Typesetting Sys... Why Am I Doing This Again?

January 19, 2024

Matei Barbu

Reader, meet your document's maker!

Typesetting is the art and science of arranging text on a page in such a way that it not only conveys its intended message, but does so with an *aesthetic grace* that could make even the most Stoic of fonts *blush*. It's a world where kerning isn't a geekish curiosity, but a pleasant adjustment of space between characters. Where leading is not about leadership, but the vertical spacing between lines of text. And where a widow is not just a spouse left alone, but a single word awkwardly sitting at the top of a page, yearning for its lost paragraph.

Thus, typesetting is, in its essence, the meticulous art of ensuring that while the words are being read, the reader is blissfully **unaware** of the typographic conductor waving its invisible baton, guiding their eyes across the sea of letters and symbols.

This ancient craft, which has evolved from painstakingly arranging individual letters in a printing press, shall not be reduced just to text documents. It also ensures that sheet music is arranged for clear interpretation and performance. In presentations, effective typesetting balances visual appeal with information clarity, aiding in audience engagement and comprehension.

Before the start of the 2000s, a name was given to the format behind the digital wizardry of modern typesetting software, *Markup*. And three actors play its role [1] in our movie:

- Presentational markup

  Hides behind the scences of traditional word-processing systems.

- Procedural markup

  Embeddes itself in text providing instructions for programs to process that text.

- Descriptive markup

  Labels parts of the document for what they are (rather than how they should be processed).

While there were speculations of supertypewriters speeding away from the tediousness of markup, we now know how far-fetched such claims were. Keep in mind that typesetting, when done right, should remain **unnoticed** by readers. So how do *document processing system* push writers towards this goal?

Markdown. A game changer?

In 2004, John Gruber and Aaron Swartz presented online [2] their concept for a "text-to-HTML conversion tool for web writers." While its syntax was intended for one purpose, "to be used as a format for writing for the web", it became much more than that. *Gasp*.

With a "less is more" philosophy, it offered a smooth learning curve, making it a favorite for beginners and experts alike. Its one-column flow fits like a glove onto the hands of mobile readers, seamlessly adapting to both light and dark themes settings. Its simplicity has led to widespread adoption in various productivity tools, including popular note-taking apps and generative AI platforms.

Soon, it was used for much more than its authors seemed fit. Extensions to its simple syntax meant that tools like *R Markdown* could typeset books [3]. While others have stayed true to Markdown's intended purpose [4], there was only so much we could do until we hit a dead end…

If you every try to wrap text around images, insert math ecuations, cite sources or even color text, then you face two options:

- Apply a quick and dirty workaround.
    1. Inline HTML for each unconventional block.
- Build a solid and scalable solution.
    1. Write a fully blown transcompiler.
    2. Feature a modular API for extensions.
    3. Build a template for each project you are working on.

> If it looks like a duck and quacks like a duck *but* it needs batteries, you probably have the wrong abstraction.
>
> — Paraphrase of the Liskov Substitution Principle

So, Markdown doesn't need batteries. No, no, no… It **needs a full-blown power generator**!

Besides architectural problems, Markdown syntax is *ambiguous*. Nothing, yes, nothing counts as a "syntax error." As such, CommonMark [5] was created to be a standard, unambiguous syntax specification for Markdown, along with a suite of comprehensive tests. One of its creators has, since then, moved on from Markdown to a "light markup syntax" language, Djot [6], for good reasons.

WYSIWYG

Visual editing tools like Microsoft Word and Confluence (see Figure 1) are *What You See Is What You Get* systems. They separate all the intricacies of their internal markup format from a user's workflow.

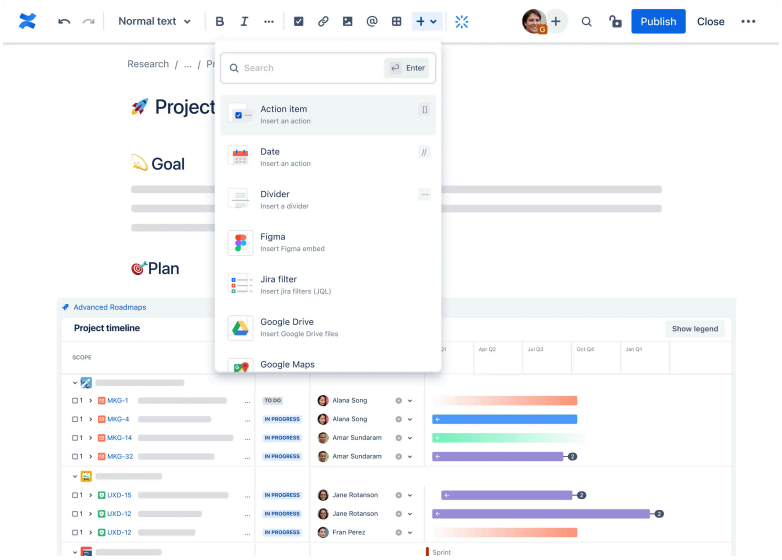


Figure 1: Screenshot of Confluence [7] depicting visual controls for formatting styles, adding images, tables, links, etc. Each button's action is rendered immediately.

