Software. Notty

John Doe · June 19, 2025

Notty (NOte Taking with TYpst) is a software for taking scientific notes in the spirit of Forester, using Typst as the markup language.

1. Design thoughts of Notty

There are already a bunch of great software exploring the possibilities of taking scientific notes. However, Notty tries to be different, and takes its own approach towards that goal. These differences reflect some thoughts behind the design of Notty.

1.1. Typst finds a sweet spot for taking scientific notes

There are already plenty of note taking tools, and many of them have more or less support for taking notes of scientific thoughts. The bad news is that most of them, if not all, do not meet the criteria of taking scientific notes.

1.1.1. Taking scientific notes need modular reusable snippets for mathematics

Scientific notes usually contains mathematical contents that are typeset in some language, most commonly LaTeX. As the note base grows, shared snippets would appear in clusters of notes. These snippets, if not made reusable as macros or functions in the typesetting language, will become unmaintainable just like how a program without usage of variables and functions are unmaintainable. In the meantime, making all the snippets accessible globally risks leakage of abstraction and namespace pollution. Hence, these typesetting snippets should act just like definitions in programming languages: reusable by referring to their name, and scoped under a module system.

1.1.2. Taking scientific notes needs full-power mathematical typesetting

Most of the note-taking software, especially those based by web techniques, has limited support for LaTeX simulation through MathJax or KaTeX. However, these simulations never satisfy one who tries to has any fine-grained control over mathematical typesetting. Anyone who wish to typeset mathematics in their notes seriously would need the full-power of it. This is not necessarily achieved using LaTeX, but at least one needs to use something that declares itself to be a full-power solution.

1.1.3. Markdown is not suitable for scientific content producing

Given that Taking scientific notes need modular reusable snippets for mathematics and Taking scientific notes needs full-power mathematical typesetting, it is obvious that Markdown, without extensions, does not satisfy any of the two principles, and any of the software-specific extensions that tries to solve these problems does not really solve

them, just causing migration friction. This is because mathematics is alien to Markdown — it is just not part of it, thus there cannot be effective integration.

1.1.4. Pure LaTeX is not suitable for scientific content producing

Many might argue against this, but I think that pure LaTeX, that is, just LaTeX itself, is not suitable for scientific content producing. What I agree is that LaTeX is suitable for serious scientific content *publishing*. In that scenario, fine-grained control over typesetting is important, and following the popular standard and utilizing the ecosystem is also important. LaTeX is probably the best one for that purpose. However, I think there are more stuff besides publishing in scientific content producing — one needs to effectively take notes of scientific contents, create new scientific contents not only for publishing (for example, private note, temporary thought, manuscript, etc.), and manage these contents. And the following disadvantages prevent LaTeX to be suitable for these purpose: its compilation is too slow, thus one cannot preview their input and reflect upon it immediately; its language is far from modern, thus it is hard for one to be able to grasp it so that they can build their own utilities on it; also, it is too heavy to be integrated into other tools.

Typst, as a typesetting software with a domain-specific script language, or as a script language with typeset content as one of its basic value type and output target, has the ability to create modular, reusable snippets with enough control over typesetting. Its markup language and script language, by both producing content as values in the language, are congruent at a higher level. And its instant-preview is really impressive.

1.2. Alternative choices for scientific notes

There are other tools designed for taking notes about scientific thoughts, just like how Typst finds a sweet spot for taking scientific notes. The most amazing one is Forester. Anyone who is interested in that topic should look into it.

2. How to set up Notty

As for now, Notty is essentially the combination of a set of Typst templates and a build script written in Python. So, there is no real installation — one just clones the repository of Notty as a starter template, edit the site configuration and maybe the template, and write their own notes.

```
git clone https://github.com/hanwenguo/notty.git
chmod +x build.py
./build.py --help
```

The build script is intended to be executed directly with execution permission like above instead of by `python build.py`. You must have Typst, uv and ripgrep installed on your machine to run the build script like that.

After cloning, you would see the following structure.

```
notty/
├── _template/     # Typst templates of Notty
│   ├── site.typ   # Site configuration
│   └── ...        # Other templates
├── public/        # Resource files to be copied to output directory
│   └── ...
├── html/          # Default output directory for HTML files
│   ├── ...
│   └── pdf/       # Default output directory for PDF files
│       └── ...
├── typ/           # Note files in Typst
│   └── ...
└── build.py       # The build script
```

All of the above directories is the default configuration, which can be overridden by editing the corresponding constants at the beginning of `build.py`. Particularly, the output directories for HTML and PDF files are independent. Thus, for example, you can put them both into a `dist` directory at the same level. That they are nested by default is just because of my personal preference.

You should also look at `site.typ` to check possible configuration options.

3. Writing in Notty

Every note should start with the following template:

```
#import "/_template/template.typ": template, tr, ln
#show: template(
  title:      [Title],
  date:       datetime(year: 2025, month: 08, day: 19, hour: 22, minute:
13, second: 29),
  tags:       (),
  identifier: "20250819T221329",
)

# the content
```

Technically, the `tags` field is optional, and the `identifier` field can be arbitrary instead of a date string. To link to other notes, use `#ln("notty:id")[text]`. To transclude other notes, use `#tr("notty:id", hide-metadata: true, open: true)`; the last two parameters are optional and values here are default values. `hide-metadata` means does not display metadata like date and author under the title, and `open` means show the content for the transcluded note when outputting to HTML. Currently, it is recommended to create hierarchy of notes only through transclusion.

3.1. Use Emacs denote package to write in Notty