MARKDOWN

A COMPLETE GUIDE

NEHAL KHAN

Getting Started

What is Markdown?

Markdown is a simple and lightweight markup language to format your plaintext documents. It was created by John Gruber in 2004. It is one of the world's most popular markup languages after HTML. One of the purposes for creating Markdown was to create web pages without all the HTML tags hindering the writing process. Therefore, Markdown is a user-readable markup language even in its source code form. You can create a Markdown source file using a .md or .markdown file extension. You can use it for writing blogs, notes, documents, online forums, documentation, static websites, and readme files.

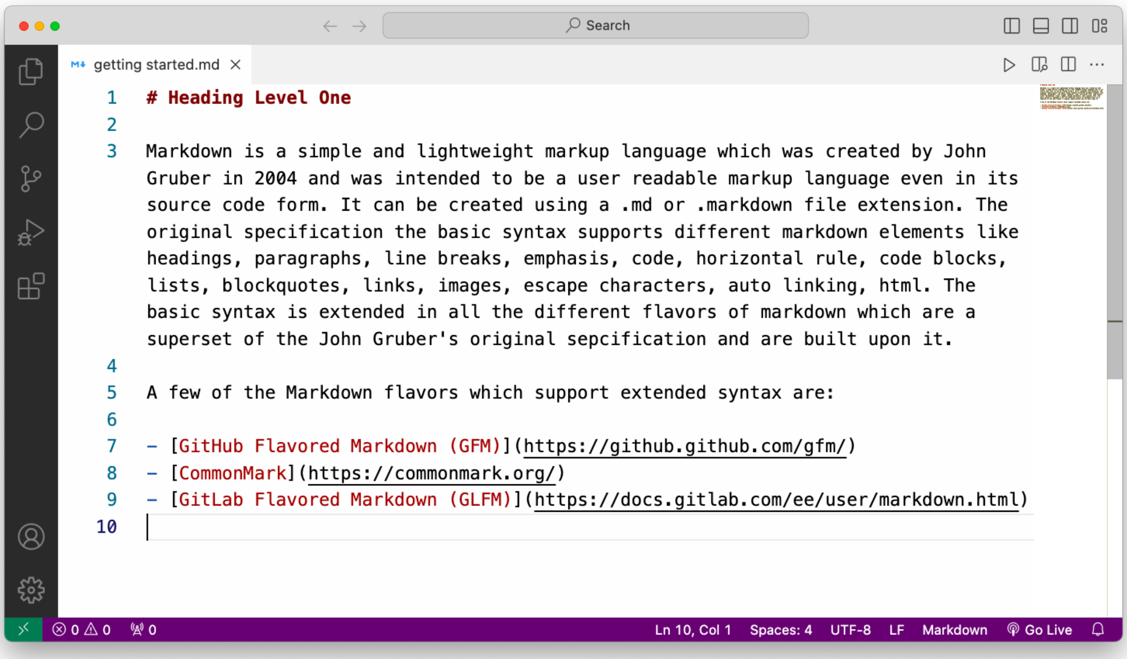
The basic syntax did not cover all the markup elements, so in the following few years, many new flavors of Markdown language came up. These new variations of the Markdown language added extended syntax for adding the markup elements missing in the original design specification. Still, the amount of syntax for the Markdown language is smaller in volume than the syntax for HTML, which works very well for its intended use of working with plaintext documents.

Using Markdown language is much different than using any WYSIWYG (What You See Is What You Get) editors like MS Word or Google Docs. In those kinds of applications, you would use different buttons and options to format your text, and the changes you make to the document are visible instantly. Markdown works very differently because you add syntax to format your text. The syntax included with the plaintext content of your source will render as different Markdown elements in your output.

You can add Markdown syntax to your text using any simple text editor. You can also choose from the several web-based and standalone applications available for free. Based on the choice of your application, you may be able to preview the rendered output in real-time. But please note that the preview option may not be available in all the applications. It may be a problem for some users, but for other users, it may not be a concern since the syntax for Markdown is designed to be user-readable even in its source code format. For instance, you can see a few examples of the syntax below.

To create a heading in your text, add a hash/pound sign before the text (e.g., # Heading Level One). And, to make your text/phrase an italic, add an * (asterisk) before and after the text (e.g., This is an *italic text*).

For example, you can also see a Markdown source file opened in the Visual Studio Code editor below.



```markdown
# Heading Level One

Markdown is a simple and lightweight markup language which was created by John
Gruber in 2004 and was intended to be a user readable markup language even in its
source code form. It can be created using a .md or .markdown file extension. The
original specification the basic syntax supports different markdown elements like
headings, paragraphs, line breaks, emphasis, code, horizontal rule, code blocks,
lists, blockquotes, links, images, escape characters, auto linking, html. The
basic syntax is extended in all the different flavors of markdown which are a
superset of the John Gruber's original sepcification and are built upon it.

A few of the Markdown flavors which support extended syntax are:

- [GitHub Flavored Markdown (GFM)](https://github.github.com/gfm/)
- [CommonMark](https://commonmark.org/)
- [GitLab Flavored Markdown (GLFM)](https://docs.gitlab.com/ee/user/markdown.html)

```

A Markdown file opened in the VS Code editor

● ● ● ● ●

Why use Markdown?

- One of the main reasons to use Markdown is that it is easy to learn and use without a huge learning curve.
- Markdown files are very portable because of the small file size since they only contain plain text. So they are easy to distribute.
- You can also open Markdown files using any text editor without worrying about different formatting settings.
- Markdown is also platform-independent. So, you can create and use Markdown files on devices running on any operating system.
- Markdown also makes writing for different mediums easier. You can write the content once and export it to multiple formats like HTML or PDF as per your preference.
- Since Markdown language is not proprietary and uses only plain text characters for formatting, it is very much future-proof.
- In a way, having limited syntax elements works in favor of making Markdown very minimal and non-invasive. So, the focus remains on writing without worrying about different font styles, sizes, templates, and layouts.
- The lack of formatting settings makes collaboration among multiple users easy without messing up the final result.

