Learning Sass

Introduction

Sass, or "Syntactically Awesome StyleSheets", is a language extension of CSS. It adds features that aren't available using basic CSS syntax. Sass makes it easier for developers to simplify and maintain the style sheets for their projects.

It adds feature like variables, functions (mixins in case of Sass), conditionals, loops, inheritance, operators.

Variables

variables in sass are declared using prefixing "$" and then the variable name

```
//syntax : $variable_name : YOUR_VALUE;
//example :
<style type='text/scss'>
$text-color : red;
p{
    color : $text-color; //value stored will be applied, here red will be the
text color
}
</style>
```

Nesting css

```
//we can nest elements like so,
.body {
    h1{
        color : blue;
    }
    p{
        color : green;
    }
}

//for better code organization
```

Reusable CSS with mixins

mixins are like functions in any other language,
a reusable block can be made where you can store your properties.
prefix your functional block name by

```
/*syntax :
@mixin block_name(arguments){
```

```
    YOUR_PROPERTIES
}
*/

//example :
<style type='text/scss'>
@mixin border-radius ($radius){
    border-radius : $radius;
}
//to include or use the mixin block "@include BLOCK_NAME"
h1{
    @include border-radius(15px/*this is parameter passed*/);
}
</style>
```

Adding logic to your sass

you can add a logical expression to your css, like if_else , else_if.

```
// @if directive is used to give if cases to
<style type='text/scss'>
@mixin div-color($color) {
    @if color == red{
        background-color : red;
    }
    @else if color == blue {
        background-color : blue;
    }
    @else if color == green {
        background-color : green;
    }
    @else{
        background-color : transparent;
    }

    div {
        @include div-color(blue); //div background color will be blue
    }
}
</style>
```

Using @for to create a loop in sass

@for is used to add looping in sass,
it is used in two ways

- start through end
- start to end

The main difference is that the "start to end" excludes the end number as part of the count, and "start through end" includes the end number as part of the count.

```
<style type='text/scss'>
@for $i from 1 through 5 {
    .text-#{$i}{ // #{$i} part is the syntax to combine a variable (i)
        font-size : 15px; //font size of 15px will be applied to mathched class
    }
}
</style>

<p class="text-1">Hello</p>
<p class="text-2">Hello</p>
<p class="text-3">Hello</p>
<p class="text-4">Hello</p>
<p class="text-5">Hello</p>
```

@each to Map Over Items in a List

`@each` directive which loops over each item in a list or map. On each iteration, the variable gets assigned to the current value from the list or map.

```
<style type='text/scss'>

$colors : (color1 : blue, color2 : black, color3 : red);

@each $key, $color in $colors{
  .#{$color}-bg{
    background-color: $color;
  }
}
  div {
    height: 200px;
    width: 200px;
  }
</style>

<div class="blue-bg"></div>
<div class="black-bg"></div>
<div class="red-bg"></div>
```

@while loop

similar functionality to the JavaScript while loop.

can be used instead of @for

```
<style type='text/scss'>
$x : 1;
@while $x <= 5{
    .text-#{$x}{
        font-size : $x * 15px; //font size will be set of each P tag
        $x : $x + 1;
    }
}
```

```
</style>

<p class="text-1">Hello</p>
<p class="text-2">Hello</p>
<p class="text-3">Hello</p>
<p class="text-4">Hello</p>
<p class="text-5">Hello</p>
```

Partials - split your css for better organization

partial file should start with underscore(' _ '). which tells Sass it is a small segment and doesn't convert it into css file

to bring the code into your main file that will be converted to css. use @import directive.

```
/* _header.scss file */

.header{
    font-size : 10px;
    color : red;
    font-weight : bold;
}

/* style.css file*/
@import 'header' //no need to put underscore, sass will understand it is a
partial
```

Inheritance in Sass

just like JavaScript, Sass has a "extend" keyword which lets you inherit from other elements rather you can borrow some properties from any element if they are similar.

```
<style type="text/scss">
.panel {
    height : 100px;
    background-color : red;
}

.main-panel {
    @extend .panel; //all panel properties will be applied.
    width : 100px;
    font-size : 15px;
}

</style>
```

NOTE : These are basics of Sass, there is always more to learn