BlaTeX

Rushi Shah
TJHSST

Abstract

Write your blog in LaTeX.

Introduction

Markdown and HTML are the standard tools used to write your every day tech blog with. But they have pretty weak support for embedding mathematical formulas, and are not conducive to writing for an extended period of time. Plus, they aren't even Turing complete! So instead of writing posts in Markdown, my project allows writers to create posts in LaTeX (a powerful markup language) and generate a static-site that can be deployed to any hosting service (like TJ's servers).

Background

Typically, static-sites like blogs are built with a tool called Jekyll. This Ruby project compiles Markdown posts into a static HTML site. BlaTeX is similar, it is a static-site compiler written in Haskell that compiles LaTeX posts into a deploy-able site. The following description has been edited to highlight the minor differences between Jekyll and BlaTeX :

~~Jekyll~~ BlaTeX is a simple, blog-aware, static site generator. It takes a template directory containing ~~raw text files in various formats~~ LaTeX files~~, runs it through a converter (like Markdown) and our Liquid renderer,~~ and spits out a complete, ready-to-publish static website suitable for serving with your favorite web server. ~~Jekyll also happens to be the engine behind GitHub Pages, which means~~ you can use ~~Jekyll~~ BlaTeX to host your project's page, blog, or website from GitHub's servers for free.

Another popular content-management system for blogging is Wordpress. However, Wordpress posts are edited in the Wordpress Administration Panel as "What you see is what you get". Neither Markdown nor LaTeX are "What you see is what you get", and thus Wordpress caters to a completely different audience (a group of mostly non-technical authors).

Development & Techniques

Requirements. This command line utility requires the Haskell Platform and LaTeX. The Haskell Platform includes GHC (a Haskell compiler) and Cabal, which is Haskell's primary package manager. LaTeX is needed to build LaTeX posts into their resulting PDFs. After the Haskell Platform has been installed, installing BlaTeX is simple:

```
> cabal install blatex
```

Overview. The project is a command line utility that is written in Haskell.

Limitations. One potential limitation of this project is that it only compiles links to PDF files. Because LaTeX can't generate HTML, the blog will only be a series of interlinked PDF files rather than a series of interlinked HTML files. This is not necessarily a limitation, but it may interrupt the status quo of typical blogs made of HTML pages.

In order to parse the user's LaTeX files into an abstract syntax tree that can be consumed by Haskell, this project uses a Haskell package called `HaTeX`. This is a wonderful package (most of the time). The problem arises in a specific parsing instance. In LaTeX, a math environment is defined between two dollar-signs (`$MATH GOES HERE$`). In order to use dollar-signs as actual dollar-signs (like "You owe me $5"), you use an escape character before the symbol (\). This is an exception to the math environment rule. However, HaTeX is not programmed to catch this exception gracefully. It matches pairs of dollar-signs and treats everything in between as math. If there are an odd number of dollar signs, or only one (even if they are escaped properly) HaTeX will just flat out fail to parse the entire file and thus BlaTeX will similarly ignore the post.

Dates are formatted in `DD MONTH YEAR` which contributes to the minimalist, classy theme of PDF blog posts; this date format is also used in the internal implementation. However, this becomes an issue if multiple posts are authored on same day because they will not be displayed in the correct order.

