What is Rheo?

The simple answer is that Rheo (*ree-oh*) is a new and more flexible way to produce and publish digital documents. The less simple answer is that Rheo is a typesetting and static site engine based on Typst. This guide explains both *how* to use Rheo, and *why* it might be for you.

Rheo allows you to produce a website, a fixed-size document, and an adaptive document from a single set of source Typst files. It allows you to do something similar to LaTeX—except that Typst is *much* simpler to write, and we can produce a greater number of formats with it. The documentation that you are reading now, for example, was typeset with Rheo. As a result, you can read it as:

- HTML - as a website for browsers.
- PDF - as a fixed-size document for printing.
- EPUB - as an adaptive document for e-readers.

Who should use Rheo?

If you write anything as simple as a blog or as complex as a dissertation or monograph in Typst, Rheo enables you to publish it in multiple formats. If you are willing to learn a little bit of syntax, you can turn a piece of writing into a website, an adaptive document, and/or a printable document.

Some of the things you can write and publish with Rheo include:

- A blog
- A paper
- A dissertation
- A book manuscript
- A novel
- A textbook
- Technical documentation

Rheo is for anyone who has ever spent regrettable hours formatting citatons, fighting with LaTeX, who has experienced the limitations of Markdown, or who wants to benefit from the richer writing experience that Typst makes possible (more on this in the next section). It is for students and teachers, humanists and scientists, bloggers and novelists.

If you have only ever used Microsoft Word to author text, or haven't heard the phrase 'markup language' before, we recommend first familiarizing yourself with Typst via the excellent tutorial. This should give you a good intuition for what Typst is—a markup language similar to but also more powerful than Markdown—and why you might want to use Rheo to typeset your documents.

Why do we need Rheo?

Rheo (*ree-oh*) is an open source typesetting and static site engine for Typst. It is a typesetting engine because it produces typeset digital documents such as PDF and EPUB, and a static site engine because it produces websites that don't require communication with a custom backend server, but rather are self-sufficient sets of files that can be natively opened a browser (static sites). Most static site engines these days employ Markdown, a markup format that is approachable and pretty generic, allowing

folks who are not familiar with or otherwise don't want to deal directly with the required file formats of the web— HTML, CSS, and Javascript— to write blog posts and other content which can then be pumped into a static site.

As useful as it is, Markdown has its ambiguities. For one, there isn't a standardized syntax for citations or footnotes. Though extensions exist that can produce these, they are not supported in the core Markdown syntax, meaning that it's not *really* Markdown and can't be relied upon to work in all contexts that support Markdown. Markdown is great when using hypertext (hyperlinks, images, etc). It's not so great when it comes to things like tables, figures, and math.

Typst is a markup language that integrates with plain text, like Markdown, making it easy to adopt and joyful to write. Unlike Markdown, however, it is also a Turing-complete programming language with a modern type system, meaning that it is possible (though not necessary) to express sophisticated conditional logic controlling where and how text is rendered. Typst has a concrete and concise syntax for footnotes and citations, and can express visual constructs such as tables, figures, colors, and mathematical formulas. It was developed as a modern alternative to LaTeX, Leslie Lamport's legendary 1980s addition to Donald Knuth's original '78 Tex typesetting system. For the past 40 years, LaTeX has been the most expressive way to produce PDF documents, rendering it the de facto standard for academic and scientific publication. In the past few years, Typst has become the most promising and powerful alternative to LaTeX due to its maintainers' effort to build out a reliable PDF compilation toolchain.

In 2025, Typst added experimental support for HTML compilation. Though there are still many features in Typst that will only produce meaningful output in the PDF toolchain, the HTML toolchain now supports all of the essential features for academic documents in the humanities: text decoration, headings, hyperlinks, footnotes, and citations.[1] This makes it an extremely good replacement for Markdown as a markup language in a static site engine, and so: enter Rheo.

Rheo is a CLI (command line interface) that produces PDF, HTML, and EPUB simultaneously from a folder of Typst documents. It is a static site engine because it can produce a fully valid website: all it needs is a folder containing valid Typst. Rheo also provides mechanisms to combine multiple Typst files into a unified EPUB or PDF, making it a tool that improves the experience of writing books, dissertations, or any other long-form text in Typst. On the other side of the same coin, Rheo allows you to produce an offline version of a website such as a blog written in Typst through its PDF/EPUB toolchain.

