



Markdown

Version CommonMark 0.31.2 · verified

2026-08-07

A gradual walk through Markdown —
blocks, inlines, links, and GFM
extensions — with the parsing rules that
decide what your text becomes.

Contents

Mental model	2
Paragraphs and line breaks	2
Headings and thematic breaks	4
Emphasis and inline code	5
Lists	6
Code blocks	7
Links and images	9
Blockquotes and nesting	10
GFM extensions	11
Escaping, HTML, and portability	12
Further reading	14

Mental model

Markdown is plain text that a parser converts to HTML, and it parses in two passes: first **block structure** — headings, lists, quotes, code — decided purely by how lines begin, indent, and separate; then **inline structure** — emphasis, links, code spans — inside each block. There is no single Markdown: CommonMark is the precise specification, GFM layers extensions on top, and every renderer adds more. Nearly every surprise traces back to the first pass, where a blank line, or its absence, is what decides the block you get.

Paragraphs and line breaks

Source	Renders as
Blank line	New paragraph
Single newline	A single space
Two trailing spaces	<code>
</code>
Backslash at line end	<code>
</code>

A paragraph is one or more lines ended by a blank line. Newlines inside it collapse to a single space, so where you wrap the source has no effect on the output — wrap at whatever column you like. To force a visible break, end the line with two spaces or a backslash.

```
One line,  
still the same paragraph.
```

```
New paragraph.\  
Forced break above.
```

Gotcha: trailing spaces are invisible, and most editors strip them on save — a hard break you cannot see silently disappears. Use the backslash form; it survives formatters and code review.

Headings and thematic breaks

Syntax	Result
# Text ... ##### Text	<h1> – <h6>
Text over ===	<h1> (setext)
Text over ---	<h2> (setext)
--- after a blank line	<hr>

ATX headings take one to six #. CommonMark requires at least one space between the # and the text, so #Heading is just a paragraph. Setext headings underline the line above and reach only two levels. Trailing # are optional closers and are stripped.

```
## ATX heading ##
```

```
#Not a heading – no space
```

```
Setext H1
```

```
=====
```

Gotcha: --- on the line directly under text is a setext <h2> , not a thematic break — setext wins that ambiguity in the spec. Always leave a blank line above a horizontal rule.

Emphasis and inline code

Syntax	Result
text or _text_	
text	
text	Both
`code`	<code>

* and _ are interchangeable at word boundaries, but _ never opens or closes emphasis inside a word — that exception exists so snake_case_name survives untouched. A code span takes its content literally; to include a backtick, fence it with more backticks.

snake_case_name stays literal.

*mid*word emphasises, _mid_word does not.

Use `` ` `` to show a literal backtick.

Gotcha: a code span strips exactly one leading and one trailing space when both are present, so ` a ` renders as a . Double the spaces on each side if you need one kept.

Lists

Syntax	Result
<code>- , * , +</code>	Unordered item
<code>1. or 1)</code>	Ordered item
<code>3. on the first item</code>	<code><ol start="3"></code>
Blank line between items	Loose list

Any of the three bullet markers works, but switching markers starts a new list. Only an ordered list's first number matters — it sets `start` and the rest are ignored,

so write `1.` everywhere and let the renderer count. Nest by indenting to the parent item's content column.

- Item
 - Nested, indented to content

1. First

1. Second, still renders as 2

Gotcha: one blank line between items makes the list loose, wrapping every item in `<p>` and adding vertical space. A stray blank line restyles the whole list, not just the item it follows.

Code blocks

Form	Notes
<code>```lang</code>	Fenced; sets the language class
<code>~~~lang</code>	Fenced, tilde variant
Four-space indent	Indented block; no language

Fence with three or more backticks or tildes. The word after the opening fence becomes `class="language-..."` on the `<code>` element. The closing fence must be at least as long as the opening one, which is how a longer fence can contain a shorter one. Indented blocks carry no language.

```
```js
let x = 1;
```
```

indented block, no language tag

Tip: to show Markdown that itself contains a fence, open the outer fence with more characters than the inner one — four backticks around three, or tildes around backticks.