GUI tools make you adjust “by hand” quirks and don’t offer programming power to users. Whereas, plain Markdown offers the simplest interface to automate the transition from source to polished documents. Font configurations are set in a template, numbered lists are corrected automatically and there’s no need to drag contents around for ordering. See just how readable the following `.md` source is!

```
# Example Source

## Subheading

3. This is the first list item.
8. This is the second list item.
1. This is the third list item.

Here's some example code:

    return shell_exec("echo $input | $markdown_script");

> And a block quote.
> From the Author,
> With Love.
```

Visual editing software employs presentational markup to assist less experienced users or those unfamiliar with advanced design techniques. So don’t expect them to go out of fashion soon. In fact, most Markdown tools [8] now boast a toolbar featuring classic controls along with a “real-time” previewer (see Figure 2) that stays in sync with the editing line inside the source.

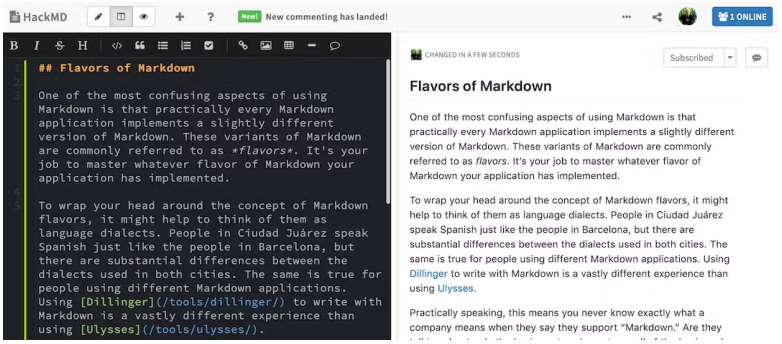


Figure 2: Screenshot of HackMd [8] depicting a side-by-side view of a document’s form.

The E-LaTeX in the room

Initially designed for preparing `man` pages, `troff` (typesetter runoff) was a significant milestone in the history of typesetting. Appearing one decade ahead of Microsoft Word, its procedural, “sweet and sour” markup syntax excelled in text formatting and layout. However, it was constrained by its limited ability to handle complex layouts and graphical elements, particularly for non-technical content.

> I mean, `troff` was collapsing all the time. A lot of the earliest users of TeX had been frustrated by `troff` breaking over and over again, so it had gotten unwieldy.
>
> — Donald Ervin Knuth

Enter Donald Knuth, a renowned computer scientist and mathematician. Dissatisfied with the typesetting quality of his work and others in mathematical texts, Knuth embarked on developing TEX. His vision was to create a system that could produce aesthetically pleasing and technically accurate mathematical typesetting. Although he figured this would take about 6 months, ultimately, it took nearly **10 years**.

In the early 1980s, Leslie Lamport extended Knuth's vision by developing LATEX, a collection of macros that operated on top of TEX. Lamport's goal was to make TEX accessible to a broader audience, enabling users to focus more on content than formatting. LATEX represented a shift towards a descriptive markup approach, allowing users to structure their documents logically. Although useful for creating sheet music and stylish presentations, its companions are primarily found within academic circles.

While a tremendous achievement, LATEX is not devoid of shortcomings.

- Being resource-hungry.

  While editing is faster than any WYSIWYG typesetter, compiling is done in multiple iterations disposing of many offline packages in order to work.

- Having a steep learning curve.

  While better than TEX, even experienced users have trouble deciphering warnings and errors.

- Operating like a Markov algorithm.

  Although Turing-complete, a string rewriting system dosen't appeal to procedural programmers.

- Resisting numerous competitors.

  XƎTEX, LuaTEX, ConTEXt... How do I choose one of these?

On-going projects

Extending beyond the simplicity of Markdown, AsciiDoc emerges as a noteworthy contender. Its key innovation lies in its ability to compile documents into both HTML and PDF formats. It caters to diverse distribution needs, particularly in professional-grade documentation where consistency across online and print media is crucial.

Pandoc, needs no introduction. It is the Swiss Army knife of document conversion, known for its unparalleled flexibility and power. This tool extends Markdown's capabilities, integrating advanced elements like footnotes and YAML metadata blocks, thus elevating it from straightforward to complex. Its most significant contribution, however, lies in its ability to fluidly convert between numerous document formats such as HTML, LATEX, Word, and many others, even presentation formats! This versatility positions Pandoc as a cornerstone in modern digital writing, especially in fields requiring frequent format conversions for publications and collaborations.

