

So last week, we have talked about JavaScript Events such as :

- onclick
- onchange
- oninput
- onkeyup
- onkeydown
- Functions such as:
- `addEventListener()`
- `createElement()`
- `classList.add()`
- `classList.remove()`
- `replace()`

We have created a simple imitation of the Add to Cart Functionality.

So for this lesson we will be talking about additional JavaScript functionalities and a little about Bootstrap.

Let's first start with **Bootstrap**.

## Bootstrap

What is Bootstrap?

Bootstrap is a CSS Framework used in web development. It is a free front-end framework designed to make web development faster and easier. It contains predefined classes that can be used to design and layout a website. It is also used to make your websites responsive.

How do we use bootstrap?

Do we download and install something?

The answer is no, there's no need to download anything.

The only thing we need to use Bootstrap are the following:

### Meta Name

- `<meta name="viewport" content="width=device-width, initial-scale=1">`

### CSS Link

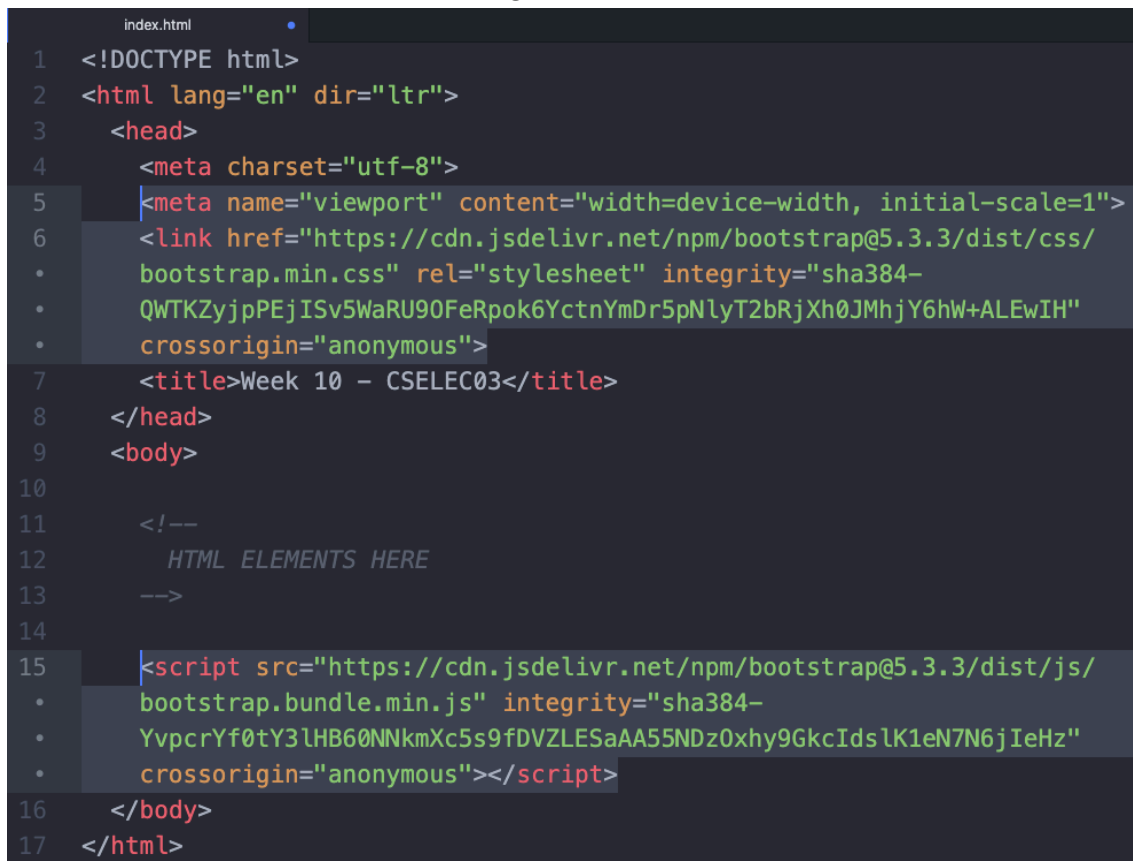
- `<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH" crossorigin="anonymous">`

