Kartik Bambhaniya

# Markdown Guide

## Introduction

Markdown is a simple way to format text documents by adding special symbols to make things like headings or bold text. It was created by John Gruber in 2004 and has become really popular.

Unlike programs like Microsoft Word where you click buttons to make text look a certain way, with Markdown, you put special codes in your text to show how you want things to look.

For example, if you want a heading, you put a hashtag before it like this: `# Heading One`. If you want to make something bold, you use two asterisks before and after the text, like this: `**this text is bold**`.

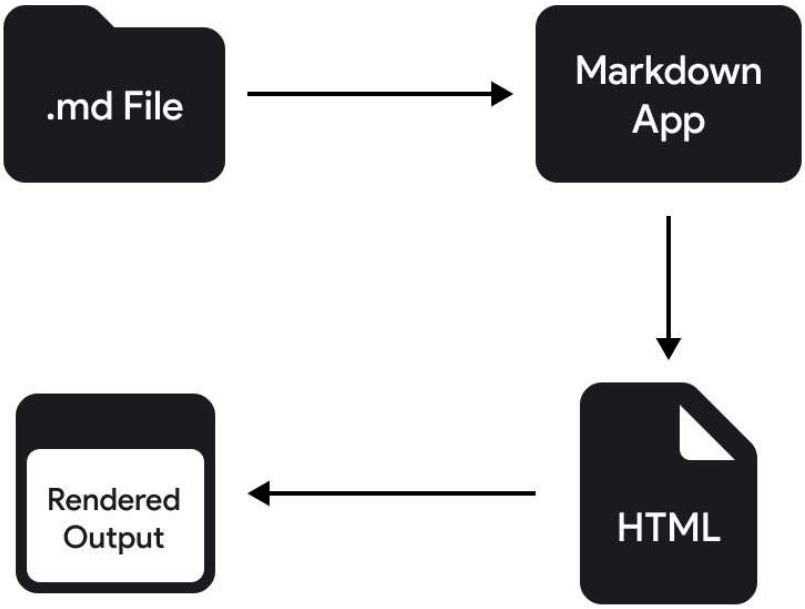
It might seem a bit strange at first if you're used to programs that show changes right away, but once you get the hang of it, Markdown can be a quick and easy way to format your text.

---

## How markdown works ??

Markdown works by using simple and intuitive syntax to format plain text documents. When you write in Markdown, the syntax you use provides instructions on how the text should be displayed or structured. Here's a basic explanation of the background process of Markdown:

1. **Text Input**: You start by typing plain text using a simple and readable syntax.
2. **Interpretation**: The Markdown processor interprets the syntax you've used as you type. It identifies patterns and structures based on the symbols and characters you include.
3. **Conversion to HTML**: Markdown is often converted into HTML (Hypertext Markup Language), which is the standard language for creating web pages. This conversion is performed by the Markdown processor.
4. **Browser Rendering**: If you're viewing the Markdown document in a web browser, the HTML code generated by the Markdown processor is rendered by the browser. This means that the browser displays the formatted content according to the instructions provided in the Markdown syntax.




*Visual overview of the Markdown process.

## Table of Contents

# Basics

## Headers

To create a heading, add one to six # symbols before your heading text. The number of # you use will determine the hierarchy level and typeface size of the heading.

```
# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6
```

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

## Text Formatting

You can indicate emphasis with bold, italic, strikethrough, subscript, or superscript text in comment fields and .md files.

| Style | Syntax | Output |
| --- | --- | --- |
| Bold | `**bold**` or `__bold__` | **bold** |
| Italic | `*italic*` or `_italic_` | *italic* |
| Strikethrough | `~~strikethrough~~` | ~~strikethrough~~ |
| Subscript | `H~2~O` | $H_2O$ |
| Superscript | `X^2^` | $X^2$ |

You can combine these formatting styles together.

## Quoting text

You can quote text with a `>`.

```
> Text that is a quote
```

> Text that is a quote

## Links

You can create an inline link by wrapping link text in brackets `[ ]`, and then wrapping the URL in parentheses `( )`.

```
Check out us on [Youtube](https://youtube.com)
```

Check out us on Youtube

```
Can also link sections of the same document by using relative links: [Link to section](#section)
```

Link to section

## Images

You can add images to your document by using the following syntax:

```
![Image](image.jpg)
```

## Lists

You can make an unordered list by preceding one or more lines of text with -, *, or +.

```
- George Washington

* John Adams

- Thomas Jefferson
```

- George Washington
- John Adams
- Thomas Jefferson

Result showing a bulleted list of the names of the first three American presidents.

