INTRODUCTION TO SASS

Syntactically Awesome Style Sheets (SASS)

What is SASS?

- SASS is a popular preprocessor to CSS
- Sass allows the use of the concept of DRY (don't repeat yourself)
- LESS is another preprocessor competitor to SASS (you would only need to learn one CSS preprocessor language).

Syllabus:

- Introduction
- SASS vs SCSS vs CSS
- Setup (Mac)
- SASS Hello World
- Sass Concepts
- Comments
- SASS Variables & CSS Variables
- Nesting
- Partials & Imports
- Mix-ins and Inheritance (@extend)
- Functions
- Conditions and Loops

Useful Links:

- http://sass-lang.com/
- https://www.youtube.com/watch?v=S4mPsoZ7sG4&index=3&list=PLUoqTnNH-2XxOt7UsKITqbfrA2ucGosCR (Brad Hussey)
- https://www.youtube.com/watch?v=P1G4_zxOxtk (Help People)
- https://www.youtube.com/watch?v=St5B7hnMLjg (The net ninja)
- https://www.youtube.com/watch?v=fbVD32w1oTo&list=RDQMITWf3p81b5Q&index=1 (LevelUpTuts)

Introduction & Overview:

What is a preprocessor? A scripting language that extends CSS and gets compiled into regular CSS syntax.

A ".sass or .scss" file is a preprocessed file that will later be converted to a ".css" file that the browser can understand.

Advantages of using SASS:

- Cleaner, reusable and extendable code
- Very easy syntax to learn
- Features like mixins, variables, extends, imports etc. (CSS like a programming language)
- More efficient workflow (especially for larger projects)

We will be looking at version 3 of SASS which introduces SCSS (Sassy CSS).

SASS VS SCSS VS CSS:

- Sass is a new syntax, it has no Curley braces or semicolons and uses indentation & nesting for hierarchy.
- SCSS is like vanilla CSS i.e. it has no new syntax to learn (valid CSS == valid SCSS) however it still access to the features of SASS such as mixins, variables etc.
- CSS is the original language that requires you to type everything, it has no mixins, variables etc. and can become very messy code.

We will use SASS syntax as it is a different syntax but once we have learned it we are able to use either SASS or SCSS (SCSS is the same as SASS but uses curly braces { } and semicolons ; in its syntax just like CSS).

Setup (Mac):

Go to http://sass-lang.com/ to follow the guide on installing SASS onto your computer.

SASS requires ruby dependency but if you are using a Mac, Ruby comes pre-installed on your machine. You can installs SASS either using a Application (e.g. Koala) or through the command line. Ruby uses gems as its package manager.

Install SASS:

1. Open Terminal
2. Install Sass using the terminal command (*sudo for super user do*):

```
sudo gem install sass
```

3. Double check SASS is installed on your machine by typing the command:

```
sass -v
```

It should return Sass version number e.g: `sass 3.5.4`

You should now be ready to use SASS in your projects.

Please refer to the sass-lang.com/install page to view installation instruction for other operating systems (OS).

SASS Hello World:

In this example we will be creating our first sass file to understand how our sass file is compiled to spit out our css file for our Hello World html page. The project files can be found in the Sass Hello World Example folder.

We can write our SASS codes in any editor of your choice (we will be using CS Code/Atom as our text editor).

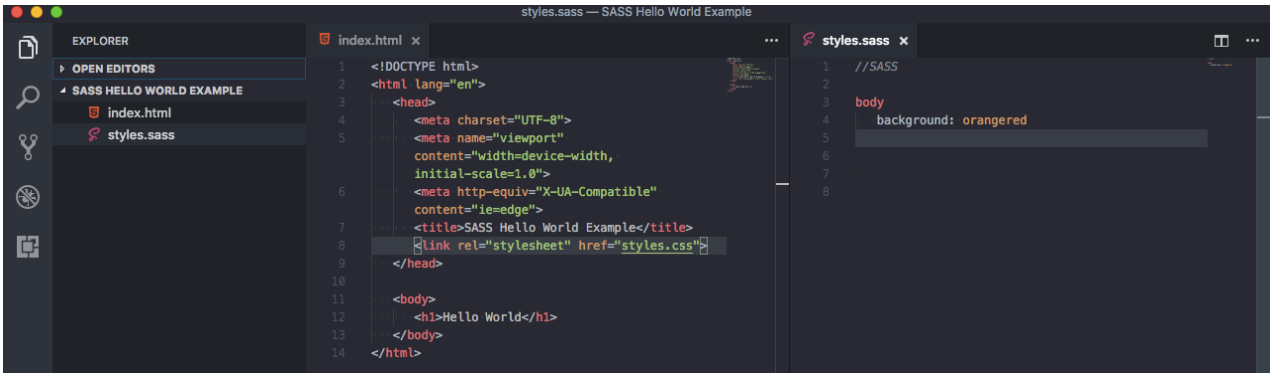
SASS require a compiler to compile the preprocessed .sass (or .scss) files into .css files. We can use either a GUI Application such as:

