Pattern Libraries

If you work on front end development it won't be long before you start hearing about pattern libraries. In this article I want to explore Pattern Libraries, what they are, what they are useful for and how they work.

In the how they work section I'll take two aspects: Using libraries to build a Pattern Library from Scratch: Fractal and Pattern Lab; the later uses the Atomic Design methodology.

What they are

Before we can delve into pattern and design libraries we need to design what they are. So here we go:

> A design library completely covers an element or a pattern: what it intended purpose is, how it looks, what behaviors and styles are available for the element and how we use it.

The patterns we build include markup, style and any default scripted behavior that we want to have for the component. If we need to change any aspect of the pattern we should create a custom pattern for the element.

What they are useful for?

The idea is that wherever we use an element from the library we know what it'll look like and how it will behave. This allows designer to create consistent interfaces for their sites and applications.

When working in a team Pattern Libraries allow for consistent work across the team. All developers will create the same elements using the patterns in the library.

Starting from Scratch (sort of)

We will use the BEM Methodology as our naming convention. I don't usually follow a naming convention for my CSS but it's agood practice to get into and it'll help

identify the different elements in CSS once the application has been built.

Most Pattern libraries use a template engine to create the components. Fractal uses Handlebars and Patternlab uses Mustache. We'll explore the implications of these choices as we create components with each library and again when we look at HTML templates.

Fractal

Working with Fractal libraries we make the following assumptions

- Node version 4.7.7 or later is already installed on your system, if it isn't you can get the latest LTS supported version for your platform from nodejs.org
- Gulp 4 is already installed globally. While it's still Alpha software it works well with existing packages. If you have an earlier version installed follow the instructions from How to install Gulp 4 before it's officially released

Setting up

For this example we'll setup the following items:

- An empty directory
- An empty Node project

```
mkdir pattern-library
cd pattern-library
npm init
```

Optionally create an empty Git project to make it easier to push to Github later.

```
git init
```

Clone the repository into the directory we created in the first step.

```
git clone  https://github.com/24ways/frontend.git .
```

The period at the end of the `git clone` command is important. It'll clone the repository into the current directory. If you skip it it will create a new directory

inside `pattern-library` and we don't want that.

Next we install Fractal and the Fractal CLI.

```
# THis will install fractal as a dependency of your project
npm install --save @frctl/fractal
# This will install  fractal CLI globally
npm i -g @frctl/fractal
```

Install the project dependencies

This will install the dependencies listed in

```
npm install
```

Commands to remember

Development

When developing components, you may want assets automatically compiled and the browser to refresh automatically. To do this, run the following task:

```
npm run dev
```

Creating a static build

To create a static instance of this project, run the following task:

```
npm run build
```

This will create a folder called www, into which the required files will be created.

Deployment

To make this project publicly accessible, you can deploy a static instance by running the following task:

```
npm run publish
```

This will publish the contents of public to your gh-pages branch.

Start your engine (server)

Start the development environment: `npm run dev` to run a development server that automatically reloads when content changes.

Open your browser and visit http://localhost:3000 to see the default pattern library.

Building components

At the simplest level a pattern is a combination of HTML, CSS, Javascript and Handlebars. The presentation layer is built with HTML and Handlebars, like the example below, taken from 24 Ways bits:

```
<footer class="c-contentinfo">
 <p class="c-contentinfo__social">
   <a href="{{ site.feed }}" rel="alternate">Grab our RSS feed</a>
   <a href="https://twitter.com/{{ site.handle }}" rel="me">Follow us on
   <a href="https://github.com/{{ site.handle }}" rel="me">Contribute on
 </p>
 <p class="c-contentinfo__copyright">
   <small>&#169; 2005-2016 24 ways and our authors</small>
 </p>
</footer>
```

Atomic Design With Pattern Lab Node

Customizing an existing Pattern Library

Bits is 24 Ways pattern library. I've always loved the navigation and how

Future enhacements: HTML Templates and Shadow DOM

Links and Resources

General Resources

- http://alistapart.com/blog/post/getting-started-with-pattern-libraries
- http://clearleft.com/thinks/165
- Pattern Primer
- https://24ways.org/2011/front-end-style-guides/
- Pattern Libraries: What They Are and Why You Need One
- Responsive Deliverables

Tools

- Handlebars

Atomic Design

- Atomic Design
- Atomic Design Methodology

Pattern Libraries

- 24 Ways
- A List Apart
- Mailchimp
- Code for America
- USPTO
- Yahoo
- IBM Design Language
- Microsoft