Rheo allows you to compile multiple Typst files that link to each other into a single output, adding what is needed (relative linking) in order to make Typst an ideal markup language for writing static sites. Typst is the most elegant and flexible way to typeset PDF documents today; Rheo extends Typst's capabilities, allowing you to additionally typeset EPUBs and generate static sites from the same source.[2] Naturally, this blog post

[1] I qualify this with 'in the humanities' as scientific papers often require tables, figures, and mathematical markup. These features are on Typst's roadmap for html, but are not yet available at time of writing (January 2026).

[2] EPUB is on Typst's roadmap, but is not yet natively supported.

was written in Typst, and this site was made with Rheo. If you're already convinced, feel free to jump ahead to Getting Started to download Rheo on your system and start writing.

The philosophy of Rheo

Rheo is a prefix or combining form in English that originates from the Greek word *rheos* (ῥέος), meaning flow, stream, or current. Rheo flows Typst documents into a number of concurrent output formats in PDF, HTML, and EPUB. But other meanings lurk beneath the surface of this basic idea. Sarah Pourciau has argued that the oceanic is a deep-rooted metaphor in computing, as all computation at some level seeks solid space in a sea of digital noise (Pourciau 2022). From Alan Turing's partial solution to David Hilbert's *Entscheidungsproblem* in the universal machine, to Claude Shannon's information theory, to Leslie Lamport's ordering of events in a distributed system, the key issue at hand is how to carve out clarity from uncertainty and confusion. Writing has played a magisterial role in calming the storm of imprecise thought. Long before computation arrived on the scene, the written word has served as the steward of reason, in the Western world and beyond, from Mesopotamian cunieform to Twitter. *Nota bene* ('Take note'): that writing can also herald chaos and confusion doesn't invalidate its capacity for spreading sensibility.

Rheo is a tool that facilitates the production and publication of documents following from the original vision of the Internet as a mechanism for lively and reasonably unfettered academic exchange, rather than the densely commercial space of platform capitalism that it has become. It should not be so difficult, given the extraordinary capacities of software and hardware today, to make a piece of writing publically available in a plain and pleasant format. That there exist digital humanities initiatives measured in months and years to bring books to the web as basic websites is a clear sign that something has gone awry.[3]

Rheo aims to enable the publication of more books, blog posts, and papers without the necessary capitalist ceremony of creating an account on Substack, Medium, or Squarespace. A website without any interactive elements such as forms, online marketplaces, or comments should be simple to set up, as it isn't rocket science in 2025 thanks to all the hard work that folks have put into Internet protocols and web standards. It *should* be simple to create a PDF or EPUB for sharing with colleagues or collaborators—as simple as it is to send an email.

This is the vision of the world to which we at the free computing lab aspire, and in search of which we have built Rheo. Rheo is the first installment in a larger set of writing tools we aim to build, which will include processes for collaboratively drafting documents, constructing and working with digital libraries, and more.

[3]There are many exciting and experimental ways of presenting text and other conent in the digital humanities. But we think that it should be easier that it currently is to publish and distribute books digitally, simply and straightforwardly.

Installation

The easiest way to install Rheo is from crates.io, the Rust language's package manager. If you don't already have Rust/cargo, you will need to install those first. Open your terminal, and run the following command:

```
cargo install rheo --locked
```

Rheo is packaged as a standalone binary, and doesn't require any version of Typst on your system. (Note that even if you already have Typst on your system, Rheo will use its own embedded version of the compiler.) Refer to Rheo's source code for more information and installation options.