- Koala http://koala-app.com/ (Free)
- Codekit https://codekitapp.com/ (Paid/Free Trial (b*uy now popup every 15mins*))

or we can use the terminal as our sass compiler. In this example we will be using the terminal to compile our .sass file into .css code.

In the SASS Hello World Example folder we have two files:

- index.html (within <Body> tags we have Hello World in <h1> tags)
- styles.sass (our sass file to make the body-colour of our html page orangered)



We require Terminal (or GUI application) to compile our sass code into plain css code which the browser can read. To do this we need to open up Terminal and cd into the directory containing our project files (.sass file). VS Code has an integrated terminal (*view>integrated terminal*) and this by default cd to the folder opened within the VS Explorer menu.

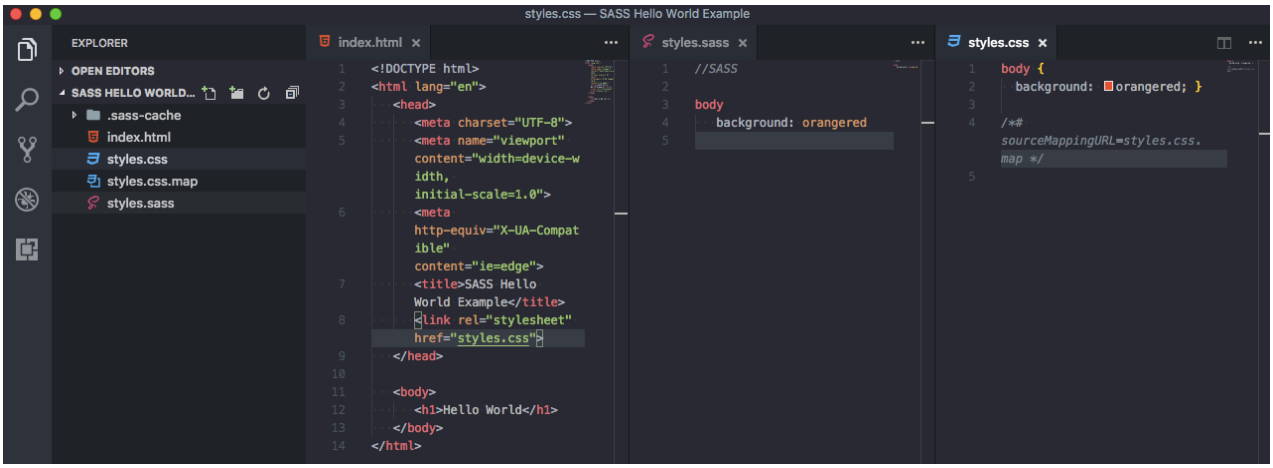
We now need to run the sass --watch command and add two arguments:

