**What is scss?**

Scss is the shortest way to write css. Writing lot of css can be headec that’s why scss make it easy. By using variables and calculations . ScSS manipulates CSS using variables, mixins, inheritance and nesting rules . Given the extensions .scss respectively, it’s translated to well-formatted CSS.

**CSS Drawbacks**

No way to re-use common style rules.

No way to specify variables that can be defined and re-used all through the style sheet.

You can’t execute computations where you can add numerical values to elements.

**Variables**

By mentioning the variables we can reuse lot of the thing in our style sheet Like colors, font stacks, or any CSS value you think you'll want to reuse. Not only reuse but in case of design change it helps a lot for consistent change in entire site. Sass uses the “$” symbol to make something a variable.

For example

$font-stack: Helvetica, sans-serif;

$primary-color: #333;

body {

font: 100% $font-stack;

color: $primary-color;

}

When the Sass is processed, it takes the variables we define for

body {

font: 100% Helvetica,

sans-serif;

color: #333;

}

**Nesting**

Scss allow you nest your CSS selectors in a way that follows the same visual hierarchy of your HTML.

For example

nav {

ul {

margin: 0;

padding: 0;

list-style: none;

}

li { display: inline-block; }

a {

display: block;

padding: 6px 12px;

text-decoration: none;

}

}

we'll notice that the ul, li, and a selectors are nested inside the nav selector. This is a great way to organize your CSS and make it more readable. When you generate the CSS you'll get something like this:

nav ul {

margin: 0;

padding: 0;

list-style: none;

}

nav li {

display: inline-block;

}

nav a {

display: block;

padding: 6px 12px;

text-decoration: none;

}

**Mixins**

We use the mixins for group declarations, for browser compatibility especially with CSS3

For example

@mixin border-radius($radius) {

-webkit-border-radius: $radius;

-moz-border-radius: $radius;

-ms-border-radius: $radius;

border-radius: $radius;

}

.box { @include border-radius(10px); }

When you generate the CSS you'll get something like this:

.box {

-webkit-border-radius: 10px;

-moz-border-radius: 10px;

-ms-border-radius: 10px;

border-radius: 10px;

}

**Extend/Inheritance**

This is one of the most useful features of Scss by Using @extend add a set of CSS properties from one selector to another.

For example

.message {

border: 1px solid #ccc;

padding: 10px;

color: #333;

}

.success {

@extend .message;

border-color: green;

}

.error {

@extend .message;

border-color: red;

}

.warning {

@extend .message;

border-color: yellow;

}

When you generate the CSS you'll get something like this:

.message, .success, .error, .warning {

border: 1px solid #cccccc;

padding: 10px;

color: #333;

}

.success {

border-color: green;

}

.error {

border-color: red;

}

.warning {

Border-color: yellow;

}

Operators

Doing math in your CSS is very helpful. Scss has a standard math operators like +, -, \*, /, and %. In our example we're going to do some simple math to calculate widths for an aside & article

.container { width: 100%; }

article[role="main"] {

float: left;

width: calc(80% - 20px);

margin-left: 20px;

}

aside[role="complimentary"] {

float: left;

width: 20%;

}

for more details please follow the links below :-

http://sass-lang.com/guide