Middle ground

Drawing to a close, we're left with a somewhat bitter aftertaste from Markdown. When we push too far such a undemanding syntax it topples over itself. Shortly, we found solace in visual editors, but they ultimately left us feeling restricted. In our search for equilibrium, LATEX emerged as the "golden mean," striking a balance between complexity and functionality, but seems destined to `troff`'s fate.

I no longer expect to find a "silver bullet" typesetting system. Now, I know how to choose the software that benefits me the most. After draging Markdown through mud, I still benefit from it, a lot.

And… Typst

Believe it or not, this document (see Figure 3) wasn't typeset in any popular program. Typst [9] compiles its own language to PDF. Being lightning-fast, cross-platform, open-source and Rust-based it sparked my interest. Far from feature complete it seems to be what I was seeking… *for now!*

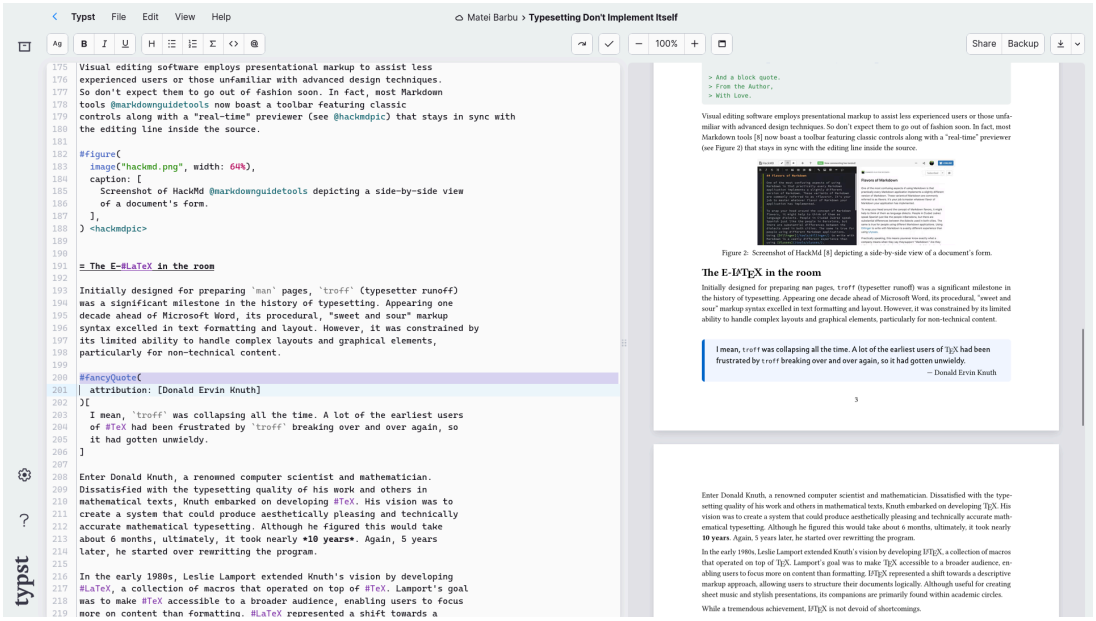


Figure 3: Screenshot of this document edited and previewed on Typst's web editor. Besides the source on the left, a template I wrote powers functions such as `fancyQuote`.

To answer the question in the title, the tools we use shape… And typesetting… To be continued…

Bibliography

[1] J. H. Coombs, A. H. Renear, and S. J. DeRose, "Markup systems and the future of scholarly text processing," *Communications of the ACM*, vol. 30, no. 11, pp. 933–947, Nov. 1987.

[2] John Gruber, "Markdown." Accessed: Mar. 25, 2004. [Online]. Available: https://web.archive.org/web/20040402182332/http://daringfireball.net/projects/markdown/

[3] "Bookdown." Accessed: Jan. 19, 2024. [Online]. Available: https://bookdown.org/

[4] "Docusaurus. Build optimized websites quickly, focus on your content." Accessed: Jan. 19, 2024. [Online]. Available: https://docusaurus.io/

[5] "CommonMark. A strongly defined, highly compatible specification of Markdown." Accessed: Jan. 19, 2024. [Online]. Available: https://commonmark.org/

[6] "Djot." Accessed: Jan. 19, 2024. [Online]. Available: https://www.djot.net/

[7] Atlassian, "Confluence. Your Remote-Friendly Team Workspace." Accessed: Jan. 19, 2024. [Online]. Available: https://www.atlassian.com/software/confluence/

[8] "Tools. Markdown Guide." Accessed: Jan. 19, 2024. [Online]. Available: https://www.markdownguide.org/tools/

[9] "Typst: Compose papers faster." Accessed: Jan. 19, 2024. [Online]. Available: https://typst.app/