• • • • •

How does Markdown work?

You write your text content, which is stored in a plaintext format using a .md or .markdown file extension. This file goes through a Markdown processor/parser. The processor parses the file and converts it into a valid, well-formatted XHTML or HTML. Then, this formatted HTML file is rendered and displayed as the final output. In some cases, the output is also available as a print-ready PDF.

Generally, the web-based or standalone application handles the whole process of file management, Markdown processing, conversion to HTML, and rendering of the output in the background. Almost every Markdown user uses a form of Markdown application or a code editor to handle the whole process.

But, to understand what happens in the background, let's see an example of the conventional way of writing using Markdown. You write your text content in a file using any text editor with a .md or .markdown extension. Then, you use a small tool or script that processes the content and generates a valid HTML file. You then use the generated HTML file to view the final rendered output.

You can see the visual representation of this whole process below.

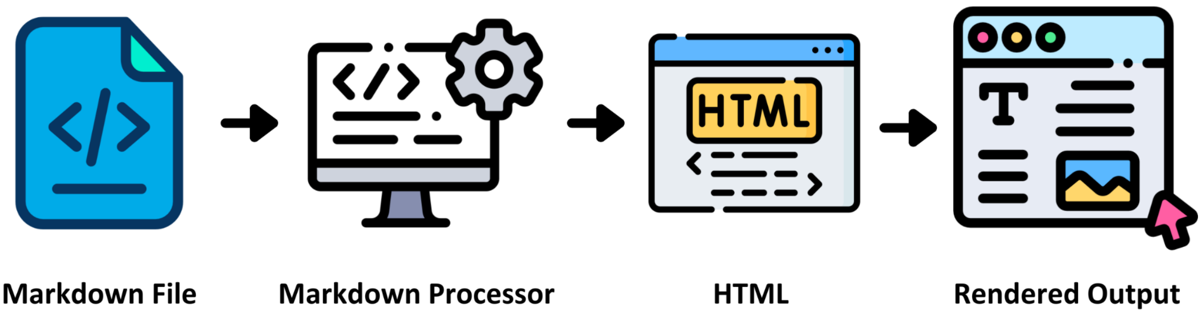


Image Credits: File by Kiranshastry via Flaticon, Software developer by Freepik via Flaticon, HTML by Freepik via Flaticon, and Web Page by Freepik via Flaticon

Note: *Since Markdown syntax gets converted to HTML, you can use different HTML tags within your Markdown files for most of the available Markdown flavors.*

• • • • •

Different Flavors of Markdown

There are many different implementations of Markdown markup language. These implementations are a superset of the original Markdown design specification. They include all the basic syntax and build upon it by adding various markup elements missing in the original Markdown specification using an extended syntax. Each implementation of Markdown supports a group of markup elements that differ slightly among themselves.

Here is a list of a few well-known implementations of Markdown. You can go to the individual specifications on their sites and check about them in detail.

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

• • • • •

What is Markdown used for?

Markdown syntax is also not that difficult to learn, and once you have learned it, you can use it for multiple purposes to write just about anything. Markdown is used for writing different things, from simple notes to creating static websites.

Below are some examples of what you can use Markdown language for:

Websites: Since Markdown was designed for the web, it is not surprising that many applications are designed to create websites using Markdown. You can use Markdown to generate static websites for hosting your blogs, technical articles, and documentation. With Markdown, writing content for a website becomes very easy. Your web content is displayed with simple formatting and a clean look, which works perfectly for the above use cases.

If you are somewhat familiar with HTML and CSS, you can look at Jekyll and GitHub Pages for generating a static website.

Some other options that can help you with static site generation are:

- Blot
- Hugo
- MkDocs

Notes: Markdown is an ideal syntax for note-taking due to its simplicity and ease of use. Evernote, one of the most popular note-taking applications, supports some Markdown syntax elements. But OneNote, another popular note-taking application, doesn't yet support it. But many other note-taking applications support Markdown.

Here is a list of a few of them.

- Obsidian
- Notable
- Boostnote
- Simplenote
- Bear
- Joplin
- Inkdrop

Documents: Markdown works very well for writing documents. It may not be as sophisticated as word processors like MS Word or Google Docs. But it is powerful enough to write proper and well-formatted documents. Even the most popular software management and version control services like GitHub and GitLab have moved towards writing the project readme files in Markdown. You can use the various Markdown apps and tools to create and export the documents to HTML or PDF formats. These file formats can later be used for distribution, uploading to a site, or printing.

Here is a list of a few Markdown editors that I recommend.

- **Windows:** MarkText, or Ghostwriter.
- **Mac:** MarkText, MacDown, or iA Writer.
- **iOS/Android:** iA Writer.
- **Linux:** Ghostwriter.
- **Web:** StackEdit, or Dillinger.

Email: If you write a lot of emails and find it frustrating to use the formatting controls available with your email providers, then you can use Markdown to write your emails.

You can write the Markdown content in your editors and paste the rendered output in your email, or you can use an extension like Markdown Here. Using this extension, you can write the Markdown syntax within your email and render it to nicely formatted content. To do this, you need to right-click on the page, which will show a Markdown Toggle option, then click on it to render the Markdown syntax to a nicely rendered output within your email.

Using the Markdown markup to format emails will keep a clean and consistent formatting across your email content.

