Scalable Front-End Architecture using SaSS

Common SaSS Mistakes

- Over use of nested selectors.
- Repetitive includes of common patterns and base module styles.
- Overuse of Mixins can become problematic too.

Say NO to nesting.

```
.module {
  .hd {}
  .bd {
    ul {
      li {
      }
    }
  }
}
```

```
.module {}
.module-hd {}
.module-bd {}
.module-list {}
.module-item {}
```

Say NO to nesting.

this doesn't allow for flexibility of your application, and could also cause slow performance on the client.

.module {}

.module .header {}

.module .bd {}

.module .bd ul {}

.module .bd ul li {}

Use the DRY Principle

```
@mixin inline-block {
   display: block;
   *display: inline;
   *zoom: 1;
}
.module-a {
   @include inline-block;
}
.module-b { … }
----------------------
.module-a { … }
.module-b { … }
```

Use the DRY Principle

If you need to have access to a base class in your markup then use a class to extend from otherwise use a placeholder selector to cut down on excess css.

@mixin inline-block { ...}

```
%ibf { @include inline-block; }
.my-module-a { @extends %ibf; }
.my-module-b { ... }
.my-module-c { ... }
```

this ends up compiling down to this.

```
.my-module-a,
.my-module-b,
.my-module-c { ... }
```

SCSS Directory Structure

- layout/ -- collections of "modules"
- module/ -- self-contained "modules"
- _base.scss -- structure of your sites layout
- _helpers.scss -- common utilities used for building a layout
- _state.scss -- global, layout, or module state changes
- _theme.scss -- visual theme of your layout
- style.scss -- inherits everything above.

Thinking about layouts

- Structures the way modules are laid out on a page
- No style should be applied directly on a layout

Thinking about modules

- Structures the way content is laid out on a page
- Style should be applied to modules.
- if your module name is .test you should prefix every child class of .test EXAMPLE: .test-header
- if you need to namespace your modules you can make them into mixins
- use placeholder selectors for base modules that are commonly used.

Thinking about _base.scss

- used to style the structure of your layout.
- things that are considered structure: font, box, floats.

Thinking about _helpers.scss

- common patterns used to help you with the structure of your layout.
- things you would consider making into helpers: text-alignment, floats, vertical-alignment, etc.
- use placeholder selectors if you don't plan to use them in your markup.

Thinking about _state.scss

- anything that modifies the original state of any part of your layout.
- you can prefix your sites layout (html, body, main content layer) with a class of something like .is-active or .is-valid.

Thinking about _theme.scss

- Anything that directly correlates to the visual style of your site.
- colors, transitions, etc.

Thinking about style.scss

- Where your all of your includes of your modules, layouts, helpers, etc get inherited
- Optionally include a css reset file here.

Tools of the trade

- Zen Coding
- Sublime Text 2 (it's awesome)
  - Package Manager, Python Interpreter, TextMate Bundle Support
- Photoshop Extensions
  - CSS Hat
  - Guide Guide
- Dwarf - On Screen Rulers / Guides

It's time for the code!