1 Publishing with Texinfo

I used to have a blog where I dumped ideas, experiments and projects. That sense of publishing in public, or working in the open, felt like a commitment that would push me to finish things. But that is not always true. And also that blog made me feel like a millennial :-p, and it was built on a framework my technical instincts didn't really like.

But I miss writing, so enter Texinfo: a familiar face.

Texinfo is the official documentation format of the GNU Project. Most people encounter it through `info` pages and assume it is only for software manuals. It is not.

With a Makefile containing one rule and a small CSS file, Texinfo produces clean, readable HTML from a plain-text source. This blog is built that way.

1.1 Why Texinfo

- **One source, multiple outputs.** The same `.texi` file generates HTML, Info, PDF (via TEX) and plain text. You decide the output format at build time and not at writing time.
- **Semantic markup.** You describe *what* something is: `@code` for inline code, `@emph` for emphasis, `@file` for filenames. The stylesheet decides how it looks.
- **No dependencies.** `texi2any` ships with most Linux distributions and is available on macOS via Homebrew. The build does not require Node, bundlers or a network connection.

1.2 A Minimal Document

Every Texinfo document starts with a small header, then content, then `@bye`:

```
\input texinfo
@setfilename doc.info
@settitle My Document
@documentencoding UTF-8
@node Top
@top My Document

Content here.

@bye
```

That is the entire scaffold. Running `texi2any --html --no-split` turns it into a single HTML file. Add `--css-ref=style.css` to link an external stylesheet.

1.3 Common Markup

A short reference for the elements used most often:

- `@emph{text}` — emphasis (a.k.a. italic), rendered as *text*.
- `@strong{text}` — bold, rendered as **text**.
- `@code{text}` — inline code, rendered as `text`.

- `@example` / `@end example` — a preformatted code block.
- `@footnote{note}` — an inline footnote.[1] The text appears at the bottom of the page with a backlink.
- `@uref{url, label}` — a hyperlink.

1.4 Limitations

Texinfo was designed for software documentation, not long-form prose. It has no native support for bibliographies, figure captions or tables with merged cells. For a short blog post these rarely matter. For a book-length work you would feel them.

The format also does not allow arbitrary HTML inline, and you must use `@html` / `@end html` blocks to escape. Use that hatch sparingly. Overuse makes the source hard to read.

[1] Texinfo is older than the modern static-site ecosystem by decades. The format was originally developed in the 1980s for GNU manuals and remains actively maintained today.

2 Conclusion

Texinfo occupies an unusual place in modern publishing. It is old, minimal and largely unchanged, yet it continues to solve a real problem: turning structured plain text into durable documentation without a large toolchain. For small websites, technical notes and software manuals, that simplicity becomes a strength rather than a limitation.

The modern web often treats publishing as an application problem requiring frameworks, package managers and build pipelines. Texinfo approaches it as a text-processing problem. The source remains readable without special tools, the output formats are generated from the same document and the entire system can be understood in an afternoon.

That does not make Texinfo the right tool for every project. It lacks many features expected from contemporary publishing systems, especially for visually complex layouts. But for writers who value plain text, reproducibility and long-term readability, it remains a surprisingly practical format decades after its creation.

And about the old posts I've written, and now locked in a version-control system, they will occasionally find a home here as well.