LaTeX Jekyll Blog: Trials and Tribulations

April 20, 2025

All I wanted was a simple way to type up my notes/musings in LaTeX, but it took a suprising emount of effort to get it set up. I am admittedly pretty stupid, so most of my issues here were probably user error, but it took me the better part of a Saturday to get this set up properly.

The workflow I settled for is as follows

1. write all my `.tex` files in neovim as I would normally
2. Use pandoc to convert to `.tex` → `.html`
3. Use a python script to remove pandoc-generated `.css` from the `.html`
4. deploy the whole thing with jekyll

Its not that complicated but it took a lot of experimentation to figure this out. The code is in the repo hosting this site. In particular I struggled finding a good way to convert `.tex` → `.html`. Theres too many options: LaTeXML, TeX4ht, Pandoc, plasTeX...etc. I tried using make4ht at first which rendered all the equations as images, then I needed a script to put them all in the right folders when building... it got complicated.

But the pandoc solution is pretty good. Im converting with pandoc in standalone mode and theres just two small issues. The first is an annoying `<!DOCTYPE...` string that it sticks at the top, and then gets rendered in the site, but its simple enough to remove that. The second is in standalone mode it generates its own `.css` which is hard to override later (it might not be hard, but im an idiot. I dont know how `html` or `.css` work). So with a simple python script we can make it work

```
import re
import sys

def clean_html(html_content):
    html_content = re.sub(
        r"<!DOCTYPE[^>]*>\s*", "", html_content, flags=re.IGNORECASE
    )
    html_content = re.sub(
        r"<style.*?>.*?</style>",
        "",
        html_content,
        flags=re.DOTALL | re.IGNORECASE,
    )
    return html_content
```

```
if __name__ == "__main__":
    input_file = sys.argv[1]
    output_file = sys.argv[2]

    with open(input_file, "r", encoding="utf-8") as f:
        html = f.read()

    cleaned_html = clean_html(html)

    with open(output_file, "w", encoding="utf-8") as f:
        f.write(cleaned_html)
```

this deletes the annoying string, and deletes the css. This way the cite css is applied to the pandoc generated `html`.