Firing up

With Rheo we can produce a static site, a PDF, and an EPUB from a Typst document. Let's create a directory with a single Typst file in it:

```
mkdir project_uno
touch project_uno/index.typ
```

As one of Rheo's outputs is a static site, the landing page will default to one named 'index'. (If this file doesn't exist, Rheo will present you with a basic listing of all the other files in the site.) Let's put some Typst in the `index.typ` file:

`project_uno/index.typ`

```
= Project uno

Project uno is a writing project.
```

Rheo aims to keep out of your way as much as possible, and doesn't require that you add any special syntax or metadata to your files to work. This single Typst file we need to get started. Provided you've already installed Rheo on your system, we can compile the project. You can tell Rheo to compile a folder by pointing the `compile` command at it:

```
rheo compile project_uno
```

This command produces a `build` subdirectory inside the base directory which contains a PDF, an EPUB, and a static site (HTML and CSS). Your project folder should now look like so:

```
.
├── build
│   ├── epub
│   │   └── project_uno.epub
│   ├── html
│   │   ├── index.html
│   │   └── style.css
│   └── pdf
```

```
│       └── index.pdf
└── index.typ
```

It's a little tiring to have to run the compile command every time we make a change, though. Let's spin up a development server to see live changes across all output formats as we edit the source:

```
rheo watch project_uno --open
```

The `--open` flag here indicates that we'd like to open the output using our system's default applications. Provided you have an EPUB reader on your system (if you don't, we recommend installing bene), you should now have a PDF, an EPUB, and a website in front of you. As simple as that!

Scaling up

Let's kill that process (with Ctrl-C). Rheo compiles documents from across your project directory towards EPUB, PDF, and HTML simultaneously, whereas the Typst compiler typically takes just one Typst file and produces one kind of output.[4] Let's add a couple of files to our project and link between them:

`project_uno/about.typ`

```
= About

Project uno is an incredible writing project that will transform the way we
understand the world.
If you want to be involved, see the #link("./contact.typ")[Contact page].
```

`project_uno/contact.typ`

```
#let email = "myemail@mydomain.com"
= Contact

To learn more about project uno, email me at #link("mailto:" + email)[#email]
```

And let's also link to the two new pages on the index page:

`project_uno/index.typ`

```
= Project uno

Project uno is a writing project.

- #link("./about.typ")[About]
- #link("./contact.typ")[Contact]
```

Now let's run Rheo again, but this time let's only build the HTML and EPUB outputs:

```
rheo watch project_uno --html --epub --open
```

[4] Typst allows you to break up your projects using modules, but still requires one entrypoint. Rheo, by contrast, enables multiple entrypoint files, corresponding to multiple standalone pages in a static site.

Note how the relative links are working across both the EPUB and the PDF. Relative linking is one of the key features in Rheo that enables you to build richer static sites and EPUBs beyond using just Typst. All of Typst's other features such as variables are fair game, too, as Rheo just uses Typst's compiler under the hood.

Adding a config

One issue with the EPUB that is currently being produced is that the `index.typ` section shows up last, after the `about.typ` and `contact.typ`, as Rheo orders files lexicographically by default. This is probably not what we want, as the index page acts as a sort of table of contents in our writing project currently.

To sophisticate the way that Rheo produces outputs, we can add a rheo.toml config at the base of the project directory:

`project_uno/rheo.toml`

```toml
version = "0.1.0"

[epub.spine]
title = "Project Uno"
vertebrae = ["index.typ", "about.typ", "contact.typ"]
```

This config uses the notion of a spine to indicate a custom order for the sections. We'll learn more about these later on in this documentation.

Let's run the `watch` command again, this time with all outputs like the first time:

```
rheo watch project_uno --open
```

Great! The EPUB order is fixed. We now, however, have three distinct PDFs that are being created: one for each page. This is because Rheo defaults to producing one PDF per file in the project directory. We can configure Rheo to merge files together into a single PDF output by specifying a PDF spine, as we did with EPUB, and setting the `merge` attribute to `true`:

`project_uno/rheo.toml`

```toml
version = "0.1.0"

[epub.spine]
title = "Project Uno"
vertebrae = ["index.typ", "about.typ", "contact.typ"]

[pdf.spine]
title = "Project Uno"
vertebrae = ["index.typ", "about.typ", "contact.typ"]
merge = true
```

Before we run this again, let's also clean the outputs in the build directory, as we don't need those individual PDFs that we produced anymore:

```
rheo clean project_uno
rheo watch project_uno --open
```

Now we have a fully featured writing project, with nice-looking and orderly outputs in PDF, EPUB, and in HTML!