1. Which file/folder to watch and
2. Where to compile the css code.

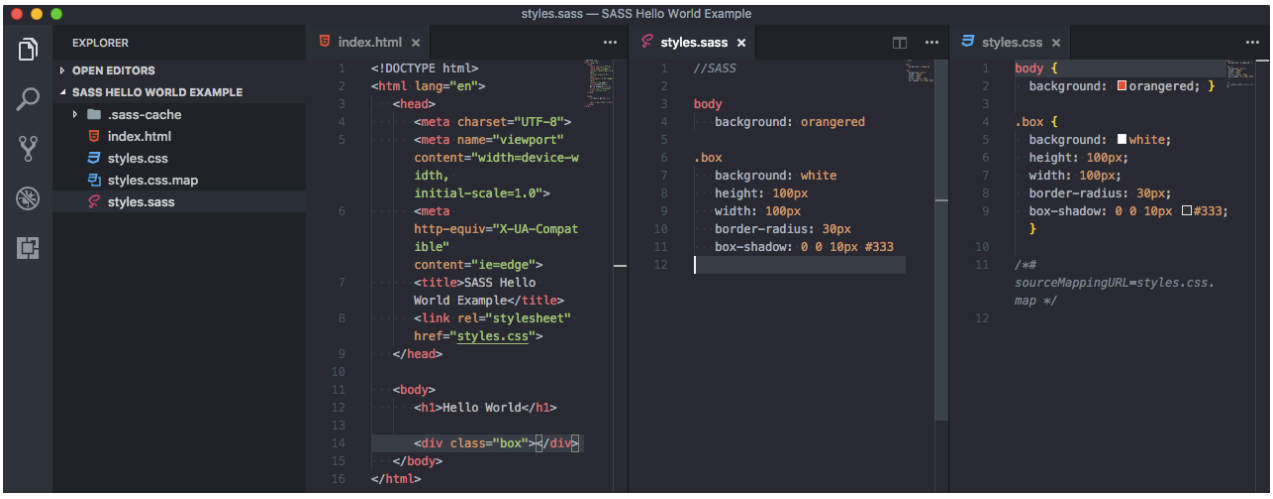
The sass command in the terminal would look something like:

```
sass --watch styles.sass:styles.css
```

This will create a styles.css file if one does not already exist and a styles.css.map (mapping file ignore but do not delete).



Leave the terminal running in the background and the sass watch command will continue to run in the background and look for any changes made to the .sass file and compile it to the .css file (i.e. every time you press save to the .sass file). In the example above we added a new box style to the .sass file and pressed save to update the .css file.



To end the sass watch command simply press Ctrl+C in the terminal.

SASS Concept:

Vanilla CSS on its own can be time consuming and while stylesheets are becoming larger and more complex, inherently they become harder to maintain. This is where a preprocessor such as SASS or LESS can help. SASS allows you to use features that are not in CSS yet like nesting, mixins, inheritance and other programming language concepts.

It must be noted that CSS is forever evolving and in the near distant future most of the features in SCSS will become part of vanilla CSS, the most recent addition to CSS being variables. Although we have variables within CSS, we can continue to use SASS for its many more useful features such as mixins and also manage/maintain our CSS codes for larger projects.

We will look at the many SASS features below and how we can utilise these SASS features/ concepts in our projects. Finally, we will look at SASS best practice for folder/file structures for our sass and css files to manage larger and modular CSS projects.

SASS Comments:

There are two methods of adding comments to your sass code:

1. SASS comments

```
//SASS Comment - this is a private one line comment
```

2. CSS Comments — add forward slash and asterisk /* */

```
/*CSS Comment - this is a public multiline comment */
```

SASS comments are single line and so requires the double forward slash every time you add a return in your code, whereas CSS comments are multi-line. SASS comments are not compiled to the .css file and so are considered to be private comments that only the developer will see whereas CSS comments are made available to the public to see within the .css file.

SASS Variables:

SASS Variables (just like in many other programming languages) are a way to store information that you can reuse throughout your stylesheets. You can store any CSS property values such as colours, font-stack etc. To declare a variable we would the $ symbol before the name of the variable — for example:

```
$background-colour: #333

$font-colour: white
```

To use the variables declared above we would write the variable name next to the CSS property — for example:

```
body

    background: $background-colour

    color: $font-colour
```

The SASS Comments & Variables folder provides a example of comments and variables.

Vanilla CSS Variables:

The syntax for creating a variable in CSS is completely different to SASS. Here is an example variable in CSS:

```
--background-colour: #333
```

To call this variable within our CSS we must type:

```
color: var(—background-colour);
```

Note: all valid CSS syntax are valid SCSS syntax - therefore we can use this syntax within our SCSS code but not within our SASS code (*SASS uses a different syntax*). SCSS also uses the $ symbol just like SASS to declare variables but have a semicolon ; at the end — for example:

```
$background-colour: #333;

$font-colour: white;
```

Nesting:

Nesting allows you to view a clear visual hierarchy of your HTML elements. CSS does not allow for nesting elements. However, using SASS we can nest elements as we would do in HTML.

Here is an example of SASS syntax for nesting a websites navigation:

SASS

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

SCSS

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

Note: the difference between SASS and SCSS syntax is that SCSS is similar to CSS i.e it adds the curly braces { } and semicolons ; to its syntax.

This will produce the CSS code:

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

Partials & Imports:

Continue…