### JavaScript Link

- `<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>`

But where do we place all of these? According to [Bootstrap Documentations](#), we should place the **meta tag** and **css link** inside the **head tag**, and the **script tag link** at the end of the **body tag** of our **HTML Document**. (See figure 1.0)

Figure 1.0



```
index.html
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/
  • bootstrap.min.css" rel="stylesheet" integrity="sha384-
  • QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH"
  • crossorigin="anonymous">
7   <title>Week 10 - CSELEC03</title>
8 </head>
9 <body>
10
11   <!--
12     HTML ELEMENTS HERE
13   -->
14
15   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/
  • bootstrap.bundle.min.js" integrity="sha384-
  • YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
  • crossorigin="anonymous"></script>
16 </body>
17 </html>
```

After setting this up, we can now fully use and utilize Bootstrap. But first, let us know first what are the classes we can use when using Bootstrap. Bootstrap gives us a set of predefined global classes that we can use throughout our web development (front-end).

Let's focus on those that we might be using in this lesson.  
Let's start with the basic: **Typography**

Heading Tags (<h1> to <h6>) have their own styles following a bolder font-weight and a responsive font-size.

**h1 heading**

**h2 heading**

**h3 heading**

**h4 heading**

**h5 heading**

**h6 heading**

*Figure 2.1  
No Bootstrap*

**h1 Bootstrap heading**

**h2 Bootstrap heading**

**h3 Bootstrap heading**

**h4 Bootstrap heading**

**h5 Bootstrap heading**

**h6 Bootstrap heading**

*Figure 2.2  
With Bootstrap*

Bootstrap uses a default font-size of 1rem (16px by default) and a line-height of 1.5.

Paragraph Tags (<p></p>) have a margin-top of 0rem, and a margin-bottom of 1rem (16px by default).

Aside from the Heading Tags, Bootstrap also has classes for each heading type.

.h1

.h2

.h3

.h4

.h5

.h6

These classes can be used if we want a text to be the same size and design as the heading type we want but does not need to be in a heading element.

```
<p>h1 No Bootstrap heading</p>  
<p>h2 No Bootstrap heading</p>  
<p>h3 No Bootstrap heading</p>  
<p>h4 No Bootstrap heading</p>  
<p>h5 No Bootstrap heading</p>  
<p>h6 No Bootstrap heading</p>
```

(Figure 3.1 No Bootstrap)

**h1** No Bootstrap heading

**h2** No Bootstrap heading

**h3** No Bootstrap heading

**h4** No Bootstrap heading

**h5** No Bootstrap heading

**h6** No Bootstrap heading

(Figure 3.2 No Bootstrap)

```
<p class="h1">h1 Bootstrap heading</p>
<p class="h2">h2 Bootstrap heading</p>
<p class="h3">h3 Bootstrap heading</p>
<p class="h4">h4 Bootstrap heading</p>
<p class="h5">h5 Bootstrap heading</p>
<p class="h6">h6 Bootstrap heading</p>
```

(Figure 4.1 With Bootstrap)

# h1 Bootstrap heading

## h2 Bootstrap heading

### h3 Bootstrap heading

#### h4 Bootstrap heading

##### h5 Bootstrap heading

###### h6 Bootstrap heading

(Figure 4.2 With Bootstrap)

Additionally, if we want our headings to be a little more exaggerated, we can use the display styles that Bootstrap offers.

.display-1

.display-2

.display-3

.display-4

.display-5

.display-6

```
<h1 class="display-1">Display 1</h1>
<h1 class="display-2">Display 2</h1>
<h1 class="display-3">Display 3</h1>
<h1 class="display-4">Display 4</h1>
<h1 class="display-5">Display 5</h1>
<h1 class="display-6">Display 6</h1>
```

(Figure 5.1 Use of display heading class)

# Display 1

## Display 2

### Display 3

#### Display 4

##### Display 5

###### Display 6

*(Figure 5.2 Display Heading Output)*

Now let's go to the **containers**.

Remember when I told you that the basic structure that we should have is:

**section**

**container**

**HTML Elements**

**container**

**section**

Why? Because we want our website to have a more **Semantic Structure**.