Collaboration: Different collaboration, team messaging, and forum-based applications like Slack, Discord, Reddit, StackExchange, and Mattermost support the Markdown language. These applications are popular ways to communicate and collaborate with coworkers, friends, and community. Though these applications don't use all the features of the Markdown language, the ones they use are useful for formatting your text.

Documentation: Markdown is a natural fit for writing documentation. You can use it for both technical and non-technical documentation. Several software management tools like GitHub and GitLab have also shifted towards using Markdown for writing documentation.

A few of the tools you can use for writing documentation are:

- MkDocs
- Docusaurus
- Jekyll
- Read the Docs

Books: You can also use the Markdown to write books. You can use your Markdown editor to write a book and later export it as a PDF. Or, you can use services like Leanpub, which takes Markdown files and publishes them as eBooks in different formats like EPUB, MOBI, and PDF.

You will also have an option to download your converted eBooks in these three formats. You can later use these downloaded files to self-publish your books in eBook or Paperback versions on other platforms.

• • • • •

How to get started?

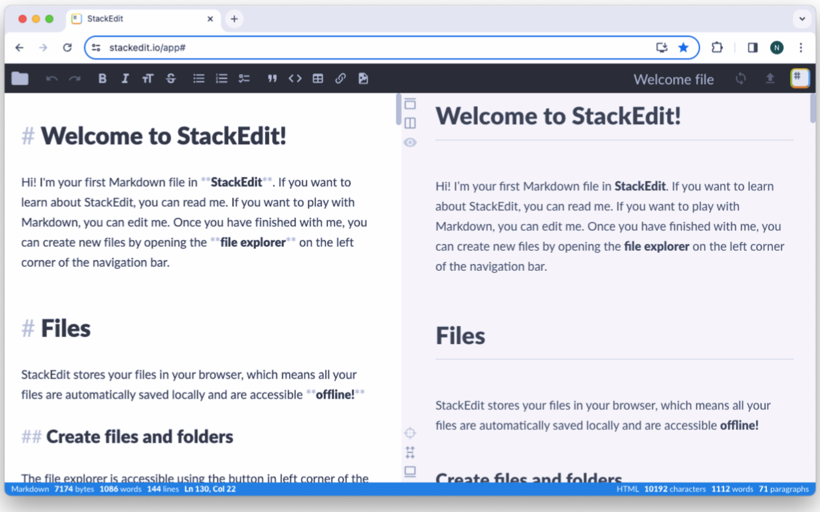
The best way to get started with learning Markdown is to use it. You can do that right away since there are plenty of web-based Markdown editors you can open up and start writing. There are different types of editors that you can use for writing Markdown, as per your preference.

Let's take a look at a few of the available options.

Online Editors:

There are many options available for online editors. Two of the best web-based editors you can use are [StackEdit.io](https://stackedit.io) and [Dillinger.io](https://dillinger.io). You can choose either of these since they are minimal and easy to use. Both will have a left pane where you can write your Markdown syntax. On the right pane, they both will show you the preview for the rendered output in real time.

StackEdit will look something like this.



```
# Welcome to StackEdit!

Hi! I'm your first Markdown file in **StackEdit**. If you want to
learn about StackEdit, you can read me. If you want to play with
Markdown, you can edit me. Once you have finished with me, you
can create new files by opening the **file explorer** on the left
corner of the navigation bar.

# Files

StackEdit stores your files in your browser, which means all your
files are automatically saved locally and are accessible **offline!**

## Create files and folders

The file explorer is accessible using the button in left corner of the
```

Welcome to StackEdit!

Hi! I'm your first Markdown file in **StackEdit**. If you want to learn about StackEdit, you can read me. If you want to play with Markdown, you can edit me. Once you have finished with me, you can create new files by opening the **file explorer** on the left corner of the navigation bar.

Files

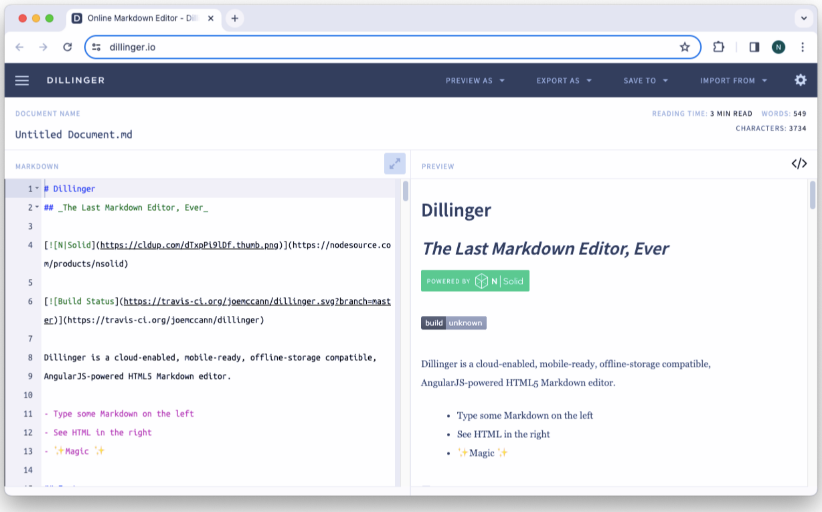
StackEdit stores your files in your browser, which means all your files are automatically saved locally and are accessible **offline!**

Create files and folders

Markdown 7174 bytes 1086 words 144 lines Ln 130, Col 22 | HTML 10192 characters 1112 words 71 paragraphs

StackEdit Home Page

Dillinger will look like this.



