Bootstrap introduction

bootstrap is a ux framework used to build responsive mobile first applications. the bootstrap library provide us with CSS files, JS files and we can install the library by linking the css file and the js files with our application. after you link bootstrap you can use the classes from the stylesheet and the js functions

Install Bootstrap

we will install bootstrap using bower. so after initiating bower run:
> bower install bootstrap --save

in your **index.html** file connect the bootstrap css file by putting in the **head** the link to the css.
<link rel="stylesheet" href="bower_components/bootstrap/dist/css/bootstrap.css" />

notice that if you want to include your css files they should be added after the bootstrap css so they will take precedence on the styling in bootstrap.

if you want you can also include the js files of bootstrap at the bottom of the body.
bootstrap js files depends on **jquery** and also depends on **tether. tether** is a library used to position absolute elements, and is used is some of bootstrap js components like tooltip.
if you installed **bootstrap** using **bower** you should see that **jquery** is installed as well but **tether** is not. so let's install **tether** by typing:
> bower install tether --save

and in the end of the body of our file include the **js** files in the following order:
<script src="bower_components/jquery/dist/jquery.js"></script>
<script src="bower_components/tether/dist/js/tether.js"></script>
<script src="bower_components/bootstrap/dist/js/bootstrap.js"></script>

notice that I included the files and not the minified files, it is common to include the regular files and minify all your files when you deploy, this way you don't have to worry that your minification will run on minified files.

Bootstrap is installed not let's examine what bootstrap is providing us.

Grid System

when using the grid system of bootstrap you have to include a div element with a **container** or **container-fluid** class.
container-fluid get's a width of 100% and **container** get's a **max-width** based on the size of the screen

place the following code in your html file to examine the differenced:

```
<div class="container" style="height: 100px; background-color: red;">
</div>

<div class="container-fluid" style="height: 100px; background-color: orange;">
</div>
```

inside the **containers** we are placing **rows** which mark a group of **coloumns**
inside each row you can place **columns** by typing:
col-xs-12
col-sm-7
col-md-3
col-lg-5

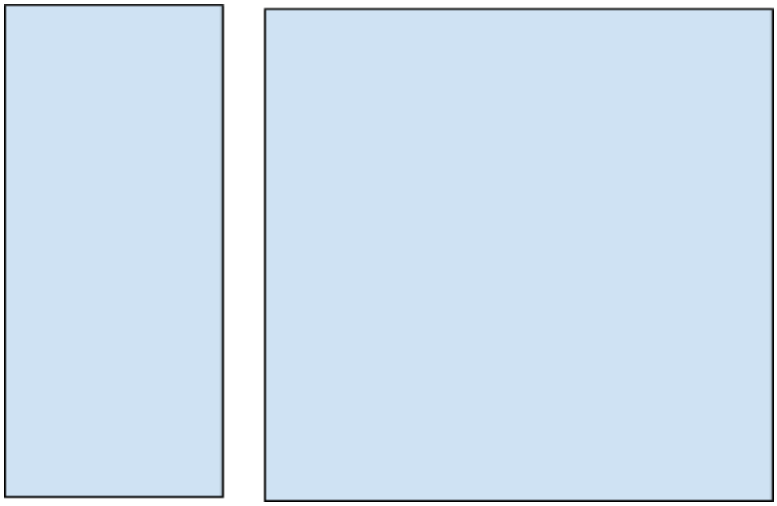
each row sums up to 12 columns.
notice that if we are above 12 columns we will drop all the extra cols to the next line
note that **col-md** will affect also large screens if there is no **col-lg**

you can also offset a col by using:
col-xs-offset-*
col-sm-offset-*
col-md-offset-*
col-lg-offset-*

the offset still needs to sum with the cols to 12 cols

for example let's say we want to a layout of left navbar and content area on md screens and on small screens the content area takes all the width.

medium screens and above



small screens

```
<div class="container" style="background-color: red;">
  <div class="row">
    <div
      class="col-sm-4 visible-sm-block visible-md-block visible-lg-block"
      style="background-color: orange;"
    >
      left side bar
    </div>
    <div
      class="col-xs-12 col-sm-8"
      style="background-color: pink;"
    >
      content area
```

```
    </div>
  </div>
</div>
```

notice that the left side bar is only visible on md, lg, small screens.

When you place another row inside a partial cols then it will create a new 12 cols grid in the partial.

for example:

```
<div class="container" style="background-color: red;">
  <div class="row">
    <div
      class="col-sm-4 visible-sm-block visible-md-block visible-lg-block"
      style="background-color: orange;"
    >
      left side bar
      <div class="row">
        <div
            class="col-xs-4"
            style="background-color: #00b3ee"
        >1</div>
        <div
            class="col-xs-4"
            style="background-color: #1b6d85"
        >2</div>
        <div class="col-xs-4"
          style="background-color: #01ff70;"
        >3</div>
      </div>
    </div>
    <div
      class="col-xs-12 col-sm-8"
      style="background-color: pink;"
    >
      content area
    </div>
  </div>
</div>
```

Student exercise

create the grid from the previous lesson with the nav, footer and three cols, this time make it so that on xs screens there is on col in the center and in sm screens there is 2 columns.

Go over the documentation of bootstrap and show them in general what is added

bootstrap also add styling to the typography so in your html file place headers.

In the Javascript show the modal and the carousel

Student exercise

This exercise will practice using bootstrap, jquery and js to Create partial TODO application. The app will contain the following sections from top to bottom.

- Create new task
    - this will be a form that we will use to create new tasks
    - the form contains the fields: Title, Description, Group, Date, Time
    - the form should display correctly on all device including mobile devices.
    - Use bootstrap to design the form, use the following classes: **container, row, col-xs-*, form-horizontal, form-group, control-label, form-control, btn, btn-primary**
    - Use the form we created in the previous lessons
    - When you publish a new Task it should be added to the task list section
- Create the task list section
    - the task list will display a list of all the tasks
    - the task list will be responsive and will display each row this number of tasks:
        - on large screens it will display 3 tasks per row
        - on small screens it will display 2 tasks per row
        - on extra small screens it will display 1 task per row
- Use bootstrap to design your responsive list use design similar to the thumbnails design: **http://getbootstrap.com/components/#thumbnails-custom-content**