Jekyll

Jekyll can be installed by following the Installation Instructions hosted on the official website. So if you are on Ubuntu Linux,

```
sudo apt-get install ruby-full build-essential zlib1g-dev
echo '# Install Ruby Gems to ~/gems' >> ~/.bashrc
echo 'export GEM_HOME="$HOME/gems"' >> ~/.bashrc
echo 'export PATH="$HOME/gems/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
gem install jekyll bundler
```

After running the above to install Jekyll, we just need to pick a project or theme to start our Jekyll server with. Check out GitHub or Google for some Jekyll themes, and clone one. Keep in mind to store this repository into a location on your host where you'd like to store the root of your blog, since we will use this repository to start our Jekyll server it will store all of our configuration files.

```
git clone https://github.com/streetturtle/jekyll-clean-dark
cd jekyll-clean-dark

# Build the site with the contents in the current directory
jekyll build

# Serve the site on a webserver and detach the process from this shell
jekyll serve --detach
```

Running `jekyll build --watch` on an active shell will check for any changes to the sites root directory and build them out to the published site.

When creating a new post, `bundle exec jekyll post "Name"` can be ran to create a draft post. The same command has various other uses that will help in the early stages of a blog -

```
Subcommands:
  docs
  import
  build, b            Build your site
  clean               Clean the site (removes site output and metadata file) without building.
  doctor, hyde        Search site and print specific deprecation warnings
```

```
help                Show the help message, optionally for a given subcommand.
new                 Creates a new Jekyll site scaffold in PATH
new-theme           Creates a new Jekyll theme scaffold
serve, server, s    Serve your site locally
draft               Creates a new draft post with the given NAME
post                Creates a new post with the given NAME
publish             Moves a draft into the _posts directory and sets the date
unpublish           Moves a post back into the _drafts directory
page                Creates a new page with the given NAME
```

Specifically, page, (un)publish, post, draft, serve, new, and build are the commands we will use heavily.

When generating a new post using `bundle exec jekyll post "Name"`, you might notice at the top of the new post generated in `./_posts/` there is a block describing the page to Jekyll. This is an important block and can be used to customize how the page is displayed based on the arguments provided. For example, below is a default new post `testpost`

```
---
layout: post
title: testpost
date: 2019-09-01 12:00
description: A test page
tags:
- test
---
# Test
```

The above block does nothing but describe our page to Jekyll, up to our first header that is actually output to our post `# Test`. The layout is described in a corresponding file stored in the `./_layouts/` directory.

If we wanted to add a custom table of contents, for example, when running github.com/allejo/jekyll-toc Jekyll theme we can simply add an argument to our header and it will create a table of contents with anchored links to each topic header automatically, just by adding `toc: true` below.

You can also customize it by styling `.toc` class in `**theme.scss**`

```
layout: post
title: testpost
date: 2019-09-01 12:00
description: A test page
```

```
tags:
- test
toc: true
```

Now if we use markdown carefully Jekyll will automatically create a nice table of content based on the content of our post, and the structure / sizes of the headers for each topic within. (The above solution is based on github.com/allejo/jekyll-toc)

To display raw code, we'll need to use some Liquid syntax -

```
{% highlight md %}
{% raw %}
code
{% endraw %}
{% endhighlight %}
```

Here, we should also be sure to define the langue for syntax highlighting with `{% highlight md %}`

When trying out new themes, some images or symbols may not work correctly. In chasing these down, we will need to use a bit of HTML, CSS, Liquid, and Mardown. For example, I noticed a symbol or image that was used for bullet points in a list of tags was broken on my specific theme, appearing as an empty box instead. To track this down, `tree -L 2` was useful in learning the layout of this unfamiliar project quickly. Eventually, through viewing the files related to tags within my theme, I found that the sidebar itself was an `include` in the below statement of `./_layouts/post.html` -

```
<div class="col-md-3 hidden-xs">
  {% include sidebar.html %}
</div>
```

So, this pointed me to check out the `./_includes/` directory, where I found the below file - `./_includes/sidebar.html`

```
<div class="sidebar ">
  <h2>Recent Posts</h2>
  <ul>
    {% for post in site.posts limit:5 %}
    <li><a href="{{ post.url | relative_url }}">{{ post.title }}</a></li>
    {% endfor %}
  </ul>
</div>
```

```
<div class="sidebar">
  <h2>Tags</h2>
  <ul class="sideBarTags">
    {% assign tags = (site.tags | sort:0) %}
    {% for tag in tags %}
    <li>
      <a href="{{ '/tag/' | append: tag[0] | relative_url }}" data-toggle="tooltip" data-placement="right" title="{{
tag[1].size }}">
        <span>{{tag[0] }}</span></a></li>
    {% endfor %}
  </ul>
</div>
```

The file above pointed me to the CSS classes below associated with each of the tags (`<ul>`) shown by the sidebar

```
<div class="sidebar">
  <h2>Tags</h2>
  <ul class="sideBarTags">
```

I knew these would be stored in the `./assets/` directory within the root of our Jekyll project, where I found `./assets/css/themes.scss` contained the below CSS statement `content: '\f02b';` - This was the symbol in the sidebar of my theme that was causing issues

```
//*************************************************************
//                    Sidebar
//*************************************************************
.sidebar li {
  margin-top: .7em;
  line-height: 1em;
}
ul.sideBarTags {
  list-style: none;
  li:before {
    content: '\f02b';
    font-family: 'FontAwesome';
    float: left;
    margin-left: -0.8em;
  }
}
```

By changing it, and also tweaking some other settings below, I was able to improve the look of the sidebar.

```
//*********************************************************
//                    Sidebar
//*********************************************************
.sidebar li {
  margin-top: .7em;
  line-height: 1em;
}
ul.sideBarTags {
  list-style: none;
  li:before {
    content: '-';
    font-family: 'FontAwesome';
    float: left;
    margin-left: -0.8em;
  }
}
```

Revision #2
Created 1 September 2019 06:21:22 by Shaun Reed
Updated 4 June 2020 16:13:08 by Shaun Reed