Untitled Document.md

```
# Dillinger
## _The Last Markdown Editor, Ever_

[![N|Solid](https://cldup.com/dTxpPi9lDf.thumb.png)](https://nodesource.com/products/nsolid)

[![Build Status](https://travis-ci.org/joemccann/dillinger.svg?branch=master)](https://travis-ci.org/joemccann/dillinger)

Dillinger is a cloud-enabled, mobile-ready, offline-storage compatible,
AngularJS-powered HTML5 Markdown editor.

- Type some Markdown on the left
- See HTML in the right
- ✨Magic ✨

```

Dillinger

The Last Markdown Editor, Ever

Dillinger is a cloud-enabled, mobile-ready, offline-storage compatible, AngularJS-powered HTML5 Markdown editor.

- Type some Markdown on the left
- See HTML in the right
- ✨Magic ✨

Dillinger Home Page

A few other options for online editors are Online Markdown Editor, Markdown Live Preview, Dingus, and Markdown Editor.

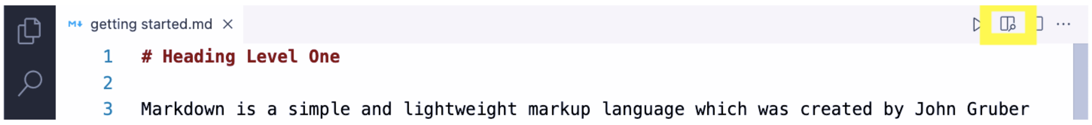
Standalone Editors:

Many standalone Markdown applications are also available. A few options are MarkText, GhostWriter, and Typora (Paid).

Code Editors:

If you are using any code editors like Visual Studio Code, Atom, Sublime, or Notepad++, you can also use them to write your Markdown markup. You can see how to set up and use your preferred code editor to write Markdown markup below.

Visual Studio Code: VS Code out of the box has support to preview Markdown files. When working with a Markdown file, to open a preview, you can click on the option with two panes and a magnifying glass in the top right corner of the editor. This option is highlighted and shown below.



```
# Heading Level One

Markdown is a simple and lightweight markup language which was created by John Gruber
```

However, the in-built Markdown processor does not support all the extended syntax, so we need to install the Markdown Extended or the Markdown Preview Enhanced extensions. These extensions will give us the support for the extended syntax and will also give you options to export the rendered output to different file formats like HTML, PDF, etc.,

Atom: You can also use Atom editor for writing Markdown, but you will need to install the Markdown-preview or the Markdown-preview-enhanced packages. Once you have installed one of the packages, you can preview your Markdown files using the Ctrl + Shift + M (Windows) or ⌘ + Shift + M (Mac) shortcuts.

Sublime Text: Sublime Text can also be used to write Markdown. If you have already set up the package control in your editor, you can install the MarkdownPreview package to preview your Markdown files.

Open the command palette and type “markdown preview”, which will show you different options. Select the “preview in browser” option and then select one of the given Markdown flavors to open the preview in your default browser.

Notepad++: To preview Markdown files in the Notepad++ editor, add the MarkdownViewer++ plugin to your editor. This plugin will allow you to see the preview of your Markdown file in the preview panel.

• • • • •

Basic Syntax

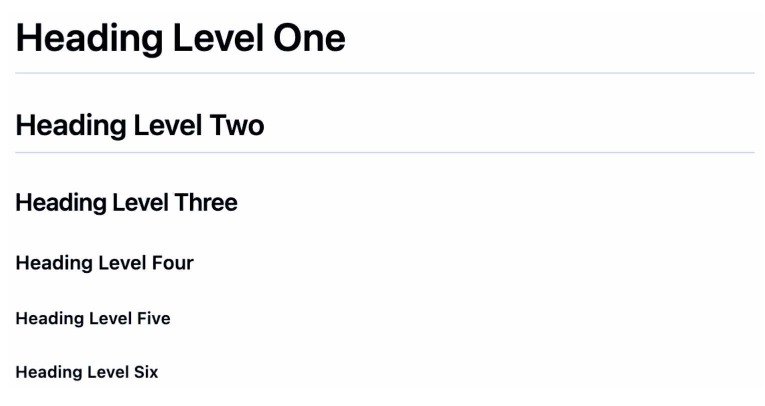
Headings

Markdown has six levels of headings. The size of a heading is determined by the number of # (hash/pound) sign(s) used before the text.

A single # sign gives you a level one heading, which is the biggest one. On the other hand, six # signs give you a level six heading, the smallest one.

```
# Heading Level One

## Heading Level Two

### Heading Level Three

#### Heading Level Four

##### Heading Level Five

###### Heading Level Six
```

The output looks like this:



Heading Level One

Heading Level Two

Heading Level Three

Heading Level Four

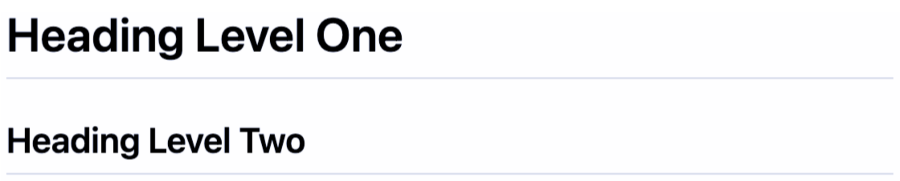
Heading Level Five

Heading Level Six

For the heading levels one and two, there is also an alternative syntax that you can use. You can write the text of your heading, and on the following line, add a couple of = (equal) signs to add a level one heading. Similarly, to add a level two heading, write the text of the heading on one line and add a couple of - (minus) signs on the following line.

```
Heading Level One
=================

Heading Level Two
-----------------
```

The output looks like this:



Heading Level One

Heading Level Two

Good Practices

Many Markdown applications may not render a proper output if there is no space between the # sign(s) and the text of your heading. As a best practice, please ensure that you have always added a space between the # sign(s) and the text of your heading, for compatibility across different Markdown applications.

