SASS

- Sass is a preprocessor scripting language that is interpreted or compiled into Cascading Style Sheets.
- Build on RUBY.
- Sass - Syntactically awesome stylesheets.
- Sass is a CSS pre-processor.
- Sass reduces repetition of CSS and therefore saves time.
- Difference b/w Sass and Scss is the syntax.

  - Sass does not have any synatx, it just purely uses indendation and tabs to differentiate bw functions. {When Sass was converted to css, we had to remove/add all the syntax in the css, this caused a lot of problem, so Scss was developed to overcome this issue.}

  - Scss uses syntax same as the css uses. We can call this as the superset of CSS.

- If you are using ATOM - sass --watch foldername (ATOM) - terminal code.
- If you are using VScode - option is there as Watch Sass (Bottom right corner condn: Installed Liver sass compiler.)

- **Variables**

Variables are a way to store information that you can re-use later.

With Sass, you can store information in variables, like:

- strings, numbers, colors, booleans, lists, nulls

Sass uses the **$** symbol, followed by a name, to declare variables.

$variablename: value;

note: Use !global to override the varibale value.

- <u>**Nesting**</u>

Sass lets you nest CSS selectors in the same way as HTML.

Scss code ex:

```
nav {
 ul {
   margin: 0;
   padding: 0;
   list-style: none;
 }
 li {
   display: inline-block;
 }
 a {
   display: block;
   padding: 6px 12px;
   text-decoration: none;
 }
}
```