Rheo allows you to write documents in plain Typst without requiring any additional syntax or metadata. Because Rheo can combine multiple files into unified outputs, however, we need a way to reference other files in the same Rheo project.

The syntax for these relative links in Rheo should be familiar, as they look just like regular Typst links, but reference a `.typ` file in the same directory as its target:

```
#link("./another-section.typ")[Another section]
```

When you compile a project with Rheo, relative links to other Typst documents in the same directory will be resolved and transformed according to the output format. What a relative link transforms to depends on both the output format and your Rheo configuration, as using features such as spines affects the control flow between your source Typst and output formats.

- In **HTML**, relative links become `<a>` tags that point to the relevant html page.
- In **PDF**, relative links either become plain text (if input Typst is not combined, and thus produces one PDF per source document), or links to the relevant sections in the output document (if your config specifies a spine with the `merge` attribute set).
- In **EPUB**, relative links become links to the relevant sections in the EPUB.

Relative linking is what allows Rheo to produce fully functional static sites. It is also a feature that you can use to help you organize large writing projects. (Note that the Typst `import` keyword works as you would expect in Rheo, and so can also/still be used as a mechanism to modularize projects.)

Rheo is a CLI that produces PDF, HTML, and EPUB simultaneously from a directory of Typst source documents. The directory that contains your Typst is called the **project directory**, and you can compile it like so:

```
rheo compile path/to/projectdirectory
```

In general, there are two ways to configure Rheo:

1. By passing **flags** directly to the CLI command.
2. By specifying configuration in a `rheo.toml` file at the root of the project directory.

If you compile a Rheo project directory without a `rheo.toml` file, the following default settings will be applied to compile your project.

```
version = "0.1.0"
content_dir = "./"
build_dir = "build"
formats = ["pdf", "html", "epub"]
```

```toml
[epub.spine]
vertebrae = ["**/*.typ"]
title = "[project directory name]"
```

To point Rheo to a rheo.toml file that is not at the root of the project directory, specify it directly via the CLI:

```
rheo compile path/to/project --config path/to/config
```

Build directory

Rheo produces outputs in a simple directory structure with one subdirectory for each kind of output. By default, Rheo produces all outputs (PDF, HTML, and EPUB) in a `build` directory instide the project directory:

```
build/
├── epub
│   └── blog_post.epub
├── html
│   ├── portable_epubs.html
│   └── style.css
└── pdf
    └── portable_epubs.pdf
```

The build directory path is calculated *relative to the content directory*. This is important, as if you change the content directory, then your build directory path will become relevant to that directory.

Configuration

CLI flag

You can specify a build directory with either the `compile` and `watch` commands:

```
rheo compile path/to/project --build-dir path/to/build
```

`rheo.toml`

The build directory is specified at the top level of the `rheo.toml`:

```toml
build_dir = "custom_build_directory"
```

Content directory

By default, Rheo will search your entire project directory for Typst documents. You can, however, indicate a specific subdirectory that Rheo should use if you prefer. This can be helpful for structuring projects, as it allows you, for example, to keep a separate `drafts` folder that Rheo will not compile.

The default content directory is the same path as the Rheo project directory. It is important to note that if you specify a custom content directory, all other configuration such as `build_dir` and `spine` globs will operate *relative to the content directory*.

Configuration

`rheo.toml`

A custom content directory can be specified at the top level of the `rheo.toml`. The path is calculated relative to the project directory:

```
content_dir = "pages"
```

Formats

By default, Rheo produces three different output formats simultaneously: **PDF**, **HTML**, and **EPUB**. There are cases in which you may only want to produce one of these formats, however, or to exclude one format because your project either cannot support or does not require it.

PDF

Typst, the programming language and compilation toolchain that underwrites Rheo, natively and fully supports PDF.

HTML

Typst experimentally supports HTML. This means that not all Typst syntax will translate to a meaningful HTML structure. The most common features in everyday prose are all supported, however, such as text markup, links, headings, footnotes, and citations. For more information on which features are currently supported in Typst's HTML export, refer to the HTML export tracking issue.