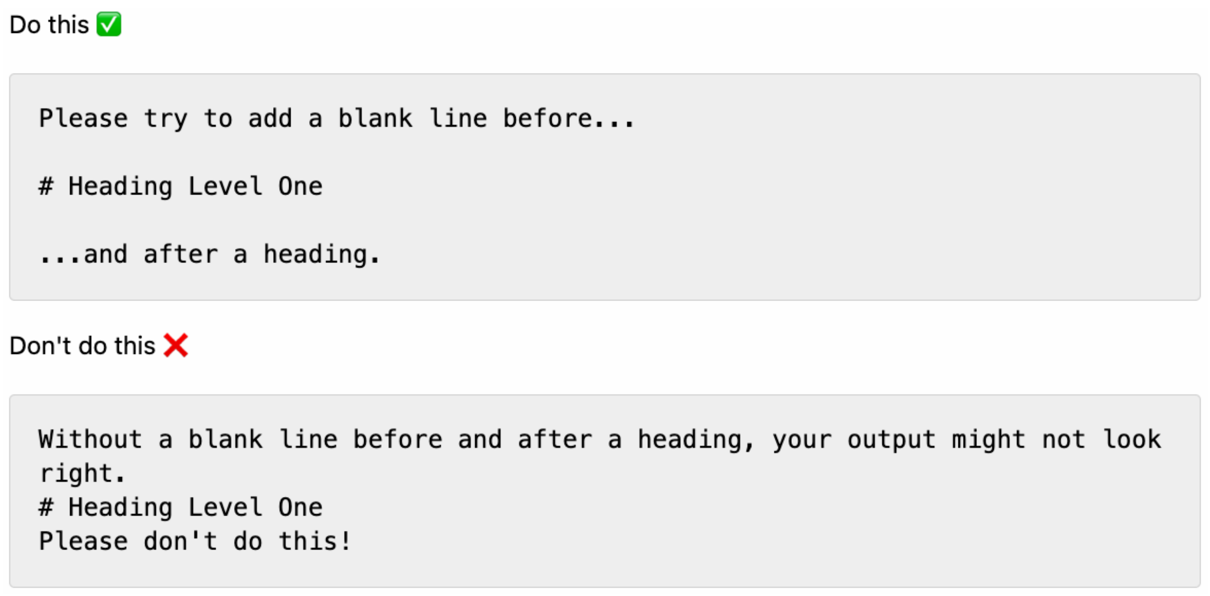
Do this ✅

```
# Heading Level One
```

Don't do this ❌

```
#Heading Level One
```

As a best practice, you should add a blank line before and after your heading(s) for better compatibility.


Do this ✅

```
Please try to add a blank line before...

# Heading Level One

...and after a heading.
```

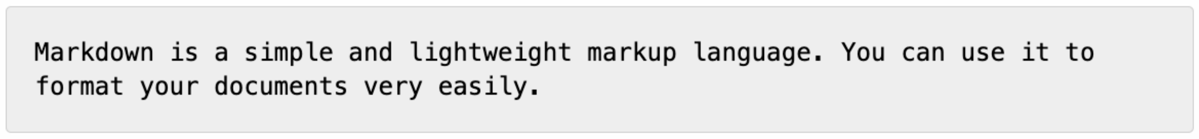
Don't do this ❌

```
Without a blank line before and after a heading, your output might not look
right.
# Heading Level One
Please don't do this!
```

• • • • •

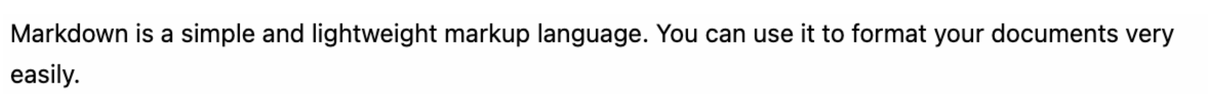
Paragraphs

You can create a paragraph by adding a blank line before and after one or more consecutive lines of text in your content.



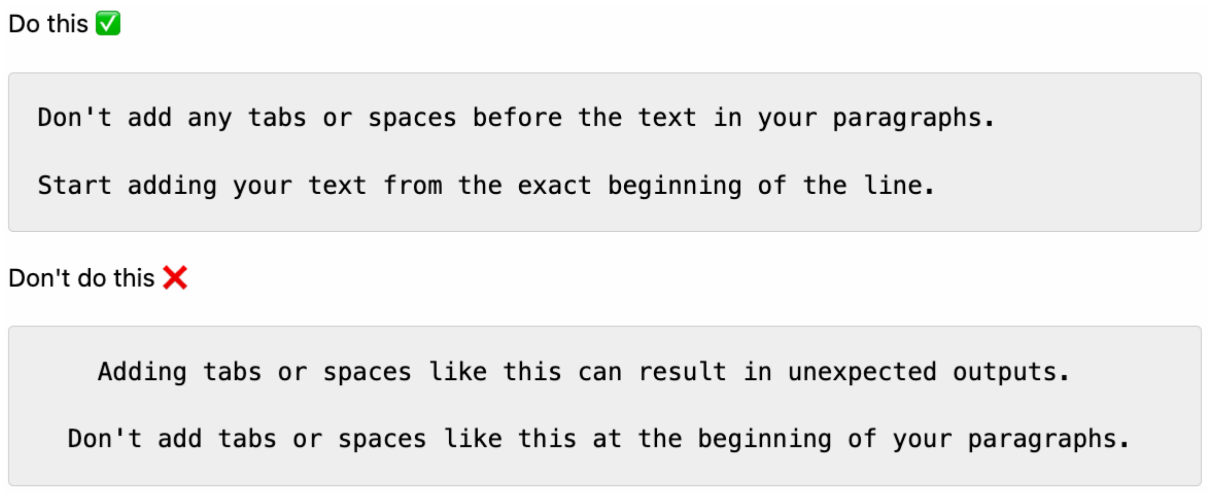
```
Markdown is a simple and lightweight markup language. You can use it to
format your documents very easily.
```

The output looks like this:


Markdown is a simple and lightweight markup language. You can use it to format your documents very easily.