Bootstrap also has classes we can use for these containers. It comes in 3 different container structures.

- `.container`, sets a `max-width` at each responsive breakpoint
- `.container-fluid`, which is `width: 100%` at all breakpoints
- `.container-{breakpoint}`, which is `width: 100%` until the specified breakpoint

**Breakpoint** : predetermined screen sizes that our content and layout will automatically adjust to ensure and maintain readability and across different screen sizes (widths)

Below is a table that shows different container classes you can use and what is its width value according to each breakpoint:

	<b>Extra small</b> <576px	<b>Small</b> ≥576px	<b>Medium</b> ≥768px	<b>Large</b> ≥992px	<b>X-Large</b> ≥1200px	<b>XX-Large</b> ≥1400px
<code>.container</code>	100%	540px	720px	960px	1140px	1320px
<code>.container-sm</code>	100%	540px	720px	960px	1140px	1320px
<code>.container-md</code>	100%	100%	720px	960px	1140px	1320px
<code>.container-lg</code>	100%	100%	100%	960px	1140px	1320px
<code>.container-xl</code>	100%	100%	100%	100%	1140px	1320px
<code>.container-xxl</code>	100%	100%	100%	100%	100%	1320px
<code>.container-fluid</code>	100%	100%	100%	100%	100%	100%

What does this table mean?

Let me try to explain it.

For example, the `.container` class:

On a screen that has less than 576px screen width, the css value for the `max-width` property will be 100%. But on a screen that has a width greater than or equal to 576px but less than 768px, the `max-width` will be 540px. Or if a screen width is greater than or equal to 1200px but is less than 1400px, the `max-width` will be 1140px.

This is the same logic for all the container classes.

## Grid

Bootstrap's Grid System uses a series of containers, rows, and columns to layout and align content. It's built with flexbox and is fully responsive.

Here is an example on how Bootstrap's Grid System works:

```
<div class="container">
  <div class="row">
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
  </div>
</div>
```

(Figure 6.1 Grid Base Structure)

Column	Column	Column
--------	--------	--------

(Figure 6.2 Grid Base Structure Output)

The above is an example of a three equal-width column grid across all breakpoints. The columns are centered in the page by the parent div using the `.container` class.

### How does it work?

1. The `.container` class centers and horizontally pads the contents.
2. `.row` is the wrapper for columns. It creates the rows of the grid, while
3. `.col` is the column element of the grid. Each Bootstrap Grid has a 12 column template. With this, we can use the `.col-#` class to determine the span of a column. (`.col-4` spans four columns)



`.col-auto` makes the column width resize based on the natural width of the content.

```
<div class="container">
  <div class="row justify-content-center">
    <div class="col-2">
      1 of 3
    </div>
    <div class="col-auto">
      Variable width content
    </div>
    <div class="col-2">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-auto">
      Variable width content
    </div>
    <div class="col-2">
      3 of 3
    </div>
  </div>
</div>
```

(Figure 7.1 Code)

1 of 3			Variable width content			3 of 3					
1 of 3						Variable width content			3 of 3		

(Figure 7.2 Output)

The above code and output shows the versatility of the Grid System. The first row is centered within the container with the use of the `.justify-content-center` class.

## Breadcrumbs

Breadcrumbs, in best practice and proper structure, must be enclosed in a `<nav></nav>` tag.

The `<nav>` tag is used to define a set of navigation links.

Not all links should be placed inside a navigation tag because it is only intended for major navigation links.

With bootstrap, you will be using either an unordered list or an ordered list to create the Breadcrumbs.

The List wrapper (`<ol></ol>` or `<ul></ul>`) will be using the `.breadcrumb` class and each list item (`<li></li>`) will be using the `.breadcrumb-item` class. Each link will be using an anchor tag (`<a></a>`) and the last list item will be a plain list item and will have a class of `.active` to make it different from the rest of the links in the breadcrumb.

```
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item"><a href="#">Page 2</a></li>
    <li class="breadcrumb-item active" aria-current="page">Current Page</li>
  </ol>
</nav>
```

Figure 8.1  
Breadcrumb

Take note that having the `aria-label` in the list wrapper tag will give the navigation (breadcrumb) a meaningful label that will help screen readers. It is also the same with the use of the `aria-current`, it will give the current page a meaningful label that represents the current page.

