Engineer

gemtext

a text format with one link per line

What it is

Gemtext is the document format of the Gemini protocol: plain UTF-8 text in which the first characters of every line decide what that line is, so a client can render a whole page in one pass from top to bottom. Its specification calls it the Gemini hypertext format, servers label it with the media type `text/gemini`, and the files conventionally end in `.gmi`.

There are six kinds of line. A line opening with `=>` is a link, made of an address, absolute or relative, and an optional label after it. A line of three backticks switches preformatted mode on or off, and whatever sits between two of them is shown exactly as written in a fixed-width font. One, two or three hash marks make a heading, an asterisk followed by a space makes a list item, and a greater-than sign makes a quote. Every other line is ordinary text, and the client wraps it to the width of whatever screen it lands on.

```
# Field notes

A text line is a paragraph, however long it runs. The client wraps it.

=> /archive/ Older notes
=> gemini://example.org/ A capsule elsewhere

* A list item
> A quoted line
```

The specification splits those six into three core types that every client must handle (text, links and the preformat toggle) and three that a minimal client is allowed to show as plain text. That is the whole format. It has no bold, no italics,

no tables, no inline images, no nesting and no way to make a word in the middle of a sentence clickable. A parser keeps a single bit of state: whether it is inside a preformatted block.

Gemtext is most often compared with Markdown, which it resembles at a glance. The difference is what each one is for. A Markdown file is source, written to be converted into something else; a gemtext file is already the finished document, and the reader's client is its only renderer. It borrows its one structural habit, the link on a line of its own, from the older Gopher menus.

Use the concept in a decision

Put "What is gemtext?" 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

Choosing gemtext is choosing who controls the page. The author decides structure: which lines are headings, which are links, where the code is. The reader's client decides everything else, from font and line length to colour, spacing and whether the headings become a table of contents. The specification states plainly that the format is not meant to give authors precise control over presentation. For some publishers that is a relief and for others it ends the conversation, and it is better to know which before converting a single page.

The relief is in what disappears. A gemtext page carries no stylesheet, no script, no web font, no cookie notice and no tracking pixel, and nothing in it needs updating except the words. Pages are small and quick to write by hand. Links never fetch anything on their own, because the specification forbids a client from opening a network connection merely to display a link, so a page cannot pull in a third-party image or font behind the reader's back.

The cost is just as concrete. Gemtext reaches only people who run a Gemini client, which is a small audience beside the web's. It has no forms; the one way a capsule asks for input is a status code in the protocol, not anything in the document. And the prose has to work without inline links. A sentence written for

HTML, “see the pricing page for details”, keeps its words and loses its pointer, so writing gemtext well means writing sentences that still make sense when the link sits on the line below.

That is why, for most organisations, the realistic question is gemtext as well as a website rather than gemtext instead of one, produced from the same source by the same build. Asked that way, gemtext costs little to add and doubles as a test of the writing. If a page still reads clearly as a column of text lines and link lines, its structure was in the words all along.

What it looks like in practice

This site is published as gemtext alongside its HTML. The Engineer in a terminal page gives the capsule’s address, `gemini://engineer.company/`, states that every page of the site also exists in the format, and names clients that read it: Amfora, Lagrange and Bombadillo in a terminal, Castor and Kristall in a window. It also gives the reason. One deploy writes one folder that every front door reads, so the gemtext copy cannot drift from the HTML, and a page that reads well as gemtext is one whose structure is real rather than painted on.

How it was built is described in the write-up of mirroring the site to Gemini and Gopher. Gemtext is an output format of the same build, not a separate toolchain. The site generator was given custom media types and output formats, and every page, section and taxonomy term gained a Gemini document beside its HTML and its Markdown twin: 777 of them at the recorded implementation stage, served by a small daemon on the same host. The new formats are marked as non-alternative representations. A separate build comparison measured zero HTML bytes changed when those formats were added; it did not measure later editorial or visual changes.

The templates behind that copy show what converting to gemtext involves. Each inline link is lifted out of its sentence: the words stay where the sentence needs them and the address moves below the text as a `=>` line of its own. Bold markers are dropped, hyphen bullets become asterisks, and a fenced code block passes through untouched, because a Markdown fence and a gemtext preformat toggle are the same three backticks.

Where it goes wrong

Half-lines that never join

Prose arrives in the client as a ragged column of short lines, broken mid-clause. The source was hard-wrapped at 80 columns, as Markdown files often are, and the specification forbids a client from joining consecutive short text lines into longer ones. Write each paragraph as one long line, or have the converter join soft line breaks before it writes the file.

Links that leave the capsule

The first link a reader follows opens a web browser. The converter wrote every link as an absolute `https://` address, and a Gemini client hands those to another program. Write navigation as root-relative paths such as `/about/`, which resolve inside the capsule, and keep one deliberate link per page to its web version.

Code that collapses into a single line

A code sample appears as one long line glued to its opening fence. The converter treated the block as prose and joined its lines with everything else. Split the document on its fences before any other step, pass each preformatted block through as written, and let the toggle line carry at most a short alt text such as the language name.

Markdown that turns into literal characters

Hyphen bullets, double asterisks, underlined headings, nested lists and bracketed links show up in the client exactly as typed. Gemtext recognises only an asterisk and a space for a list item and hash marks for a heading, and it has no inline markup at all. Convert those constructs on the way out, or write the capsule natively.

Sentences that point at nothing

A paragraph says “click here” or “as linked above” and the reader has no idea where to go. The link was lifted out of the sentence, and the words left behind never said where they led. Write link labels that name their destination, so both the sentence and the `=>` line make sense on their own.

Further reading

- A quick introduction to gemtext markup — the project’s own short tour of every line type, with notes for people who know Markdown.

- Gemini hypertext format specification — the normative text: line types, parser state, wrapping rules, the `lang` parameter and a formal grammar.
- Gemini network protocol specification — how a capsule serves gemtext: requests, status codes and the TLS rules a client follows.
- Project Gemini FAQ — the reasoning behind the design, including why links stand on lines of their own.

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 gemtext?” 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/gemtext/