Good Practices

Please don’t add any indentation using either tab(s) or space(s) before the start of your paragraph unless you are using it within a list. You will learn more about this in the upcoming Lists section.


Do this ✅

```
Don't add any tabs or spaces before the text in your paragraphs.

Start adding your text from the exact beginning of the line.
```

Don't do this ❌

```
    Adding tabs or spaces like this can result in unexpected outputs.

  Don't add tabs or spaces like this at the beginning of your paragraphs.
```

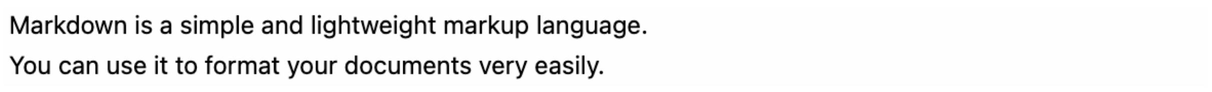
• • • • •

Line Breaks

You can add a line break in your paragraph by adding two or more white spaces (trailing whitespaces) at the end of a line. Then, you can continue to add the remaining text from the following line.

```
Markdown is a simple and lightweight markup language.  
You can use it to format your documents very easily.
```

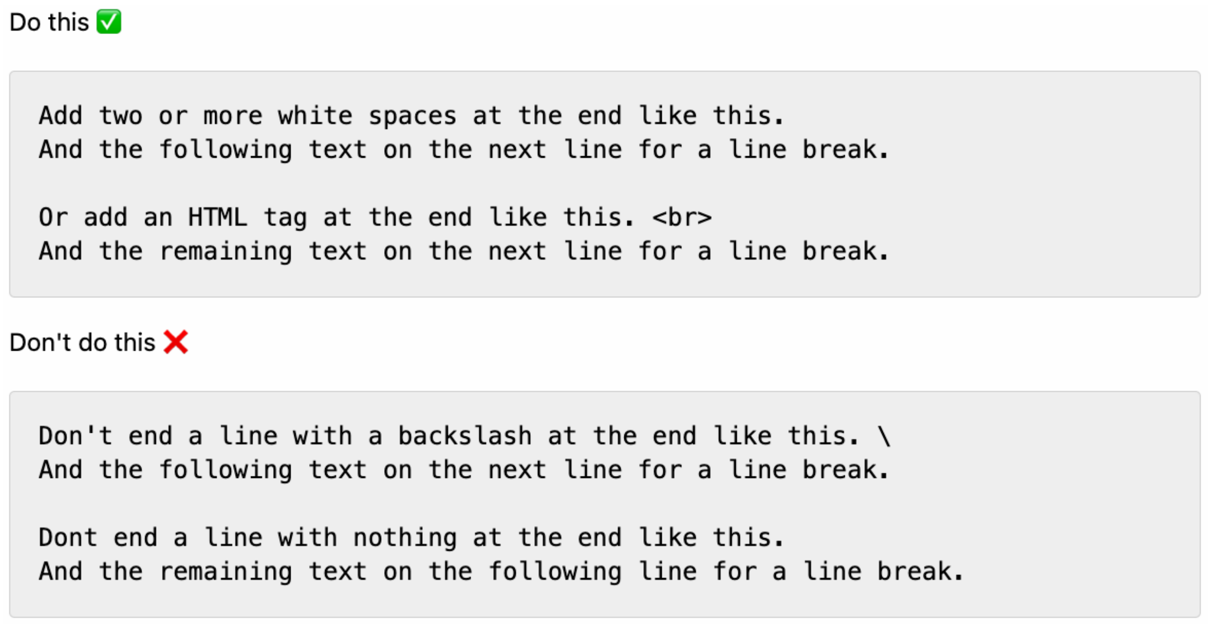
The output looks like this:


Markdown is a simple and lightweight markup language.
You can use it to format your documents very easily.

Good Practices

Some flavors in Markdown allow you to add a \ (backslash) sign at the end of the line to add a line break. Some don't even need you to add anything at the end of the line for a line break. But these two ways are not widely supported, so it's a good practice to avoid using them for better compatibility.

The trailing white spaces work in almost every Markdown flavor. But please be careful while adding them since white spaces are not readily visible, and you may accidentally add them in the wrong places. If your Markdown flavor supports HTML tags, add a <br> tag at the end of the line to create a line break. So, the use of either of these two options will be good for compatibility.



Do this ✅

```
Add two or more white spaces at the end like this.
And the following text on the next line for a line break.

Or add an HTML tag at the end like this. <br>
And the remaining text on the next line for a line break.
```

Don't do this ❌

```
Don't end a line with a backslash at the end like this. \
And the following text on the next line for a line break.

Dont end a line with nothing at the end like this.
And the remaining text on the following line for a line break.
```

• • • • •

Emphasis

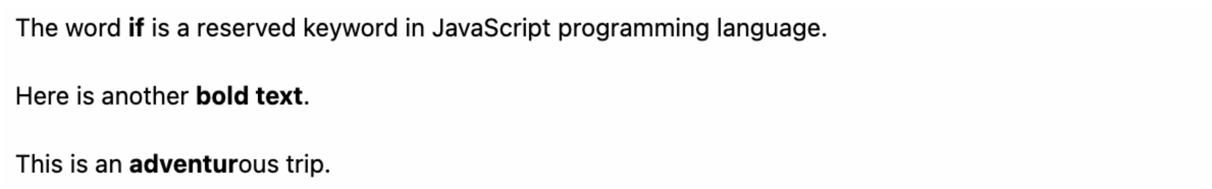
You can emphasize the text in your content as bold, italic, or both bold and italic.