Research Theory and Design Criteria

Developmental Procedures

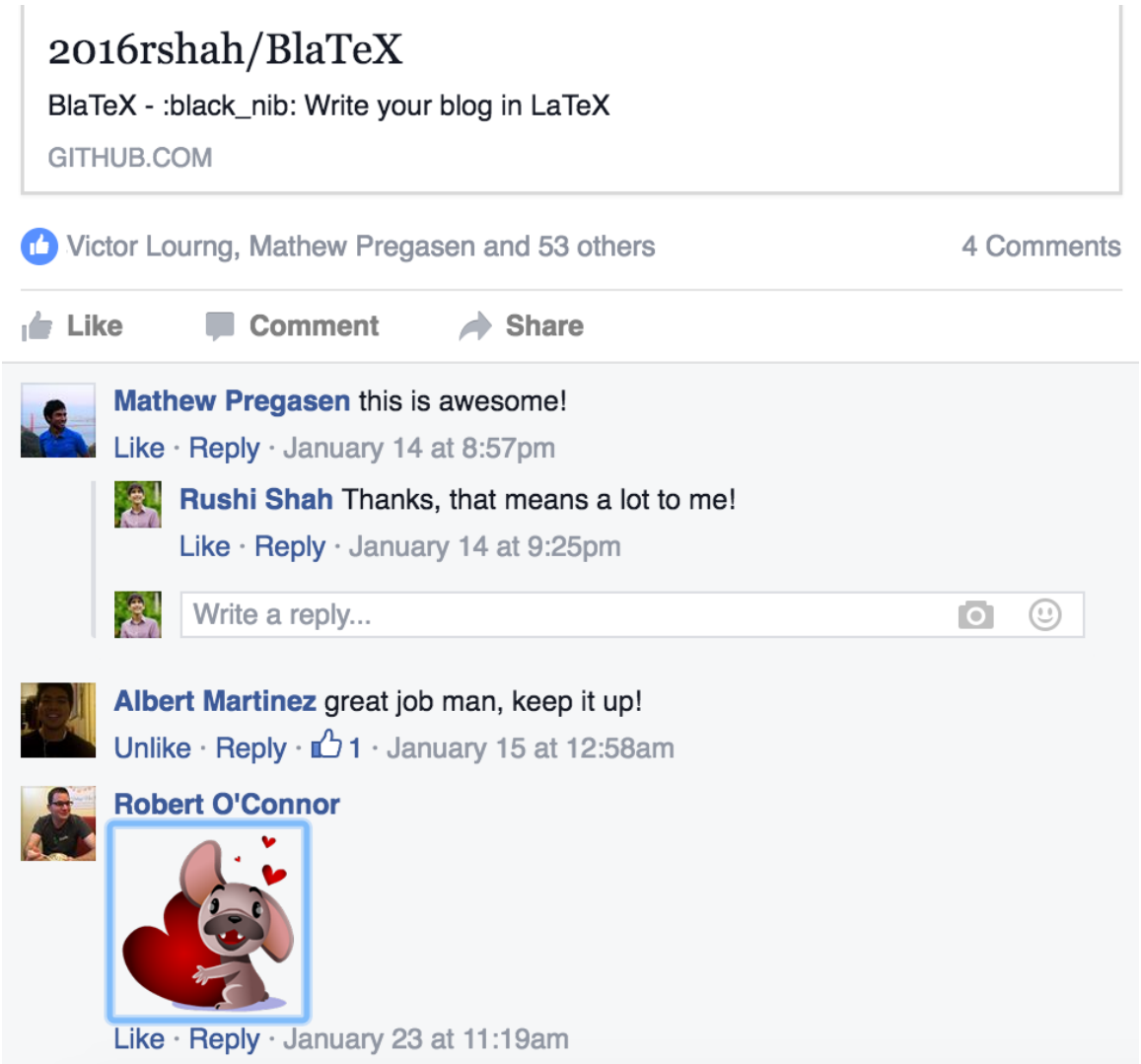
Internal Testing and Analysis

This tool was written in Haskell, a statically-typed, functional, compiled programming language. The compiled nature filtered out most insidious bugs in the code. Haskell development is also strongly integrated with using a Haskell REPL. Thus, each function was experimented with in the REPL after being written and before being pushed to production.

User Interface Testing and Analysis

In order to test this project, I used it to create and maintain `http://rshah.org/blog`. Through my own use I discovered multiple improvements that could be made to the user interface and experience, which I promptly implemented.

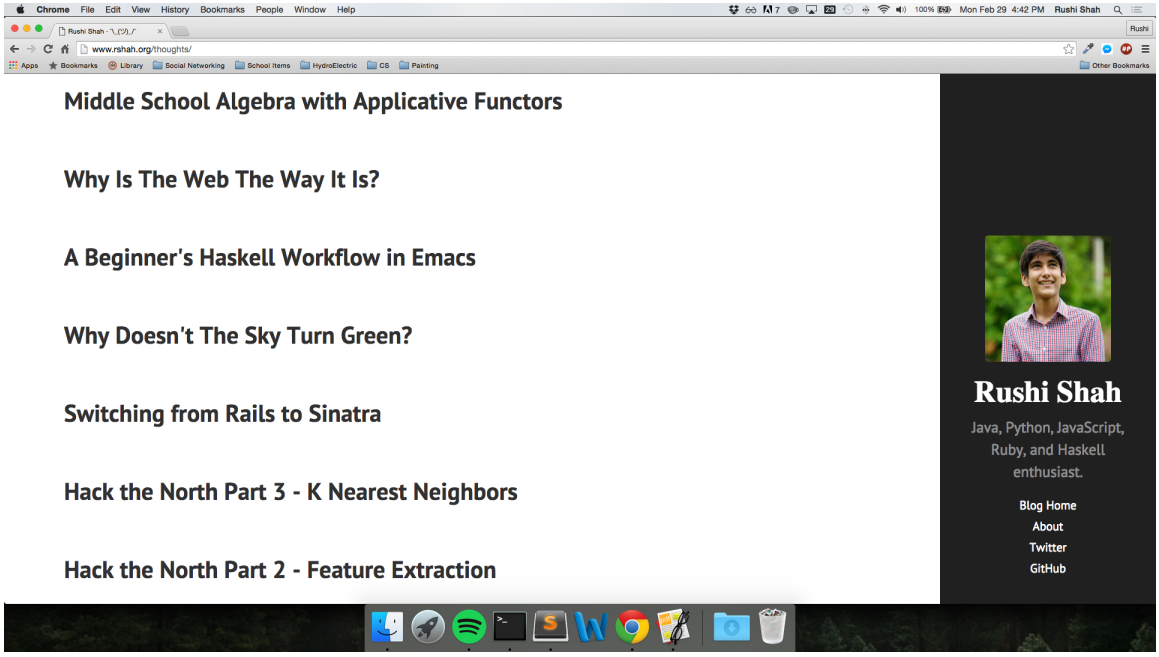
I also posted my project in a Facebook group for functional programming and recieved an overwhelmingly positive response:



It is also an open source project on Github, and met a similarly positive reaction from that community (with 17 Github "stars").

Visual representation of Results

My blog at `http://rshah.org/blog` is a visual representation of a static site compiled with BlaTeX.



Results, Discussion, and Conclusion