Using Jekyll to create a basic personal website

Doug Brinkerhoff

November 18, 2019

This tutorial outlines the steps necessary to create a basic personal website using github pages and jekyll. github pages will automatically use jekyll to render and serve the code that you keep stored in a specially-named github repository.

Creating a github pages repository

To begin, you'll need to create a new github repository, according to a very specific naming convention:

```
{your-github-username}.github.io
```

Once this is complete, you can clone the repo that you just created to your local machine. The first thing that I like to do is to populate a .gitignore file, so that my .swp files (etc.) don't get tracked. Open a file called .gitignore, and add the following lines

```
.swp
_site
```

Initialize the website

The first thing that is typically needed for a website is a index.html file. Create this file and add

```
---


---
Hello World!
```

The dashes are the markdown frontmatter, while the second part is just HTML code. Add this file to the git repo and push to github via

```
git add index.html
git commit -a -m "Added index.html"
git push
```

After a minute or two, if you use a browser to navigate to your-username.github.io, you'll see your html code hosted there.

Adding a theme

Of course we would like to avoid having to build an entire website (css stylesheets and all) from scratch. Github pages allows us to do this using remote themes. There a bunch of themes available online, and we can access most of them seamlessly using github pages and jekyll. We just need to ensure that it knows where the theme is located, and also that we have te necessary Ruby dependencies. First, let's open _config.yaml and add the lines

```
remote_theme: mmistakes/minimal-mistakes@4.17.2
plugins:
- jekyll-include-cache
```

The first line tells jekyll to use the remote theme minimal mistakes (you could use a different one, of course), while the second line specifies a ruby gem needed to use this theme (jekyll-include-cache, which caches certain expensive website elements like sidebars). To use this gem, we'll need a Gemfile as well. Open a file called "Gemfile" and add the following lines

```
source "https://rubygems.org"
gem "github-pages", group: :jekyll_plugins
gem "jekyll-include-cache"
```

Make sure you add both of these new files to your github repo. Finally, to ensure that the theme gets used, open index.html, and add a layout line to the markdown frontmatter

```
---
layout: home
---
Hello World!
```

Push to github, and in a few moments you should be able to access your nice new site.

Developing offline

It's pretty annoying to develop online, especially given the lag between when you push, and when github-pages renders changes to your site. You can also develop locally by using the command

```
bundle exec jekyll serve
```

Note that you might have to install some dependencies for the above command to work

Populating the site

Note that we haven't added any information to this thing yet. For example, by default the title so far is just the site URL. We can change this globally by setting up our _config.yaml file

```
locale: "en-US"
title: "My academic webpage"
name: "Doug Brinkerhoff, esq."
description: "A personal academic website demonstration"
url: "https://douglas-brinkerhoff.github.io"
search: true
logo: "/assets/images/brinkerhoff_headshot.jpg"
author:
  name             : "Doug Brinkerhoff"
  avatar           : "/assets/images/brinkerhoff_headshot.jpg"
  bio              : "I am an Assistant Professor of Computer Science at the University
  location         : "UM"
```

The above command does a few things: sets the english version used by the website, the title, the author, sets up a logo and author avatar (found in a new folder, assets/, that you'll need to create and add to your github repository), and a few other things.

We could also use the _config.yaml file to set up some defaults for our various webpages. For example, we could define the "pages" group, which always displays author information on the left side.

```
defaults:
  # _pages
  - scope:
      path: ""
      type: pages
    values:
      layout: single
      author_profile: true
```

Adding additional pages

We might want to add additional pages, for example a link to a CV. This is pretty easy. We just need to build a new markdown file for each page that we'd like to make. We'll store them in the _pages directory. Make the _pages directory and open a new file inside it called cv.md, and populate it wit ht the following lines

```
---
permalink: /cv/
```

```
---
Publications:
Brinkerhoff and Dunbar (2022), "Cats on ice", Journal of Feline Cryogenics.
```

Once this file is created, we need to edit our _config.yaml file so that jekyll knows where to find it. Add the following lines

```
include:
  - _pages
```

Finally, we need to setup the links in the header. This is done via the file _data/navigation.yaml. Create this file, and add the following lines (note the indentation):

```
main:
  - title: "CV"
    url: /cv/
```

We can use the same procedure as above to make other pages as well.

Blog posts

Finally, we can use this platform as a blog. Each blog post should be stored in a new directory called _posts, in its own file with the naming convention

```
YYYY-MM-DD-title.md
```

with format

```
---
title: "Title of the post"
---
Content
```

In principle, anything following this format, should show up on the first page. Due to the eccentricities of this particular theme, we need to add two things to the _config.yaml file.

```
defaults:
  # _posts
  - scope:
      path: ""
      type: posts
    values:
      layout: single
      author_profile: true
      read_time: true
```

```
      comments: true
      share: true
      related: true

paginate: 5
paginate_path: /page:num/
```

Feel free to experiment

The above short tutorial should get you to a working but very bare-bones page. You should feel free to experiment with other themes and additional components. Be creative!