Bold

You can add bold text to your content by adding two * (asterisk) or two _ (underscore) signs before and after the word or a phrase. If you want to bold just a few letters of a word, you can add two asterisks before and after just those letters without any space(s).

```
The word **if** is a reserved keyword in JavaScript programming language.

Here is another __bold text__.

This is an **adventur**ous trip.
```

The output looks like this:

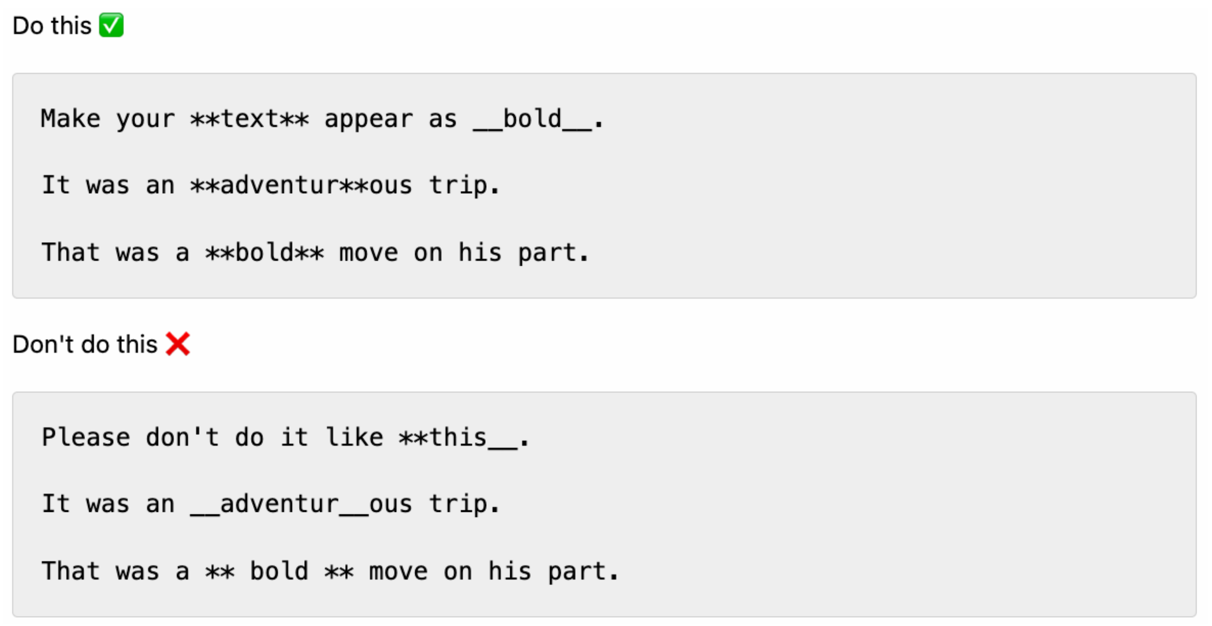

The word **if** is a reserved keyword in JavaScript programming language.

Here is another **bold text**.

This is an **adventur**ous trip.

Good Practices

You can use either of the two ways to make the text bold. But don’t mix and match the opening and closing characters. You can use two asterisks or two underscores to emphasize a complete word. But, if you want to do it to a few characters of a word, then two underscores may not show a proper output. So, it’s better to use two asterisks for better compatibility. Don’t add spaces between two asterisks/two underscores and your text. The output may not render as it should and give you just literal characters.


Do this ✅

```
Make your **text** appear as __bold__.

It was an **adventur**ous trip.

That was a **bold** move on his part.
```

Don't do this ❌

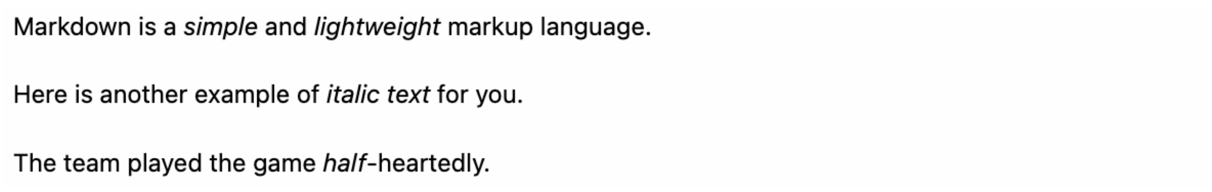
```
Please don't do it like **this__.

It was an __adventur__ous trip.

That was a ** bold ** move on his part.
```

Italic

You can make your text italic by adding a single * (asterisk) or an _ (underscore) sign before and after the word or phrase. If you want to italicize just a few letters of a word, you can add one asterisk without any space(s) before and after just those letters of the word.

```
Markdown is a *simple* and *lightweight* markup language.

Here is another example of _italic text_ for you.

The team played the game *half*-heartedly.
```

The output looks like this:



Markdown is a *simple* and *lightweight* markup language.

Here is another example of *italic text* for you.

The team played the game *half*-heartedly.

Good Practices

You can use either the asterisk or underscore to make the text italic. But don’t mix and match the opening and closing characters.

