SASS

Overview

SASS (Syntactically Awesome Style Sheets) allows for features which don’t exist in CSS such as variables and mixins to be used and then compiled into CSS files. SASS uses two syntaxes, .scss and .sass, scss is the most common and is what will be shown here.

Variables

Objects to store CSS information which can be reused throughout the stylesheet. SASS uses the $ symbol to make a variable such as:

$font-stack: Helvetica, sans-serif;

$primary-color: #333;

body {

font: 100% $font-stack;

color: $primary-color;

}

Nesting

Similar to HTML, SASS allows for clear nesting in CSS. Overly nested CSS can prove hard to maintain so it should be kept to a minimum:

nav {

ul {

margin: 0;

padding: 0;

list-style: none;

}

li { display: inline-block; }

a {

display: block;

padding: 6px 12px;

text-decoration: none;

}

}

Modules

SASS can be written in multiple files and imported as modules using the @use rule:

// \_base.scss

$font-stack: Helvetica, sans-serif;

$primary-color: #333;

body {

font: 100% $font-stack;

color: $primary-color;

}

// styles.scss

@use 'base';

.inverse {

background-color: base.$primary-color;

color: white;

}

Mixins

Mixins let you make groups of CSS declarations which can be resused throughout the application. To make a mixin, use the @mixin rule and give the mixin a name, it can then be used as a CSS declaration starting with an @include:

@mixin transform($property) {

-webkit-transform: $property;

-ms-transform: $property;

transform: $property;

}

.box { @include transform(rotate(30deg)); }

Extend/Inheritance

@extend allow sharing of a set of CSS from one selector to another. Extends can be used with both normal selectors and also placeholders which start with a % symbol and are only printed if extended into a class.

/\* This CSS will print because %message-shared is extended. \*/

%message-shared {

border: 1px solid #ccc;

padding: 10px;

color: #333;

}

.message {

@extend %message-shared;

}

.success {

@extend %message-shared;

border-color: green;

}

.error {

@extend %message-shared;

border-color: red;

}

Operators

SASS allows for simple mathematical operators to be used in the file:

.container {

width: 100%;

}

article[role="main"] {

float: left;

width: 600px / 960px \* 100%;

}

aside[role="complementary"] {

float: right;

width: 300px / 960px \* 100%;

}

Maps

SASS maps allow for keys to be assoicated with values, similar to a ruby hash:

$breakpoints: (

'small' : 767px,

'medium' : 992px,

'large' : 1200px

);

Responsive Design

Using maps and a mixin to define screen size in one place in the application, then use strings to define specific

SASS File Structure

Work in progress:

stylesheets/

- application.scss (import everything global)

partials/

- \_base.scss (responsive screen sizes, grids, mixins etc)

- \_buttons.scss (buttons)

- \_typography.scss (fonts)

-\_normalize.scss (reset css to standard state)

-homepage\_style.scss (specific style for homepage)

others/ (specific style for other sections of the site)