Engineer

Markdown

plain text with a few marks that mean formatting

What it is

Markdown is a way of writing a document as an ordinary text file in which a handful of typed conventions, such as asterisks around a word for emphasis, a hash in front of a heading, a hyphen in front of each list item and square brackets followed by parentheses for a link, tell a converter which HTML to produce. The name covers both the syntax and the program it first shipped as, files usually end in `.md` or `.markdown`, and the dialects that grew from it, CommonMark and GitHub Flavored Markdown (GFM) among them, all answer to it.

John Gruber developed it in 2004 in collaboration with Aaron Swartz, and wrote the first converter in Perl; its last release, Markdown 1.0.1, is dated 17 December 2004. His stated goal was readability above all: a Markdown document should be publishable as it is, as plain text, and should not look as though tags or formatting instructions had been used to mark it up. The biggest single influence on the syntax, he wrote, was the way people already formatted plain-text email. Anything the syntax does not cover is written as HTML in the middle of the text, with no switch or delimiter, because in his framing Markdown is a writing format and HTML is the publishing format.

A few lines show most of it:

```
The move took **one evening**, in three steps:

- stop the writes
- copy the data with `pg_dump`
- point the [application](https://example.com/app/) at the new host
```

A converter turns that into a paragraph with a `strong` element, a list of three items, a code span and a link, and the source still reads as a note somebody typed.

Use the concept in a decision

Put “What is Markdown?” to work in a real decision. We can help relate the concept to the system you are building or buying. A site that is quick, findable and editable by your own staff, without a bill for every change of wording.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Developed 100 web applications using HTML/HTML5 — 100 web applications built

Developed 100 web applications using HTML/HTML5 Get in touch →

Why it matters

The source is the document, and plain text outlives every tool around it. A Markdown file opens in any editor, compares line by line in version control and reviews cleanly in a pull request. No vendor can retire the format, and moving a body of writing from one system to another is a matter of copying files.

The price is ambiguity. The CommonMark project puts it plainly: Gruber’s description does not specify the syntax unambiguously, and his script had bugs and was last updated in December 2004, so by 2014 there were dozens of implementations, each filling the gaps its own way. Its example is a document that renders one way on a GitHub wiki and another way when converted to DocBook with Pandoc. And because Markdown rejects no input as a syntax error, a divergence surfaces only when somebody looks at the output. The project points to Babelmark 3, which runs the same input through more than twenty implementations to see whether they agree.

CommonMark, started in 2014, answers with a specification written by John MacFarlane that pins each rule to examples of input and the exact HTML it must produce: 652 of them in version 0.31.2, dated 28 January 2024. GitHub defines its own dialect, GFM, as a strict superset of CommonMark, adding tables, task list items, strikethrough, extended autolinks and a filter that disarms a short list of raw HTML tags. Hugo’s Markdown converter, Goldmark, conforms to the CommonMark specification and is compatible with GFM.

Markdown also has a name on the wire. Since March 2016, RFC 7763 registers `text/markdown` for the whole family of syntaxes, with a required `charset` and an optional `variant` parameter that says which dialect a file is written in.

What it looks like in practice

The hand-written pages of this site are Markdown, laid out by Go templates, and the credits page lists both among the tools the pages are made with, beside Hugo, the static site generator. The same page names markdownlint among the linters that keep the source honest, in a quality gate where every diagnostic is an error.

The page on making a site readable to AI agents turns the format round: every page here has a Markdown twin at the same address, reached by adding `index.md` or by asking for `text/markdown`, and the build reads every twin for leftover markup and refuses to publish when one is wrong. The twin of that page is one request away:

```
curl -sL https://engineer.company/ai-ready/index.md
```

Around 50,000 words of software documentation went into Markdown across GitHub, Craft and Confluence, much of it written by hand. It covered the architecture, the processes and the how-to material people kept asking about, was kept version-controlled and findable, and got new people up to speed faster.

Where it goes wrong

Bold prints as asterisks. A reader sees two asterisks on either side of a word that was meant to be bold, and every check stays green, because the HTML is valid and the links still work. CommonMark opens emphasis only on a left-flanking run of asterisks, and a run followed by punctuation, such as the bracket of a link, counts only when whitespace or punctuation stands in front of it. A letter in front keeps the asterisks literal, and so does anything a template engine puts there before the Markdown is parsed: Hugo replaces a shortcode with a placeholder made of letters and digits. This site's Markdown twin template records that removing the space between a shortcode and the bold beside it broke the bold on 60 bullets across three languages. Keep the space, and check the rendered page rather than the source; this site's gate has a check that reads the built HTML for Markdown that reached a reader as punctuation.

The same file renders two ways. A nested list flattens on one system and nests on another, or a hash at the start of a line makes a heading in one converter and stays text in the next, and nothing reports it, because Markdown has no syntax errors to report. The cause is a dialect difference: two converters filling one of the original gaps in different ways. Name the dialect a project is written in, CommonMark or GFM in most cases, use a linter that holds files to it, and run doubtful input through more than one implementation before relying on it.

The Markdown copy of a page carries HTML. A reader of the plain-text version meets raw `svg` elements, or a phone number that opens with a numeric character reference where the plus sign should be. A component that renders to HTML for the web page renders the same HTML into every other format unless it is told otherwise; Hugo looks up a shortcode's template per output format and falls back to the HTML one. This site's shortcode templates record both cases, 49 raw `svg` elements in the Markdown copy of the credits page and the contact page's number escaped for HTML, and a shortcode that writes any markup now gets its own template for the Markdown output. The gate also checks that the Markdown twins carry no raw HTML and no HTML entities, skipping code.

Raw HTML disappears from the page. A block of HTML written into the Markdown is missing from the built page, and the source looks fine. Gruber's syntax lets HTML stand anywhere in the text, but a converter is free to refuse it: Hugo's `unsafe` setting for Goldmark is off by default, so raw HTML mixed into Markdown is not rendered, and GFM escapes a short list of tags such as `script` and `iframe` so that they print as text. Reach for what the generator offers instead, a shortcode or a render hook in Hugo, or turn the setting on deliberately and accept what it lets through.

The linter passes what it was told to ignore. A Markdown file with broken structure gets through review because markdownlint reports nothing. The rules that would have caught it were switched off in its configuration, with no reason written down. This site's development rules record that markdownlint once sat in a JSON file with six rules disabled and no comments, and that when the six were measured, two were hiding real defects. Write the reason beside every disabled rule, in a format that can hold a comment, so that the next person can disagree with it.

Further reading

- CommonMark Spec — the specification, rule by rule, each one tied to examples of input and the HTML it must produce.
- CommonMark — why a specification was needed, how implementations diverged, and who maintains it.
- Markdown, by John Gruber — the original description, the design goal and the Perl script.
- Markdown: Syntax — Gruber's own account of the syntax, including the rule for inline HTML.
- GitHub Flavored Markdown Spec — GitHub's dialect as a superset of CommonMark, with each extension marked.
- RFC 7763: The text/markdown Media Type — the media type, its required `charset` and its `variant` parameter.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how "What is Markdown?" applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/markdown/