An alternative to LaTeX? Designing and implementing an extendable markup language

Eli Adelhult
adelhult@student.chalmers.se

Jonathan Widén
widenjo@student.chalmers.se

October 4, 2022

Background

LaTeX has for a long time been the de facto standard for writing in many areas of academia. And for good reason. The underlying TeX typesetting system is very ergonomic when writing complex maths equations and also provides unmatched control over document layout. However, this isn't the only thing that users want. In most cases, people only want an easy way of combining text, images, and equations to produce documents with relatively simple layouts, and there is no simple solution to this in LaTeX itself. The system is heavily designed around typesetting and does not providing good ergonomics for writers.

This is, unfortunately, a situation where the typesetting system is a hinder. It is too complicated for the users to handle by themselves, so users resort to various packages to handle it for them. But since every package works differently users resort to searching for solutions online, and when the result isn't what they expect, they wrap one package output in another package. The end result is a long preamble of endless packages which the they know little about, just that "it works" when it does work, and when it doesn't, unhelpful errors deep within the typesetting system are printed without giving a clue as to what the problem is.

Additionally, LaTeX is designed to target paper. This isn't a problem per se, it is very useful to have powerful typesetting when the goal is to print a paper or to make a book, but when the target is to produce a document which will be viewed on a digital device this is far from optimal. Instead of being limited to papers, a computer may dynamically combine content from multiple files or the Internet, include additional media such as videos, hide and show content dynamically and much more that is far from possible when targeting paper and thus far from possible when using LaTeX.

There are other alternatives of course, HTML[1], Markdown[2], Roff[3], Orgmode[4] and more recently experiments like Nota[5] and Djot[6] to name a few. However, most development and research focus on document markup languages has been put on lightweight languages intended for wiki-systems or note taking apps, with large differences in terms of extensibility and portability. There is still a big and interesting space to explore. Especially when it comes to designing languages with stronger programming facilities and support for third party packages to extend the capabilities of the language.

Project description

This project intends to design and implement an experimental document markup language to explore ideas related to languages and user-friendly tooling. We hope to find a good middle ground for a language that lies somewhere in between Markdown and LaTeX when it comes to complexity and expressiveness. To simplify the scope, we will opt for simpler controls over layout and typesetting than LaTeX. The main focus areas for the language will be:

- A powerful extension system that will enable adding additional capabilities as seen fit, similar to packages in LaTeX.
- A stable and robust document generation system, which resolves depending parts of the document in the correct manner.
- High portability so that the language may be used consistently with different editors, on different platforms (and maybe on the web), and with possible integration into other software.

On the technical side this project would include:

- Writing a parser for the language.
- Transpiling into a popular presentational format such as HTML and/or PDF (potentially using the popular tool Pandoc).
- Embedding a virtual machine (perhaps WebAssembly) or a scripting language interpreter, to allow for portable external packages that add extra capabilities to the language.
- Designing a scheduling mechanism for the document generation. I.e. solving the problem of which order references, external packages, table of content etc. should be generated.

We would suggest implementing the project in Rust or Haskell but are of course open for other suggestions, both regarding the choice of language and implementation related details.

In the end of the project we hope that we together have built a useful tool for generating documents, perfect for writing your daily notes, interactive notebooks, blog posts and hand-ins.

[1] `https://en.wikipedia.org/wiki/HTML`
[2] `https://en.wikipedia.org/wiki/Markdown`
[3] `https://en.wikipedia.org/wiki/Roff_(software)`
[4] `https://orgmode.org`
[5] `https://nota-lang.org`
[6] `https://djot.net`

Suggested reading

An introduction to the experimental language Nota.

A New Medium for Communicating Research on Programming Languages. Will Crichton, Stanford University.
`https://willcrichton.net/nota`

General thoughts about designing domain specific languages as well a discussion related to the document preparation languages PIC and EQN.

Brian Kernighan on successful language design. University of Nottingham.
`https://www.youtube.com/watch?v=Sg4U4r_AgJU`

An interview that discusses the importance of interlinked documents and unexplored ideas in document presentation tooling.

Ted Nelson explores what computers could've become. Devon Zuegel, Notion.
`https://www.notion.so/blog/ted-nelson`

Thoughts on parsing Markdown.

Beyond Markdown. John Macfarlane.
`https://johnmacfarlane.net/beyond-markdown.html`

Authors

Eli Adelhult (D) and Jonathan Widén (D).

Relevant programs

D, IT and DV.

Recommended prerequisites

Basic knowledge about LaTeX and HTML as well as some programming experience. But, most importantly an interest or curiosity for language design and research.

Supervisor

Magnus Myreen (examiner of the compiler construction course) has agreed to supervise the project.