and after compilation the **CSS** code looks 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;
}
```

So instead of writing individual block, Scss helps you ease the work with the help of nesting.

- **Partials and Imports**

The partials file name should always begin with an **"_"**.

By default, Sass transpiles all the .scss files directly. However, when you want to import a file, you do not need the file to be transpiled directly.

Sass has a mechanism for this: If you start the filename with an underscore, Sass will not transpile it. Files named this way are called partials in Sass.

So, a partial Sass file is named with a leading underscore

***_filename.scss*.**

Just like CSS, Sass also supports the @import directive.

The *@import* directive allows you to include the content of one file in another.

The CSS *@import* directive has a major drawback due to performance issues; it creates an extra HTTP request each time you call it. However, the Sass *@import* directive includes the file in the CSS; so no extra HTTP call is required at runtime!

@import filename;

- <u>**Mixin and Include**</u>

The *@mixin* directive lets you create CSS code that is to be reused throughout the website.

@mixin name { property: value; property: value;}

The *@include* directive is created to let you use (include) the mixin.

selector {@include mixin-name;}

Mixins accept arguments. This way you can pass variables to a mixin.

Here is how to define a mixin with arguments:

@mixin name($width, $color) { border: $width solid $color;}

It is also possible to define default values for mixin variables:

@mixin name($width: 1px, $color: red) { border: $width solid $color;}

- **Extend and Inheritance**

The *@extend* directive lets you share a set of CSS properties from one selector to another.

The *@extend* directive is useful if you have almost identically styled elements that only differ in some small details.

@extend .classname;

- **Strings**

- **quote(string)** - Adds quotes to string, and returns the result.

- **str-index(string, substring)** - Returns the index of the first occurrence of the substring within string.

- **str-insert(string, insert, index) -** Returns string with insert inserted at the specified index position.

- **str-length(string)** - Returns the length of string (in characters).

- **str-slice(string, start, end)** - Extracts characters from string; start at start and end at end, and returns the slice.

- **to-lower-case(string)** - Returns a copy of string converted to lower case.

- **to-upper-case(string)** - Returns a copy of string converted to upper case.

- **unique-id()** - Returns a unique randomly generated unquoted string (guaranteed to be unique within the current sass session).

- **unquote(string)** - Removes quotes around string (if any), and returns the result.

- **Numeric**

- **abs(number)** - Returns the absolute value of number.

- **ceil(number)** - Rounds number up to the nearest integer.

- **comparable(num1, num2)** - Returns whether num1 and num2 are comparable.

- **floor(number)** - Rounds number down to the nearest integer.

- **max(number...)** - Returns the highest value of several numbers.

- **min(number...)** - Returns the lowest value of several numbers.

- **percentage(number)** - Converts number to a percentage (multiplies the number with 100).

- **random()** - Returns a random number between 0 and 1.

- **random(number)** - Returns a random integer between 1 and number.

- **round(number)** - Rounds number to the nearest integer.

- **Map**

note: Sass maps are immutable (they cannot change). So, the map functions that return a map, will return a new map, and not change the original map.

- **map-get(map, key)** - Returns the value for the specified key in the map.

- **map-has-key(map, key)** - Checks whether map has the specified key. Returns true or false.

- **map-keys(map)** - Returns a list of all keys in map.

- **map-merge(map1, map2)** - Appends map2 to the end of map1.

- **map-remove(map, keys...)** - Removes the specified keys from map.

- **map-values(map)** - Returns a list of all values in map.

- **Color**

- **rgb(red, green, blue)** - Sets a color using the Red-Green-Blue (RGB) model. An RGB color value is specified with: rgb(red, green, blue). Each parameter defines the intensity of that color and can be an integer between 0 and 255, or a percentage value (from 0% to 100%).

- **rgba(red, green, blue, alpha)** - Sets a color using the Red-Green-Blue-Alpha (RGBA) model. RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity of the color. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

- **hsl(hue, saturation, lightness)** - Sets a color using the Hue-Saturation-Lightness (HSL) model - and represents a cylindrical-coordinate representation of colors. Hue is a degree on the color wheel (from 0 to 360) - 0 or 360 is red, 120 is green, 240 is blue. Saturation is a percentage value; 0% means a shade of gray and 100% is the full color. Lightness is also a percentage; 0% is black, 100% is white.

- **hsla(hue, saturation, lightness, alpha)** - Sets a color using the Hue-Saturation-Lightness-Alpha (HSLA) model. HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity of the color. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

- **grayscale(color)** - Sets a gray color with the same lightness as color.

complement(color) - Sets a color that is the complementary color of color.

- **invert(color, weight)** - Sets a color that is the inverse or negative color of color. The weight parameter is optional and must be a number between 0% and 100%. Default is 100%.

Sass Get Color Functions

- **red(color)** - Returns the red value of color as a number between 0 and 255.

- **green(color)** - Returns the green value of color as a number between 0 and 255.

- **blue(color)** - Returns the blue value of color as a number between 0 and 255.

- **hue(color)** - Returns the hue of color as a number between 0deg and 255deg.

- **saturation(color)** - Returns the HSL saturation of color as a number between 0% and 100%.

- **lightness(color)** - Returns the HSL lightness of color as a number between 0% and 100%.

- **alpha(color)** - Returns the alpha channel of color as a number between 0 and 1.

- **opacity(color)** - Returns the alpha channel of color as a number between 0 and 1.

Sass Manipulate Color Functions

- **mix(color1, color2, weight)** - Creates a color that is a mix of color1 and color2. The weight parameter must be between 0% and 100%. A larger weight means that more of color1 should be used. A smaller weight means that more of color2 should be used. Default is 50%.

- **adjust-hue(color, degrees)** - Adjusts the color's hue with a degree from -360deg to 360deg.

- **adjust-color(color, red, green, blue, hue, saturation, lightness, alpha)** Adjusts one or more parameters by the specified amount. This function adds or subtracts the specified amount to/from the existing color value.

- **change-color(color, red, green, blue, hue, saturation, lightness, alpha) -** Sets one or more parameters of a color to new values.

- **scale-color(color, red, green, blue, saturation, lightness, alpha)** - Scales one or more parameters of color.

- **rgba(color, alpha)** - Creates a new color of color with the given alpha channel.

- **lighten(color, amount)** - Creates a lighter color of color with an amount between 0% and 100%. The amount parameter increases the HSL lightness by that percent.

- **darken(color, amount)** - Creates a darker color of color with an amount between 0% and 100%. The amount parameter decreases the HSL lightness by that percent.

- **saturate(color, amount)** - Creates a more saturated color of color with an amount between 0% and 100%. The amount parameter increases the HSL saturation by that percent.

- **desaturate(color, amount)** - Creates a less saturated color of color with an amount between 0% and 100%. The amount parameter decreases the HSL saturation by that percent.

- **opacify(color, amount)** - Creates a more opaque color of color with an amount between 0 and 1. The amount parameter increases the alpha channel by that amount.

- **fade-in(color, amount)** - Creates a more opaque color of color with an amount between 0 and 1. The amount parameter increases the alpha channel by that amount.

- **transparentize(color, amount)** - Creates a more transparent color of color with an amount between 0 and 1. The amount parameter decreases the alpha channel by that amount.

- **fade-out(color, amount)** - Creates a more transparent color of color with an amount between 0 and 1. The amount parameter decreases the alpha channel by that amount.

- **IF ELSEIF ELSE statement**

@if $id condition { .... }

@elseif $id condition { .... }

@else { .... }