You can use an asterisk or an underscore to emphasize a complete word. But, if you want to do it to a few characters of a word, an underscore may not show a proper output. So, it’s better to use an asterisk for better compatibility.

Don’t add spaces between an asterisk/underscore and your text. The output may not render it as emphasized and give you just literal character. In some cases, it may be added as a list item if the word you want to italicize is at the beginning of the phrase.

Do this ✅

```
Make your *text* appear as _italic_.

The team played the game *half*-heartedly.

They laughed *wholeheartedtly* after seeing their old yearbooks.
```

Don't do this ❌

```
Please don't do it like *this_.

The team played the game _half_-heartedly.

They laughed * wholeheartedtly * after seeing their old yearbooks.
```

Bold and Italic

You can make your text look bold and italic by adding three * (asterisk) or three _ (underscore) signs before and after the word or phrase. If you want just a few letters of a word to appear bold and italic, you can add three asterisks without any space(s) before and after just those letters in the word.

```
Your passwords should be ***alphanumeric*** with at least a few special
characters.

This is another example of both ___bold and italic text___.

You can use it like **_this_** as __*well*__.

The Team Coach was very ***un***happy with the results.
```

The output looks like this:

Your passwords should be ***alphanumeric*** with at least a few special characters.

This is another example of both ***bold and italic text***.

You can use it like ***this*** as ***well***.

The Team Coach was very ***un***happy with the results.

Good Practices

You can use three asterisks or three underscores to make the text bold and italic. But don’t mix and match the corresponding pairs of opening and closing characters. You can use three asterisks or three underscores to emphasize a complete word. But, if you want to do it to a few characters of a word, then three underscores may not generate a proper output. So, it’s better to use three asterisks for better compatibility. Don’t add spaces between the asterisks/underscores and your text. The output may not render as emphasized and give you just the literal characters.

Do this ✅

```
Make your ***text*** appear as ___bold and italic___.

I was very ***un***happy with the results.

The approach was a bit ***skeptical***.
```

Don't do this ❌

```
Please don't do it like **_this*__.

I was very ___un___happy with the results.

The approach was a bit *** skeptical ***.
```

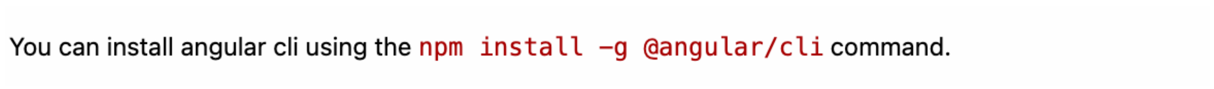
• • • • •

Code

You can add text as inline code within your content using the ` (backtick) sign before and after a word or a phrase. Any special characters except the ` (backtick) and \ (backslash) you use inside the inline code will not be rendered as per the Markdown syntax and will be displayed as literal characters.

```
You can install angular cli using the `npm install -g @angular/cli`
command.
```

The output looks like this:


You can install angular cli using the `npm install -g @angular/cli` command.

Note: *You can use \ (backslash) to escape special characters in Markdown and display them as literal characters. You will learn more about this in the upcoming Escaping Characters section.*

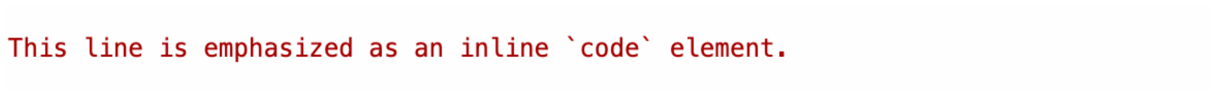
Adding a backtick or backslash in your inline code

You can escape a backtick or a backslash sign in your inline code and display them as a literal character by surrounding the phrase or paragraph with two ` (backtick) signs instead of one.

When you add a backtick between the opening and closing pair of backticks for inline code, the middle backtick may interfere with a closing one, and the remaining phrase will be missing the code styling, thus messing up the whole output. So, if we use two backticks as an opening and closing pair, the backtick(s) we may use in between will be displayed as the literal character(s).

```
`` This line is emphasized as an inline `code` element.``
```

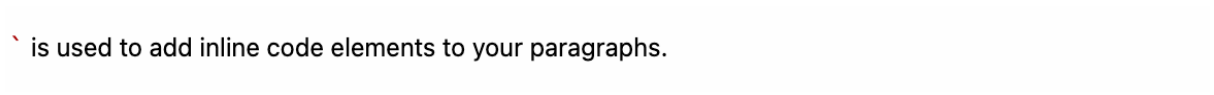
The output looks like this:


``This line is emphasized as an inline `code` element.``

You can also add a backtick as a single inline code element using two backticks as opening and closing pairs and adding a space before and after the backtick to get it as a literal character.

```
`` ` `` is used to add inline code elements to your paragraphs.
```

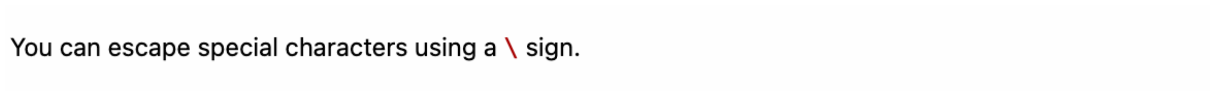
The output looks like this:


` is used to add inline code elements to your paragraphs.

When we want to display a backslash as a single inline code element in a phrase, the closing backtick may be escaped by the backslash they are enclosing. So, we add the backslash using two backticks to render it as intended.

```
You can escape special characters using a ``\`` sign.
```

The output looks like this:


You can escape special characters using a \ sign.

● ● ● ● ●