Personal website built using GitHub Pages with Jekyll. Custom utility code was written to build Markdown web pages from MS Word documents.

The purpose of this project was to build a personal website to host my work. After considering several options I settled on using GitHub Pages to host the website. This choice was made for a few reasons: 1) there are amazing opensource frameworks such as Jekyll that have been developed to run on top of GitHub Pages, making development much easier, and 2) hosting a website on GitHub Pages is free.

In this document I will walk through how I set up my website and some of the nuances I learned. Before diving in, I first want to thank the people whose incredible work I have built on top of, especially [otiliastr](https://github.com/otiliastr/otiliastr.github.io) whose website provided a template for my own.

The repository for my website is located [here](https://github.com/chrisnielsen/chrisnielsen.github.io) and the repository for the utility code written to convert the MS Word documents into Markdown web pages is located [here](https://github.com/chrisnielsen-utilities/word2markdown)

# GitHub Pages Website

My website was built for GitHub Pages using Jekyll. [Jekyll](https://github.com/jekyll/jekyll) is a static site generator that is written in Ruby. Many different Jekyll themes have been developed, each providing a unique stylistic website template. See [here](https://jekyllthemes.io/) for a large list of available themes. Specifically, the theme that I chose for my website was based upon the [al-folio](https://github.com/alshedivat/al-folio) template, which was designed as a theme for academics to host their content. This theme has been used for personal websites as well as websites for labs, courses, and conferences. The `al-folio` repository has been forked many times and adapted for specific use cases. The exact theme for my website was forked from [here](https://github.com/otiliastr/otiliastr.github.io).

It is incredibly easy to set up your own GitHub Pages website. Simply find a website template that you like, fork the repository, and rename the repository `your-GitHub-username.github.io`. Any repository that is named `your-GitHub-username.github.io` will get automatically hosted as a GitHub Pages website.

For the website template that I used, there are a few details that must be understood. The first, and most important, is that any changes made to the website source content must be deployed before they will be seen on the live website. This deployment step is performed using the bash command `./bin/deploy`. On Windows, I ran this command using the Git Bash command line with Ruby version `2.7.4`.

The deployment operation works by constructing the static website in the `master` branch using source material that is in the `src` branch. \*\*It is critical that you should only be making changes in `src`. During the deployment operation, the deployment script will switch the branch to `master`, and upon completion will switch the branch back to `src`. If the deployment operation fails, you will need to manually switch the branch back to `src`. This is very important to know so that you don’t accidently make changes to `master` instead of `src`.\*\*

Another important detail is that the path to each page is defined by a unique permalink in the page header. To add a new folder to our website (e.g. a folder containing Markdown files) we must ensure that the folder name starts with an underscore (e.g. `\_tutorials`), and that a permalink to the folder is added in the `\_config.yml` file as a collection. Check out the `\_config.yml` file [here](https://github.com/chrisnielsen/chrisnielsen.github.io/blob/src/\_config.yml#:~:text=collections%3A,news%3A) for examples of this.

# Building Markdown Web Pages from MS Word Documents

Most of my writing is performed in MS Word using MathType for equation editing. Therefore, to accelerate the process of getting these written documents onto website, it is important to have an efficient system of converting the MS Word documents into Markdown files that could be rendered on the website. To achieve this goal, I built a Python utility that could perform this conversion process. The repository located [here](https://github.com/chrisnielsen-utilities/word2markdown) contains the Python notebook to perform the processing. See the repository `README.md` file for more details.

My general workflow when using this utility consists of the following (\*\*IMPORTANT: remember to specify `src` branch before running script\*\*):

1. When I am ready to publish a `.docx` document on the website, I copy it over to the appropriate subfolder in the `input` path specified in the `process\_documents.ipynb` notebook.
2. Ensure that the copied document meets all the formatting rules specified in the `word2markdown` README.md located [here](https://github.com/chrisnielsen-utilities/word2markdown/blob/main/README.md)
3. Run the `process\_documents.ipynb` Python notebook ensuring that the correct website repository path is specified
4. Push the code to the `src` branch and deploy the code using `./bin/deploy`