Links and images

| Syntax | Result |
|---|----------------|
| <code>[text](/url "title")</code> | Inline link |
| <code>[text][id] plus [id]: /url</code> | Reference link |
| <code><https://x.com></code> | Autolink |
| <code>![alt](/img.png)</code> | Image |

Reference definitions may sit anywhere in the document — they are stripped from the output — and their labels match case-insensitively. `[id][]` and a bare `[id]` both reuse the label as the text. Wrap a URL containing spaces in angle brackets. An image is a link with a leading `!`, and its `alt` text is all a screen reader receives.

```
[The spec][cm] and <https://commonmark.org>
```

```
[cm]: https://spec.commonmark.org/ "CommonMark"
```

Gotcha: in plain CommonMark a bare `https://x.com` stays literal text. It only becomes a link under GFM's autolink extension, which also links bare `www.` hosts by prepending `http://`.

Blockquotes and nesting

| Syntax | Result |
|--------------------------|-----------------------------|
| <code>> text</code> | Blockquote |
| <code>> ></code> | Nested blockquote |
| A lone <code>></code> | Blank line inside the quote |

Mark each line with `>`. Lazy continuation lets you drop the marker on later lines of the same paragraph. Any block nests inside a quote — lists, code, further quotes. Separating two paragraphs inside one quote needs a lone `>`; a genuinely blank line ends the quote.

```
> First line,  
still quoted by lazy continuation.
```

>

> Second paragraph in the same quote.

Gotcha: lazy continuation extends a *paragraph* only.

A line starting any new block — a list item, a fence, a heading — falls out of the quote unless it carries its own > .

GFM extensions

| Extension | Syntax |
|---------------|---|
| Tables | <code>a b over --- ---</code> |
| Task lists | <code>- [x]</code> and <code>- []</code> |
| Strikethrough | <code>~~text~~</code> |
| Autolinks | Bare <code>https://</code> or <code>www.</code> |

GitHub Flavored Markdown is CommonMark plus five extensions. A table's delimiter row must hold exactly as many cells as its header row, or the whole construct

degrades to a paragraph. Colons in that row set column alignment, and a literal `|` inside a cell is escaped as `\|`.

```
Col	Aligned
a	1
```

- [x] Done
- [] Not done

Gotcha: the GFM spec requires two tildes for strikethrough, but GitHub's own renderer also accepts one. Write `~~text~~` — the single-tilde form works on github.com and almost nowhere else.

Escaping, HTML, and portability

| Construct | Behavior |
|--------------------------------------|--------------------------------|
| <code>*</code> | Escapes ASCII punctuation only |
| <code>&copy;; , &#42;</code> | Decoded to a literal character |

| Construct | Behavior |
|-----------------|------------------------------|
| Raw HTML | Passed through by CommonMark |
| [^1], > [!NOTE] | Neither CommonMark nor GFM |

A backslash escapes any ASCII punctuation character and nothing else — `\a` stays `\a`. Entity and numeric references are decoded, so `*` yields an asterisk that cannot start emphasis. CommonMark passes raw HTML straight through; GFM's tag filter escapes nine tags, including `script`, `iframe`, `style`, and `textarea`.

```
\*literal asterisks\* and \a stays escaped
```

```
&copy; 2026 &mdash; &#42;not emphasis&#42;
```

Warning: footnotes, > [!NOTE] alerts, YAML frontmatter, and math are renderer-specific, not CommonMark or GFM. A README that looks right on GitHub can lose content entirely in another parser.

Further reading

- [CommonMark Spec 0.31.2](#)
- [GitHub Flavored Markdown Spec](#)
- [GitHub Docs — Basic writing and formatting syntax](#)
- [Babelmark — compare Markdown renderers](#)