Exercise (Instructions): Less

Objectives and Outcomes

In this exercise you will learn to write Less code and then automatically transform it into the corresponding CSS code. At the end of this exercise you will be able to:

- Write Less code using many of the features of Less
- Automatically convert the Less code into CSS

Adding Less Variables

- Open the *conFusion* project in Brackets or any other text editor of your choice. In the css folder, create a file named *mystyles.less*. We will add the Less code into this file.
- Add the following Less variables into the file:

```
@white: #ffffff;
@off-white: #eeeeee;
@lt-gray: #dddddd;
@indigo: #303f9f;
@dark-indigo: #1a237e;
@light-indigo: #7986CB;

// Height variables
@carousel-height: 300px;
```

We have just added a few color and a height variable. We will make use of these variables while defining the classes.

Less Mixins

- Next we add mixing into the file as follows:

```
.zero-margin (@pad-up-dn: 0px, @pad-left-right: 0px) {
        margin:0px auto;
```

```
        padding: @pad-up-dn @pad-left-right;
}
```

We will make use of this to define several row classes next.

- Using the Mixin class that we defined earlier, add the following row classes to the file:

```
.row-header {
        .zero-margin();
}
.row-content {
        .zero-margin (50px, 0px);
        border-bottom: 1px ridge;
        min-height:400px;
}
.row-footer {
        .zero-margin(20px, 0px);
        background-color: #AfAfAf;
}
.jumbotron {
        .zero-margin(70px, 30px);
        background:@light-indigo;
        color:floralwhite;
}
```

Note the use of the mixin with various parameters in defining the classes.

Nesting Selectors

- Next we add a carousel class to illustrate the use of nesting of classes in Less, as follows:

```
.carousel {
        background:@dark-indigo;
        .item {
```

```
                height: @carousel-height;
                img {
                        left: 0;
                        min-height: @carousel-height;
                        position: absolute;
                        top: 0;
                }
        }
}
```

- Add another group of nesting classes with the navbar-inverse class as follows:

```
.navbar-inverse {
        background: @indigo;
        color:@white;
        .navbar-nav>.active>a {
                background: @dark-indigo;
                color: @white;
                &:hover {
                        background: @dark-indigo;
                        color: @white;
                }
                &:focus {
                        background: @dark-indigo;
                        color: @white;
                }
        }
        .navbar-nav>.open>a {
                background: @dark-indigo;
                color: @white;
                &:hover {
                        background: @dark-indigo;
                        color: @white;
                }
```

```
                &:focus {
                        background: @dark-indigo;
                        color: @white;
                }
        }
        .navbar-nav {
                .open {
                        .dropdown-menu>li>a {
                                background-color: @indigo;
                                color:@off-white;
                                &:hover {
                                        color:#000000;
                                }
                        }
                        .dropdown-menu {
                                background-color: @indigo;
                                color:@off-white;
                        }
                }
        }
}
```

Add the remaining CSS classes

- Finally we add in those CSS classes which do not fall into any of the above categories. These still use the Less variables, but do not use nesting or mixins:

```
address {
        color:#0f0f0f;
        font-size:80%;
        margin:0px;
}
```

```
body {
        align:center;
        padding:50px 0px 0px 0px;
        z-index:0;
}
.tab-content {
        border-bottom: 1px solid @lt-gray;
        border-left: 1px solid @lt-gray;
        border-right: 1px solid @lt-gray;
        padding: 10px;
}
.affix {
        top:100px;
}
#carouselButtons {
        bottom: 0px;
        position: absolute;
        right:0px;
}
```

Installing and using the lessc Compiler

- Now we install the node module to support the compilation of the Less file. To do this, type the following at the command prompt:

```
    npm install -g less
```

This will install the *less* NPM module globally so that it can be used by any project. **Note: if you are executing this on a Mac or Linux machine, you may need to add "sudo" to the beginning of this command**. This will make available the *lessc* compiler for us so that we can compile Less files.

- Next, go to the CSS folder on your machine and rename the *mystyles.css* file that you have there as *mystyles-old.css*. This is to save the CSS file that we have been using so far. We will be creating a new *mystyles.css* file by compiling the Less file.

- Next type the following at the command prompt to compile the Less file into a CSS file:

```
lessc mystyles.less > mystyles.css
```

Conclusions

In this exercise you learnt to write Less code and then automatically generating the CSS file by compiling the Less code.