Exercise: Responsive Design and Bootstrap Grid System Part 1

Objectives and Outcomes

This exercise introduces you to responsive design and Bootstrap support for mobile first responsive design through the use of the grid system. At the end of this exercise, you will be able to:

- Create responsive websites using the Bootstrap grid system
- Reordering content using push, pull and offset classes

Note: In this exercise we will continue to update the *index.html* file in the *conFusion* folder that we created and edited in the previous lecture.

Bootstrap Grid System and Responsive Design

Bootstrap is designed to be mobile first, meaning that the classes are designed such that we can begin by targeting mobile device screens first and then work upwards to larger screen sizes. The starting point for this is first through media queries. We have already added the support for media queries in the last lesson, where we added this line to the head:

```
1       <meta name="viewport" content="width=device-width, initial-scale=1, shrink
          -to-fit=no">
```

The *viewport* meta tag ensures that the screen width is set to the device width and the content is rendered with this width in mind. This brings us to the second issue, designing the websites to be responsive to the size of the viewport. This is where the Bootstrap grid system comes to our aid. Bootstrap makes available four sizes, xs for extra small, sm for small, md for medium and lg for large screen sizes. We have already seen the basics of responsive design. In this exercise, we will employ the Bootstrap grid classes to design the websites. We would like our website to have the content stacked on extra small devices, but become horizontal within each row for smaller devices and beyond. Towards this goal, we will make use of the classes .col-*, .col-sm-*, col-md-*, and .col-lg-* for defining the layouts for the various device sizes. We can specify how many columns each piece of content will occupy within a row, all adding up to 12 or a multiple thereof.

Using a Container class

- We use the container class to keep content within a fixed width on the screen, determined by the size of the screen. The alternative is to use the container-fluid class to make the content automatically to span the full width of the screen. We will discuss further about this when we discuss the Bootstrap grid system in the next lecture. Add the container class to the first div right after the </header> in the file as follows.

```
<div class="container"> ...
```

Dividing the content into rows

- Let us now add the class *row* to the first-level inner *div* elements inside the container. This organizes the page into rows of content. In the next exercise, we will see how we can add other classes to the rows.

```
    <div class="row"> ...
```

Creating a Jumbotron

- Let us add the class jumbotron to the header class as shown below. This turns the header element into a Bootstrap component named Jumbotron. A jumbotron is used to showcase key content on a website. In this case we are using it to highlight the name of the restaurant.

```
        <header class="jumbotron"> ...
```

- In the header add a **container** class to the first inner div and a row class to the second inner div.

Creating a footer

- Finally, in the footer add a **container** class to the first inner div and a row class to the second inner div.

Applying column classes within each row

- In the header row, we will display the restaurant name and the description to occupy 8 columns, while we will leave four columns for displaying the restaurant logo in the future. Let us go into the jumbotron and define the classes for the inner divs as follows:

```
                <div class="col-12 col-sm-8"> ... </div>

                <div class="col col-sm"> ... </div>
```

- For the remaining three div rows that contain content, let us define the classes for the inner divs as follows:

```
<div class="col-sm-4 col-md-3"> ... </div>

<div class="col-sm col-md"> ... </div>
```

- For the footer, let us define the classes for the inner divs as follows:

```
<div class="col-5 col-sm-2"> ... </div>

<div class="col-6 col-sm-5"> ... </div>

<div class="col col-sm-4"> ... </div>

<div class="col-auto"> ... </div>
```

Now you can see how the web page has been turned into a mobile-first responsive design layout.

Using Push, Pull and Offset with column layout classes

- In the content rows, we would like to have the title and description to alternate so that it gives an interesting look to the web page. For extra small screens, the default stacked layout works best. This can be accomplished by using the .push-sm-* and .pull-sm-* for the first and the third rows as follows:

```
<div class="col-sm-4 push-sm-8 col-md-3 push-md-9"> ... </div>

<div class="col-sm pull-sm-4 col-md pull-md-3"> ... </div>

```

- For the div containing the <ul> with the site links, update the class as follows:

```
    <div class="col-5 offset-1 col-sm-2">
```

Conclusion

In this exercise, we reviewed responsive design and the Bootstrap grid system.

Mark as completed