To order your list, precede each line with a number.

```
1. James Madison
1. James Monroe
1. John Quincy Adams
```

1. James Madison
2. James Monroe
3. John Quincy Adams

## Task lists

To create a task list, preface list items with a hyphen and space followed by [ ]. To mark a task as complete, use [x].

```
- [x] Finish my changes
- [ ] Push my commits to GitHub
- [ ] Open a pull request
```

- [x] Finish my changes
- [ ] Push my commits to GitHub
- [ ] Open a pull request

## Using emoji

You can add emoji to your writing by typing :EMOJICODE:, a colon followed by the name of the emoji.

```
:point_right: How it's Look ?? - it's ready to merge :smiley: !!
```

👉 How it's Look ?? - it's ready to merge 😃 !!

Result showing how emoji codes for +1 and shipit render visually as emoji.

For a full list of available emoji and codes, see The Emoji Cheat Sheet.

## Paragraphs

You can create a new paragraph by leaving a blank line between lines of text.

## Footnotes

You can add footnotes to your content by using this bracket syntax:

```
Here is a simple footnote[^1].
A footnote can also have multiple lines[^2].

[^1]: My reference.
[^2]:
    To add line breaks within a footnote, prefix new lines with 2 spaces.
    This is a second line.
```

The footnote will look like this in the rendered document :

> Here is a simple footnote[1].
> A footnote can also have multiple lines[2].
>
> ---
>
> 1. My reference. ↩
> 2. To add line breaks within a footnote, prefix new lines with 2 spaces.
>    This is a second line. ↩

---

# Advanced Formatting

## Code Blocks

You can format text as code with backticks.

````
```
function test() {
  console.log("notice the blank line before this function?");
}
```
````

Result is look like this :

```
function test() {
  console.log("notice the blank line before this function?");
}
```

You can add an optional language identifier to enable syntax highlighting in your fenced code block.

Syntax highlighting changes the color and style of source code to make it easier to read.

For example, to syntax highlight Python code :

````
```python
def hello_world():
    print("Hello, World!")
```
````

Result is look like this :

```
def hello_world():
    print("Hello, World!")
```

## Footnotes

You can add footnotes to your content by using this bracket syntax:

```
Here is a simple footnote[^1].
A footnote can also have multiple lines[^2].

[^1]: My reference.
[^2]:
    To add line breaks within a footnote, prefix new lines with 2 spaces.
    This is a second line.
```

The footnote will look like this in the rendered document :

> Here is a simple footnote[1].
> A footnote can also have multiple lines[2].
>
> ---
>
> 1. My reference. ↩
> 2. To add line breaks within a footnote, prefix new lines with 2 spaces.
>    This is a second line. ↩

## Creating a Table

You can create tables with pipes | and hyphens -. Hyphens are used to create each column's header, while pipes separate each column. You must include a blank line before your table in order for it to correctly render.

```
| First Header | Second Header |
| ------------ | ------------- |
| Content Cell | Content Cell  |
| Content Cell | Content Cell  |
```

| First Header | Second Header |
| --- | --- |
| Content Cell | Content Cell |
| Content Cell | Content Cell |

Result table with two columns of equal width as rendered. Headers render in boldface, and alternate content rows have gray shading.

The pipes on either end of the table are optional.

Cells can vary in width and do not need to be perfectly aligned within columns. There must be at least three hyphens in each column of the header row.

```
| Command    | Description                                 |
| ---------- | ------------------------------------------- |
| git status | List all new or modified files              |
| git diff   | Show file differences that haven't been staged |
```

| Command | Description |
| --- | --- |
| git status | List all new or modified files |
| git diff | Show file differences that haven't been staged |

Result with two columns of differing width as rendered. Rows list the commands "git status" and "git diff" and their descriptions.

