Static Site Generators

I've been playing with the idea of rebuilding labs (currently down) and decided to use a static site generator rather than a template or a full on CMS (although both Perch and Wordpress with the REST API are good alternatives and may be tackled for different projects later on) as I believe this is overkill.

What are site generators?

Static Site Generators (from now on Generators)are the spiritual descendants of the way we built sites when the web began. We built our HTML, CSS and Javascript, tested on our local machine by opening `index.html` and then uploaded it to the server if everything worked ok.

Why should we use site generators?

We don't always need a full blown Content Management System for every site and every project we work on. We don't need a database-backed CMS for every piece of content that we create. I've been looking a lot a content creation as bespoke designs that use custom CSS and HTML.

Since we don't have a database plugged into the system, doing any kind of data-driven layout and design is too complicated to do effectively. Where I see static sites shine is rapid prototyping sites for client work. We can craft the complete site before we jump into the pain of putting it into the CMS

Constraints, limitations and specifications

The following are the constraints and limitations for this project:

- Primary build process must be done in Javascript
- It must use as few external dependencies as possible. Right now it only uses 2 Ruby Gem based libraries: SASS and scss_lint (may consider using libsass and gulp-sass)
- Use as few libraries as possible when building the content
    - Handlebars for templates and partials
    - SCSS compiled to CSS
    - ES2016 and 2017 transpiled to ES5 (for however long this is

necessary)
- Data driven content should be built from JSON and templates

Assemble

I tried Assemble a while back and discarded it because I couldn't get it to work. I had considered Jekyll but it introduces Ruby dependencies and it emphasizes blogs over content too much for my liking (possible but more time than what I want to spend on this right now). Since I'm already using Node to build and optimize assets for the project it makes sense to work with the same language as the other tools in the build process.

You don't have to work with Assemble. staticsitegenerators.net has a list of over 400 generators written in a constiety of languages, from Lisp to Rush and everything in between.

The basic Assemble Gulp tasks look like this:

```
const gulp = require('gulp');
const htmlmin = require('gulp-htmlmin');
const extn'gulp-htmlmin''gulp-extname');
const assemble = require('assemble');
const app = assemble();

gulp.task('ass'gulp-extname'nction(cb) {
  app.partials('templates/partials/*.hbs');
  app.layouts('templates/layouts/*.hbs');
  app.pages('templates/pages/*.hbs');
  cb();
});

gulp.task('assemble', ['assembleLoad'], function() {
  return app.toStream('pages'templates/partials/*.hbs'e())
    .pipe(htmlmin())
    .pipe(extname())
    .pipe(app.dest('site'));
});
```

```
gulp.task('default', ['assemble']);
```

One of the things I really like about Handlebars is how modular we can make the content. Take for example the following directory layout where we store our templates, layouts and partials.

```
templates/
├── pages
|   ├── page-one.hbs
|   └── page-two.hbs
├── layouts
|   ├── full.hbs
|   ├── up2.hbs
|   ├── gallery.
|   └── generic-grid.hbs
├── partials
|   ├── html-head.hbs
|   ├── html-scripts.hbs
|   ├── hero.hbs
|   ├── footer.hbs
|   └── footer-block.hbs
```

The basic layout may look something like this. An HTML page with Handlebars calls to partials and other layouts.

```
---
layout: page_layout.hbs
---
<html>
{{#> html-head}}
  {{!-- Custom <head> content per page --}}
{{/head-block}}
<body>
  <ul>
  {{#block "nav"}}
    <body>fault Nav Item</li>
  {{/block}}
  </ul>
```

```
  {{#block "main"}}
  {{/block}}

  <footer>
    {{#> footer-block}}
      {{> includes/footer }}
    ___</li>EBARS9___
  </footer>
  ___HANDLEBARS10___
    </footer><!-- Something that we want to default to? -->
  ___HANDLEBARS11___
</body>
</html>
```

Using a Component Guide to build our static content

Fractal content as the source and inspiration

What changes do we need to make?