EPUB

Typst does not yet support EPUB, but it is supported in Rheo. As EPUB export is on Typst's roadmap, Rheo will track this feature closely and look to integrate with it when it lands in the future.

Configuration

CLI flag

You can constrain Rheo to producing one or more formats by passing one or more of the following flags to `compile` or `watch`:

```
rheo compile path/to/project --pdf
rheo compile path/to/project --html
rheo compile path/to/project --epub
```

`rheo.toml`

You can also specify formats at the top level of `rheo.toml` in an array that contains one or more formats. The default, if `formats` is not specified, is an array with all three formats:

```
formats = ["pdf", "html", "epub"]
```

Spines

A spine in Rheo is the backbone or 'table of contents' of Typst source files that should be compiled to an output format. It takes its name from the epub specification, in which the spine articulates—or *reticulates*— the set and order of chapters included.

You can specify a spine's vertebrae for any output format using an array of glob strings in `rheo.toml`:

```
[epub.spine]
title = "My epub"
vertebrae = ["intro.typ", "*.typ"]
```

Notice how the first entry `intro.typ` is a specific file, whereas the second `*.typ` captures a range of files. When a glob string captures a range of source files, they will be ordered lexicographically in the spine.

EPUB

An EPUB must have a spine in order to be valid. By default, Rheo will infer the following spine if not specified:

```
[epub.spine]
title = "[project folder name]"
vertebrae = ["**/*.typ"]
```

PDF

By default, Rheo generates one PDF per Typst source file. You can specify a spine for the PDF format in order to reticulate multiple source documents into a single output PDF by indicating the vertebrae and setting `merge` to `true`:

```
[pdf.spine]
title = "My reticulated pdf"
vertebrae = ["intro.typ", "*.typ"]
merge = true
```

In a PDF generated in this way, relative links will resolve to internal document links that point to the relevant section.

HTML

Rheo does not currently support customizing HTML spines. The default spine uses all Typst files:

```
[html.spine]
title = "[project folder name]"
vertebrae = ["**/*.typ"]
```

When Rheo generates HTML, it injects a default stylesheet into the generated static site for a simple, modern, and mobile-friendly aesthetic. 'Screening the subject' is a website generated with the default Rheo stylesheet for reference.

You can fully customize the stylesheet by adding a `style.css` at the root of your project directory. Note that if your project contains a custom `style.css`, *none* of the styles in the default stylesheet will be applied. If you want to build on the default styles, copy and paste the default stylesheet into the `style.css` file in your project directory.

Frequently Asked Questions

What is the difference between Typst and Rheo?

Typst is a markup/programming language that provides its own toolchain which includes a CLI. You can use the Typst CLI to compile one Typst document to one kind of output file:

```
typst compile source.typ # compile to PDF
typst compile --features html --format html source.typ # compile to HTML
```

Rheo compiles a *project folder* to three outputs—PDF, HTML, and EPUB—concurrently. It allows you to configure how certain source files should be merged (to produce a 'combined' EPUB or PDF file, for example, via spines), and also allows you to enrich certain outputs (such as HTML via custom CSS) with non-Typst content. Rheo supports EPUB natively, which is not currently supported by the upstream Typst CLI (though it is on the roadmap). In summary, Rheo is a opinionated way to manage writing projects with Typst.

How do I read EPUBs on my system?

Mileage varies greatly on EPUB reading niceness across systems! If you're interested to learn more, we have written more about this here. If you don't have a good EPUB reading experience currently, we recommend trying bene, an EPUB reading system that we are developing.

Who maintains Rheo?

Rheo is developed by the Free Computing Lab, an academic research consortium that researches the nature of computing freedom. If you're interested to learn more or get involved, you can join our Zulip.

Can I contribute to Rheo?

Yes! Rheo is written in Rust and developed in public through Github. You can track development and submit issues or requests for features through that platform. While in principle we welcome community pull requests, it's best to join our Zulip and ask about it first, to confirm that your work will not go to waste.

Bibliography

Pourciau, Sarah. 2022. "On the Digital Ocean." *Critical Inquiry* 48 (2): 233–61. https://doi.org/10.1086/717319.