docs](https://www.codecademy.com/resources/docs) is a free, easily accessible reference for coding terms and concepts, available to developers all over the world.

Take a look at the README file on [Codecademy’s docs repository](https://github.com/Codecademy/docs" \t "_blank) for the source code.

GitHub uses headers to automatically generate a table of contents for README files based on the Markdown sections. Something like this

## Usage  
Please follow these steps to get your application up and running.  
   
### Requirements  
The following packages are required for this script to work.  
- pandas  
- numpy  
- requests  
   
To install these packages, you can simply run: `pip install -r requirements.txt`

GitHub will find the headers and include them in a table of contents based on the header hierarchy:



**Level Up README’s With Media**

Documentation doesn’t have to be boring. Go for images or videos to make a project more understandable and appealing! We can add project’s logo, diagrams, screenshots, or even GIFs!

For Markdown files living in a repository, the path to the image can either be to an URL or to an image file in the repository. For example, if we had an image named diagram.png inside the images folder of the repository, we could reference it like so ![Use Case Diagram](./images/diagram.png) in the README.md.

**Conclusion**

Write engaging README’s! Make sure they include easy-to-follow details for your project. This improves your documentation skills and makes you a better developer and communicator of code.

If you want to see more examples of good README’s, take a look at the README on Codecademy’s [40Phaser](https://github.com/Codecademy/40Phaser) and [stuff.js](https://github.com/Codecademy/stuff.js) repositories. For more information on GitHub’s formatting syntax, take a look at their [docs](https://docs.github.com/en/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax).

Multiple choice

**Which of the following should you NOT include in your project’s README file?**

A clever name for your project.

The license under which your code is made available.

A list of your project’s features.

(Selected)Correct:

Documentation describing each file in your project.

👏Since the README file is usually the first thing people see in your GitHub repository, it should give a clear and concise introduction to your project as a whole rather than a detailed summary of every file.