Using Pandoc and Typst to Produce PDFs

A Tutorial

by JMax

2024-07-10

I recently responded to someone on Mastodon who asked about producing decent-looking PDFs from markdown. I replied eagerly with "OMG Typst!" and linked to my earlier blogpost about developing an entire book layout template for Pandoc and Typst. The response I then received was this was "far beyond" what they need – and on reflection I had to admit that my blog post was a bit, well, *niche*.

I recalled that a decade ago, when Pandoc was new (or at least I was new to Pandoc), I produced a set of tutorials for producing high-quality EPUBs from Pandoc, and that these were quite well received. These days I find EPUB pretty uninteresting,[1] but I'm still using Pandoc for a variety of purposes, and especially using it to drive PDF (that is, "print") production via Typst. Typst emerged as an output option for Pandoc in mid-2023 and since then has gotten even smarter and smoother.

So here's an attempt at a basic tutorial for using Typst with Pandoc to produce beautiful – and easily customizable – PDFs and indeed printed pages.

Now… you know how when you google a recipe online, and you get to a food blog, and you have to scroll through 37 screens of narrative and photographs before they finally give you the recipe? Let's cut to the chase: here's the recipe. Now you can read the rest of this post to learn what it means.

Download simplePandocTypst.template

Some Background

Typst is a new (since 2023) open-source tool for typesetting and page layout. It was designed to replace LaTeX, which has been the standard way of producing scientific publications for decades (in the niche where InDesign and DeskTop Publishing apps fall down because of complex equation formatting). LaTeX has been used – and more significantly,

[1] Check out Pandoc's excellent documentation, and you'll see loads of EPUB production details.

learned – by generations of grad students in the hard sciences, as a way of formatting theses, articles, reports, and so on. But where LaTeX is weird, old, and arcane, Typst was designed as a modern, streamlined tool to produce excellent typography through templated pages.

Now, *Typst.app* is an interactive website which invites you to compose Typst documents and see the typeset results on the fly. This is presumably aimed at those chemistry and math grad students who would otherwise write LaTeX code, often in an interactive web interface just like this. But Typst is *also*, significantly, a downloadable piece of open-source software. Technically, what you're downloading is the 'typst compiler' – you feed it a text file with Typst sourcecode (specifying document properties, formatting, and so on), and it outputs a PDF file to spec.

That's all very well but I'd just as soon not have to learn to write Typst source code at all (as I don't deal with complex mathematical equations in my life). This is where Pandoc comes in.

Pandoc, which bills itself as a "universal document converter" is an open-source tool that will take just about any structured text format as input, and produce just about any other structured text format as output. But in practice, what it does exceptionally well is producing various outputs from **markdown** formatted text files. Which is to say *minimally formatted* text files, since the point of markdown is to be the simplest possible document formatting system. That is markdown's great virtue: the formatting information is there, and is explicit and unambiguous, but it doesn't get in the way of you seeing the text itself.

Using Pandoc with Typst

Now, when I say that Typst is an available output for Pandoc, I mean that we can pass a markdown-formatted text file to Pandoc, and it will convert it to Typst code, which we can then give to Typst to make a PDF. Actually, it's easier than that, because Pandoc will both convert to Typst code *and* ask Typst to render that as a PDF in one step.

```
pandoc mymarkdownfile.txt --pdf-engine=typst -o myPDF.pdf
```

You can try that, and you'll get a nice-looking output. Pandoc uses a default *template* for Typst that defines page size, margins, and some basic typography. Typst right out of the box already does a lot of the work – page numbering and so on. But we can have some fine-tuned control over that by tweaking the template that Pandoc uses. In fact, Pandoc's default Typst-formatted PDF may be good enough for basic use.

Basic document metadata

Typst expects to see some basic metadata about the document you're producing: the title, the author, and the date, at minimum (though more is possible). In a markdown document, this is typically written as a metadata block at the top of the file like so:

```
---
title: "Using Pandoc and Typst to produce PDFs: A Tutorial"
author: JMax
date: 2024-07-10
---
```

Pandoc recognizes this, and so Typst will, by default, format them in a centered header, as if it were a journal article. You don't *have to* include those elements, but if you do, Typst will style them as frontmatter in your PDF.

We can change how it treats those items by tweaking the template, of course, as we can change pretty much anything else: page size, orientation, columns, margins, typography, and so on.

Customizing the template

Pandoc has a hidden cache of all the default templates it uses for different outputs, and you can ask Pandoc to show you its default template for any given format. We can ask for the Typst default template like so:

```
pandoc -D typst
```

You *could* save that into a file, and modify it as you like, and you'd have a new 'custom' template you can use. You can specify your own template instead of the default one by telling Pandoc:

```
pandoc inputfile.txt --pdf-engine=typst template=myTypst.template -
o output.pdf
```

A Starter Custom Template

But here I'm going to short-circuit things, and give you a starter kit for a custom template. This is one that I've developed, bit by bit, over the past few months. It's a simpler version of the book-layout template I blogged about earlier, but it's the same in principle.

You can think of these templates as the equivalent of CSS stylesheets: they define the specs for all the various things in your content. Typst templates are more complicated than CSS, but that's because there's a lot more intelligence here: for instance, you could tell it to start page numbering only after the third right-hand page. There is actually a ton of programmable stuff that you could put into these templates; the one I'm offering here is pretty straightforward so you can go in and muck with it as you like.

For instance, I've set up the basic font as Times New Roman, because it's available on almost every computer; you can change this to any font you have, though. I've set the default paper size to US Letter, but if you're in Europe, you can set this to A4 or whatever

you like. My paragraph styling is to indent the first line of all subsequent paragraphs, but I've included an optional instruction for flush-left paragraphs with space between them if you prefer that.

My template tries to cover a lot of formatting bases: images, footnotes, header and footer, code snippets, etc. You should, of course, consult the extensive (if not always crystal-clear) documentation at https://typst.app/docs/

Download, try it out, modify it. Let me know what you think.

Download simplePandocTypst.template