CSS & SASS & SCSS & LESS

Css – stands for **Cascading Style Sheets**

Sass – stands for **Syntactically Awesome Style Sheets**

Scss – stands for **Sassy Cascading Style Sheets**

Less – stands for **Leaner Style Sheets**

-------------------------------------------------------------------------------------------------------------

**Sass** is the most mature, stable, and powerful professional grade CSS extension language in the world.

**SCSS** is fully compatible with the syntax of CSS, while still supporting the full power of Sass. Scss is an extension of the syntax of CSS.

Use \_OurStyleSheetName. Why because \_ sign will says that we are not including this file in index.html page. We are include in some other css file.

SCSS – contains

---------------------------------

* Variables -- $
* Nesting
* Partials -- @import
* Extend/Inheritance -- @extend
* Mixins -- @mixin and @include
* Operations

-------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------

Variables

Think of variables as a way to store information that you want to reuse throughout your stylesheet. You can store things like colors, font stacks, or any CSS value you think you'll want to reuse. Sass uses the $ symbol to make something a variable.

In style.scss file.

$var-color: red;

$var-padding: 15px 15px;

$var-margin: 10px 10px;

.para{

color: $var-color;

padding: $var-padding;

margin: $var-margin;

}

-------------------------------------------------------------------------------------------------------------

Nesting

nav {

ul {

margin: 0;

padding: 0;

list-style: none;

}

li { display: inline-block; }

a {

display: block;

padding: 6px 12px;

text-decoration: none;

} }

----------------------------------------------------------------------------------------------------

Partials

You can create partial Sass files that contain little snippets of CSS that you can include in other Sass files. This is a great way to modularize your CSS and help keep things easier to maintain. A partial is simply a Sass file named with a leading underscore. You might name it something like \_partial.scss. The underscore lets Sass know that the file is only a partial file and that it should not be generated into a CSS file. Sass partials are used with the @import directive.

Import

CSS has an import option that lets you split your CSS into smaller, more maintainable portions. The only drawback is that each time you use @import in CSS it creates another HTTP request. Sass builds on top of the current CSS @import but instead of requiring an HTTP request, Sass will take the file that you want to import and combine it with the file you're importing into so you can serve a single CSS file to the web browser.

// \_reset.scss

html,

body,

ul,

ol {

margin: 0;

padding: 0;

}

// base.scss

@import 'reset';

body {

font: 100% Helvetica, sans-serif;

background-color: #efefef;

}

Extend/Inheritance

This is one of the most useful features of Sass. Using @extend lets you share a set of CSS properties from one selector to another. It helps keep your Sass very DRY.

.panel{

Border: 3px solid #ccc;

Background: #ccc;

Color: #fff;

Margin: 10px;

}

.little-panel{

@extend .panel;

Font-size: 16px;

}

.big-panel{

@extend .panel;

Font-size: 20px;

}

-------------------------------------------------------------------------------------------------------------------------------

Mixins – Include

Some things in CSS are a bit tedious to write, especially with CSS3 and the many vendor prefixes that exist. A mixin lets you make groups of CSS declarations that you want to reuse throughout your site. You can even pass in values to make your mixin more flexible. A good use of a mixin is for vendor prefixes.

**Border-radius**

@mixin border-radius($radius){

-webkit-border-radius: $radius;

-moz-border-radius: $radius;

-ms-border-radius: $radius;

border-radius: $radius;

}

.little-panel{

@extend .panel;

@include border-radius(5px);

}

.big-panel{

@extend .panel;

@include border-radius(25px);

}

-------------------------------------------------------------------------------------------------------------------------------

Operations

Doing math in your CSS is very helpful. Sass has a handful of standard math operators like +, -, \*, /, and %.

.button {

Background: #ccc;

Color: #fff;

Display: inline-block;

Padding: 10px;

}

.jumbo-button{

@extend .button;

Padding: 10px + 30;

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

}

.light-button{

@extend .button;

Padding: 10px + 10;

}