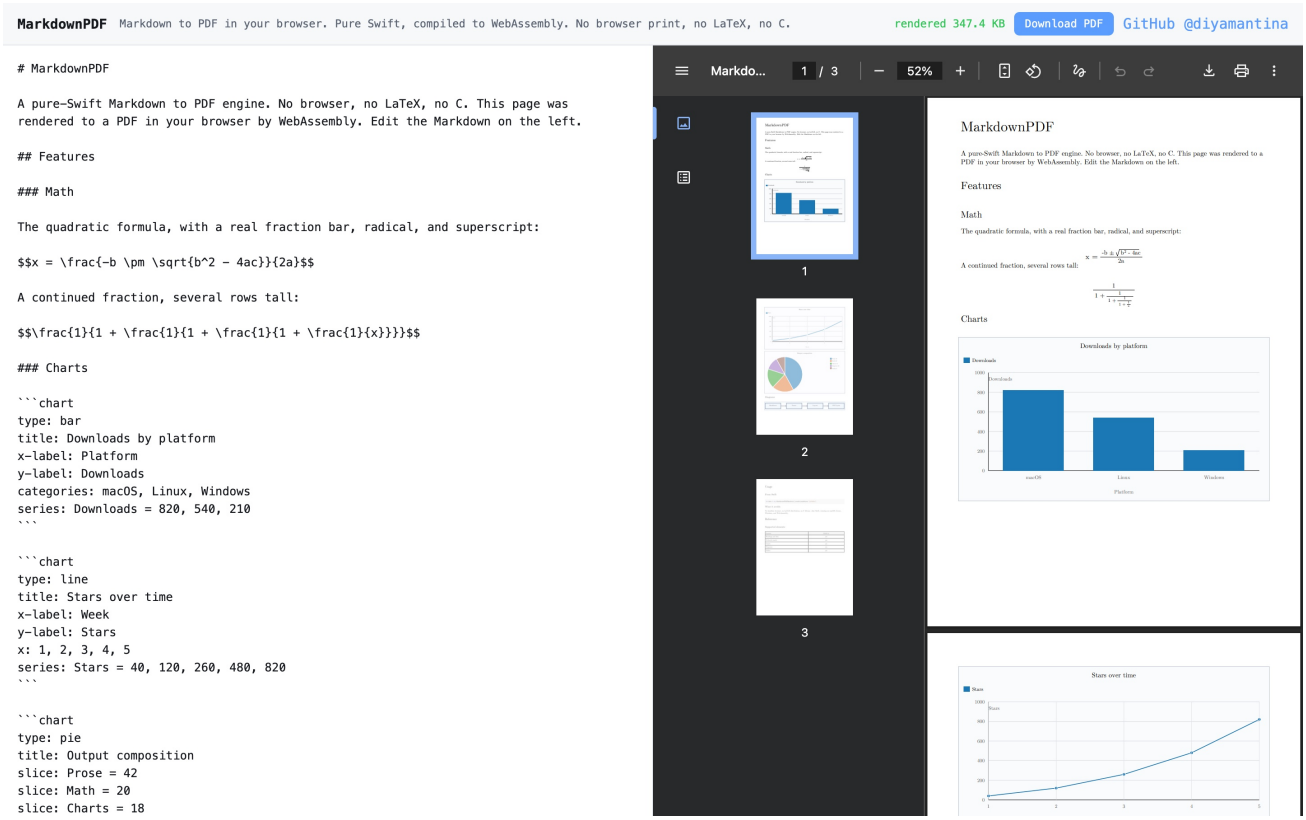
Markdown to PDF in your browser, in pure Swift

There is a playground at pdf.aleahim.com. Type Markdown on the left, get a real PDF on the right. No server is involved. The PDF is produced by a pure-Swift engine compiled to WebAssembly and run inside your browser.



What it is

MarkdownPDF turns Markdown into a PDF with no LaTeX, no headless browser, and no C Markdown or PDF library underneath. It parses the Markdown, lays the document out, and writes the PDF bytes directly in Swift. It runs on macOS and Linux, the core also builds for Windows and WebAssembly, and that last one is what makes the in-browser demo possible.

The reason I wanted this: every Markdown to PDF path I tried pulled in something heavy. A browser to print from, a full LaTeX distribution, or a C library bound into the build. I wanted one Swift package that takes Markdown and emits the PDF, so it drops into a server or an app with nothing to provision on the machine.

What it renders

In one codebase:

- A TeX-style math subset. Fractions, radicals, superscripts, and big-operator limits lay out as real two-dimensional math. The quadratic formula and a several-rows-tall continued fraction are in the default sample.
- Native vector charts: bar, line, pie, and scatter, drawn with PDF path operators rather than rasterized images.
- A Mermaid flowchart subset, drawn as native PDF shapes.
- Embedded TrueType subsetting, DEFLATE compression, tagged PDF, and PDF/UA and PDF/A profiles checked with veraPDF on a fixture corpus.

Headings, lists, tables, code blocks with syntax highlighting, links, and images are all there too.

How it runs in the browser

The browser demo is small. There is a tiny executable that reads Markdown from standard input, renders it, and writes the PDF bytes to standard output. That executable is compiled to wasm32-unknown-wasip1. In the page, a WASI shim runs the wasm, feeds the text area in as stdin, and captures stdout as the PDF, which is shown in an iframe with a download fallback.

Two things were worth getting right. The wasm has to be instantiated asynchronously, because synchronous instantiation is disallowed on the main thread for modules over eight megabytes. And the engine is large, so the playground ships a gzipped wasm, about eighteen megabytes on the wire, and inflates it in the browser. First load downloads it once, then it is cached.

The pieces, and where they live

The work is split into focused packages, each open source under MIT:

- The engine: MarkdownPDF. The parser, layout, and PDF byte serialization.
- The math: MathTypeset. A dependency-free Swift math typesetting package. It is its own package because a static site generator I work on shares it, so the same TeX source typesets the same way whether the target is a web page or a PDF.
- The command line: MarkdownPDFCli. `markdownpdf input.md output.pdf` and a resume variant.
- The playground: pdf.aleahim.com, with source at markdownpdf-playground. The wasm build, the WASI harness, and the page itself.

There are no third-party runtime dependencies. The only package the engine pulls in is my own MathTypeset, which itself has none.

Honest limits

It is early and I say so. Math renders with an embedded Latin Modern Math font, so plus-minus, sigma, radicals, and Greek come out as real glyphs, not ASCII. One wrinkle worth noting: the engine embeds TrueType (glyf) fonts only for now, and Latin Modern Math ships as OpenType CFF, so the playground bundles a TrueType conversion of it (the MATH table is preserved). The full witness test suite runs on macOS and Linux; Windows and WebAssembly are build gates that compile the engine but do not yet run the suite. Text beyond Western European needs a supplied font, and shaping for Arabic and Hebrew is the open frontier where help is welcome.

Try it

The demo is at pdf.aleahim.com, the engine is at codeberg.org/MarkdownPDFHQ/MarkdownPDF. If you render Markdown to PDF anywhere that you would rather not install a browser or a LaTeX stack, this might fit.