Maintain Websites Using Hugo

This document assumes you have Hugo 0.52 or newer installed. Use the `hugo version` command to determine which version you have access to. Once the steps below have been followed to set up Hugo we will migrate an existing site so that it can be managed by it.

Set up Hugo

- Create a directory structure for your new website: `hugo new site <your directory's name>`
- Switch to YAML config files instead of the default TOML: `mv config.toml config.yml`
- Edit the newly generated configuration file:

```
languageCode: "en-us"
title: "Your site's name"
```

- Refer to the official documentation for more information about the default directory structure
  - When Hugo runs it creates a `resources` directory which isn't covered in the official documentation. It is a resource cache where generated files, such as images, are stored.
  - To make use of Hugo's asset processing features a top level `assets` directory needs to exist
  - The remaining files for your site that won't be processed by Hugo should go in the `static` directory
  - More information about asset management:
    - https://gohugo.io/hugo-pipes/introduction/
    - https://gohugo.io/hugo-pipes/minification/
    - https://gohugo.io/hugo-pipes/fingerprint/
- Create a new `assets` directory and copy your style sheet to that location:

```
mkdir -p assets/css
cp ~/path/to/your/style.css assets/css/
```

- To minify and fingerprint our `style.css` use the following lines in your template

```
{{ $style := resources.Get "css/style.css" | minify | fingerprint }}
<link rel="stylesheet" href="{{ $style.Permalink }}">
```

- Remove any unneeded directories and create three new ones:

```
rm -rf archetypes data themes
mkdir -p layouts{_default,partials,shortcodes}
```

- Start the Hugo server: `hugo server`
- Visit http://localhost:1313/ Any changes made in your project directory will be reflected in your browser

Migrate a Website to Hugo

We will make changes to the Social Justice Repair Kit site so that it can be managed using Hugo.

- Place any JavaScript, images, and any other non-CSS assets in the `static` directory
- In your `layouts/partials/` directory create templates that will be commonly used by other pages, for example, `header.html` and `footer.html` These will be referenced by other files using the `partial` keyword
- Create a `layouts/_default/baseof.html` template that contains general markup that will be used by every page
- Create a `layouts/_default/list.html` template which will be used by every section's index page in your site
- Create a `layouts/index.html` template that will be used by the home page
- Optionally create additional subdirectories in the layouts directory for any other section pages
- Once the layout templates are in place you can start adding markdown files in your `content` directory which will make up the majority of site maintenance work

It would be ideal if the majority of user managed content is authored solely using markdown with a minimal amount of shortcodes. One way of accomplishing this is by splitting up markdown files in the `content` directory, including a type property in the front matter, and then referencing that type property in layout templates where most of the markup can live.

Further Exploration

So far the Inclusive Cities and Social Justice Repair Kit projects have used Hugo to maintain content and only a subset of its features have been tested. As we use it for other projects we will most likely need to spend time trying out the following features:

- Hugo's multilingual support – it looks like translations live along side files for accompanying languages
- Search – Algolia provides free accounts to open source projects