## Formatting within Table

You can use formatting such as links, inline code blocks, and text styling within your table:

```
| Command      | Description                                      |
| ------------ | ------------------------------------------------ |
|  `git status`  | List all  _new or modified_  files                  |
|  `git diff`    | Show file differences that  **haven't been**  staged |
```

| Command | Description |
| --- | --- |
| `git status` | List all *new or modified* files |
| `git diff` | Show file differences that **haven't been** staged |

Result with two columns of differing width as rendered. The commands "git status" and "git diff" are formatting as code blocks.

You can align text to the left, right, or center of a column by including colons : to the left, right, or on both sides of the hyphens within the header row.

```
| Left-aligned | Center-aligned | Right-aligned |
| :----------- | :------------: | ------------: |
| git status   |   git status   |    git status |
| git diff     |    git diff    |      git diff |
```

| Left-aligned | Center-aligned | Right-aligned |
| :--- | :---: | ---: |
| git status | git status | git status |
| git diff | git diff | git diff |

Result with three columns as rendered, showing how text within cells can be set to align left, center, or right.

## Creating a collapsed section

You can temporarily obscure sections of your Markdown by creating a collapsed section that the reader can choose to expand. For example, when you want to include technical details in an issue comment that may not be relevant or interesting to every reader, you can put those details in a collapsed section.

Any Markdown within the `<details>` block will be collapsed until the reader clicks to expand the details.

Within the `<details>` block, use the `<summary>` tag to let readers know what is inside. The label appears to the right of.

````
<details>
<summary>Tips for collapsed sections</summary>

### You can add a header

You can add text within a collapsed section.
You can add an image or a code block, too.

`ruby
   puts "Hello World"
`

</details>
````

▶ Tips for collapsed sections

## Creating Mermaid diagrams

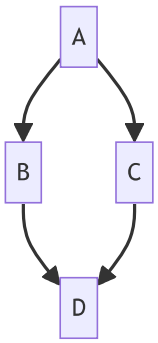
Mermaid is a Markdown-inspired tool t
hat renders text into diagrams. For example, Mermaid can render flow charts, sequence diagrams, pie charts and more. For more information, see The Mermaid Documentation.

For example, you can create a flow chart by specifying values and arrows.

Here is a simple flow chart:

````
```mermaid
graph TD;
    A⟶B;
    A⟶C;
    B⟶D;
    C⟶D;
```
````

Result is look like this :




---

# Mathematics Expressions Writing

To enable clear communication of mathematical expressions, GitHub supports LaTeX formatted math within Markdown. For more information, see LaTeX/Mathematics in Wikibooks.

## Writing Inline Expressions

There are two options for delimiting a math expression inline with your text. You can either surround the expression with dollar symbols ( `$` ), or start the expression with `$` and end it with `$` .

```
This sentence uses `$` delimiters to show math inline: $\sqrt{3x-1}+(1+x)^2$
```

This sentence uses `$` delimiters to show math inline: $\sqrt{3x-1}+(1+x)^2$

Result showing how a mathematical expression displays. The equation is the square root of 3 x minus 1 plus open paren 1 plus x close paren squared.

## Writing expressions as blocks

To add a math expression as a block, start a new line and delimit the expression with two dollar symbols `$$` .

```
**The Cauchy-Schwarz Inequality**
$$\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)$$
```

**The Cauchy-Schwarz Inequality**

$$\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)$$

Result showing how a complex equation displays. The bolded text reads "The Cauchy-Schwarz Inequality". Below the text, there is an equation showing open paren the sum from k equals 1 to n of a sub k b sub k close paren squared is less than or equal to open paren the sum from k equals 1 to n of a sub k squared close paren times open paren the sum from k equals 1 to n of b sub k squared close paren.

Alternatively, you can use the ```` ```math ```` code block syntax to display a math expression as a block. With this syntax, you don't need to use $$ delimiters. The following will render the same as above:

````
**The Cauchy-Schwarz Inequality**

```math
\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)
```
````

**The Cauchy-Schwarz Inequality**

$$\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)$$

## Writing dollar signs in line with and within mathematical expressions

To display a dollar sign as a character in the same line as a mathematical expression, you need to escape the non-delimiter $ to ensure the line renders correctly.

Within a math expression, add a \ symbol before the explicit $.

```
This expression uses `\$` to display a dollar sign: $\sqrt{\$4}$
```

This expression uses `\$` to display a dollar sign: $\sqrt{\$4}$

Result showing how a backslash before a dollar sign displays the sign as part of a mathematical expression.