Playing with Typst

John Maxwell

2023-11-29

Sometime in early 2023 Typst appeared: a new, from-the-ground up open-source typesetting engine designed to replace the superannuated LaTeX that so much scientific publishing still relies on. LaTeX, for those who don't know, is a set of "user-friendly" macros, dating from the early 1980s, for driving TeX, a mathematics-friendly typesetting and document prep system dating from the 1970s. TeX was developed so long ago that it predates most of the software systems and conventions we now take for granted. Certainly word processors were in their infancy at that point. Unix was still a baby. There was no Internet; no PDFs. Not even PostScript—desktop publishing was a gleam in the eye of crazy young folks at Xerox PARC. And typesetting was still an industry of its own.

TeX *wrote the book* (that's a weird thing to say; Donald Knuth, who wrote TeX, wrote the book) on computer typesetting in those days. Knuth, who is remembered as one of the most influential computer scientists of his generation, worked on everything, from how to drive printing devices right down to how to represent fonts as geometric models. Along the way, he developed the better part of modern algorithms for doing hyphenation and justification in paragraphs, algorithms that are still used by software today. And TeX's real superpower, the reason why it was so successful and has hung around for so long, was setting equations and formulae, especially the ones that operate in two dimensions (that is, not neatly on one line). Knuth reportedly wrote it so that his publisher could typeset his own book properly. Knuth also wrote things like this:

> When certain concepts of TeX are introduced informally, general rules will be stated; afterwards you will find that the rules aren't strictly true. In general, the later chapters contain more reliable information than the earlier ones do. The author feels that this technique of deliberate lying will actually make it easier for you to learn the ideas. Once you understand a simple but false rule, it will not be hard to supplement that rule with its exceptions.

Writing in TeX, or more accurately, with LaTeX (nobody writes raw TeX anymore, we hope) is positively arcane by modern standards. It's effectively like writing computer programs in the middle of your writing that define how to display the text. And yet, generations of grad students in math, chemistry, computer science, and similar disciplines have learned the art in order to complete their theses and dissertation because that's *just how it's done*. While LaTeX fans like to say that LaTeX is preferable to writing in a word processor because you're not encouraged to obsess about styling, but to be honest, that's because you have to write code to do styling in TeX, and LaTeX gives you just enough scaffolding to scaffold most of the formatting for the usual genres: theses, articles, software docs.

So there's history, and lots of it there. It's now 2023, and this thing Typst appears. In principle, it's not that different: you're writing a scientific paper, and you use this to (a) format your equations and such exactly perfectly and (b) use its templates to handle the generic formatting of frontmatter, bibliography, etc.

You're still embedding formatting codes in the middle of your text—and in ways completely different from the highly structural XML/HTML way of doing so. But Typst is a complete re-design and rebuild. If you squint, it looks vaguely like writing LaTeX, but the designers (Martin Haug and Laurenz Mäjde) had decades of hindsight to help them stick to what's essential and avoid the truckloads of cruft that's accrued to the LaTeX way of doing things over the years (e.g., do you need to produce anything other than a PDF? Probably not). So it's way simpler and cleaner, and easier to get up to speed. The Typst.app website gives you a side-by-side window for trying it out, with a lovely debugger to help you compose and understand the code. And then you can just download the source and start cranking out perfectly typeset pages.

Do I care, personally? Not all that much. I am a humanities scholar, and so my need to typeset scientific formulae is approximately zero. In fact, my need to typeset *anything* anymore is pretty minimal, as I've moved into a world in which I can do *almost* anything in a web browser (including printed documents, *including* official letterhead), and for the remaining edge cases, I'll use Adobe InDesign.[1] I do avoid word processors like the plague, but that's for other reasons. So why am I interested in Typst?

First off, I appreciate any tool that actually embodies proper typesetting—and for my money, MS Word is *way* off this standard. So Typst for me is a tool that does the right thing. It does a substantially better job of this than a web browser, even a web browser with a specialized pile of CSS & javascript loaded in it.[2] That's a good start. Second, it's open-source software, which means it's a bit of a bulwark against Adobe's monopoly of the printing world, which is always a good thing. Third, as a modern, streamlined tool, it integrates nicely with my existing pipelines—the same kind of thing that goes from my keyboard to publishing on this blog, for instance.

I doubt I would have taken Typst all that seriously were there not support for it in Pandoc since v3.1.2. I rely on Pandoc regularly to process markdown content into various outputs. Typst source is structured text, so it wasn't a terribly big deal for someone (Louis Vignoli, I think) to contribute reader and writer components for it in Pandoc. The result is that I can take this blog post and fire it through a different pipeline and produce a formatted PDF. That's not really practically necessary, but it isn't hard to imagine publishing pipelines that go to the web, to epub, and to print equally… Typst is an easy and elegant solution for the print part of that.

[1] We figured out how to make InDesign work with XML content, via ICML, so it's not all hand carving.

[2] PagedJS is, AFAIK, the leading CSS-based option for book typesetting.