Org-mode export to Reveal.js and PDF

Adam Thornton

September 16, 2025

Org-mode export to Reveal.js and PDF

If you're me, you think better in a text editor than in Powerpoint.

A few years ago I ran across org-reveal. It was revelatory.

It was also pretty manual.

Automating Emacs

Emacs has had a batch mode forever (as far as I know, anyway).

Clever use of `--load` and `--eval` let you turn Emacs into a batch transmogrifier.

Why Emacs 30?

Emacs 30 lets you use a shebang to treat eshell just like any other shell script. I thought this was going to be useful, but it turned out not to be. But if you wondered why this container was built on Debian testing (in September 2025), "Emacs 30" is the answer.

Automating artifact generation

In essence all you need to do for Reveal.js is to ensure that org-reveal-mode is available. For PDF output you need LaTeX and an org-mode new enough to support LaTeX output.

Then you just call the appropriate export function once you've loaded your org document into the buffer.

Ensuring you have that support

The problem is, no one's CI runner is going to have Emacs with a new org-mode and org-reveal preloaded.

The obvious thing to do is to build a container with Emacs and a LaTeX setup and run it via Docker (or podman).

And the great news is, this has already been (mostly) done by Olivier Berger.

Differences from Berger's work

Berger uses Emacs with X enabled, because he prefers those fonts, and goes to some trouble to avoid running Emacs in batch mode. I don't care: I just get Verdana into the container and set up my PDF export to use it.

I don't care about "plain" HTML export–if I am exporting to HTML, what I want is Reveal.js.

I go a little farther and create a GitHub Actions workflow to allow publication straight to GitHub Pages.

GitHub Action integration

The trick here is getting both local builds and GitHub Actions to work correctly and cache built containers.

In the GitHub CI YAML you will see that the Docker `Build container` step caches to and from GHA, and also uses `--load` to make the image available to susbsequent steps.

In the Makefile the `docker-container` target tries to use the target container to run a trivial command; if it fails it builds that container.

That ensures that, if building locally, you build the container (which is never pushed) unless it's already in your local Docker environment. If building via GitHub Actions, you will at that point have the container because it will have been built or retrieved from cache and then stored in Docker by `Build container`.

PDF Export

Generating PDF output from org-mode is straightforward, at least in the sense that generating LaTeX from org-mode is easy.

In theory, once you have a working LaTeX installation, generating a PDF from that is easy. In practice, that "once you have" is doing a lot of load bearing. Fortunately, this is what containerization is for.

Don't look at how to fix up the installed TeX.

On the bright side, the PDF version looks OK.

Source Code

This project lives at GitHub.

I may at some point make it into a template project, but for now, all you really need to do is clone it and then replace the org document with your own. Fix up the `Makefile` and modify the CSS as you like,

and then `make site` (which is also used as the CI check) will do the trick.