---

[Home](#) / [Page 2](#) / Current Page

Figure 8.2  
Output

The Divider ("/") is automatically being added by Bootstrap so there is no need to type it in the HTML Document. But this can be changed, how? Here is how. This can be done using the Bootstrap Custom CSS Style Property `--bs-breadcrumb-divider`. It accepts a string value of what we want the divider to look like. It can also accept urls.

[Home](#) > Library

```
<nav style="--bs-breadcrumb-divider: '>';" aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>
```

Figure 9.1  
> as Divider

[Home](#) › Current Page

```
<nav style="--bs-breadcrumb-divider: url(&#34;data:image/svg+xml,%3Csvg xmlns='http://
www.w3.org/2000/svg' width='8' height='8'%3E%3Cpath d='M2.5 0L1 1.5 3.5 4 1 6.5 2.5 8l4-4-4-
4z' fill='currentColor'/%3E%3C/svg%3E&#34;);" aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Current Page</li>
  </ol>
</nav>
```

Figure 9.2  
SVG as Divider

## Navbar

Bootstrap has its own navbar style classes that are paired with its own script for further functionalities.

Navbars come with built-in support for a handful of sub-components. Choose from the following as needed:

- **.navbar-brand** for your company, product, or project name.
  - This can be an image, a text, or a combination of an image and a text
- **.navbar-nav** for a full-height and lightweight navigation (including support for dropdowns).
- **.navbar-toggler** for use with our collapse plugin and other navigation toggling behaviors.
- Flex and spacing utilities for any form controls and actions.
- **.navbar-text** for adding vertically centered strings of text.
- **.collapse.navbar-collapse** for grouping and hiding navbar contents by a parent breakpoint.
- Inside the div that uses the **.collapse.navbar-collapse** class, it should have an unordered list with each list item using the **.nav-item** class
- Each list item may have either a dropdown or a link.
- Each link (anchor tag) uses the **.nav-link** class to be combined with another class for additional functionality
- If we want our link to be a dropdown link, we add the **.dropdown-toggle** class
- Then for the dropdown items, on the same list item tag, create another list using the **.dropdown-menu** class and each list item will use the **.dropdown-item** class
- If we want our link to be the active link (current page we are on) we combine the **.active** class to it
- And if we want a link to be disabled, we use the **.disabled** class
- Each aria labels are used similar to how it was used in the breadcrumbs navigation, it gives a meaningful label to each element on what it is representing.
- Add an optional **.navbar-scroll** to set a **max-height** and scroll expanded navbar content.

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
      target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
      expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-
            bs-toggle="dropdown" aria-expanded="false">
            Dropdown
          </a>

```

Figure 10.1  
Code

```

      <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
        <li><a class="dropdown-item" href="#">Action</a></li>
        <li><a class="dropdown-item" href="#">Another action</a></li>
        <li><hr class="dropdown-divider"></li>
        <li><a class="dropdown-item" href="#">Something else here</a></li>
      </ul>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</
      a>
    </li>
  </ul>
</div>
</div>
</nav>

```

Figure 10.2  
Code Continuation

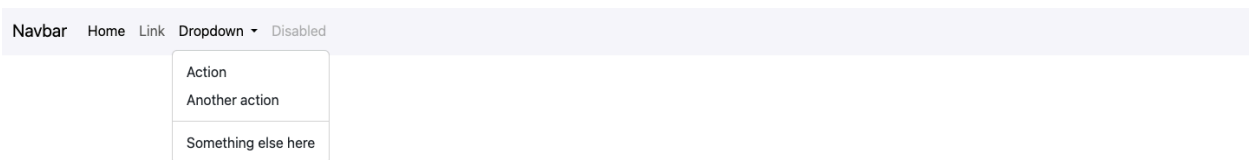


Figure 10.3  
Output

```

</li>
<li class="nav-item">
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</li>
</ul>
<form class="d-flex">
  <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
  <button class="btn btn-outline-success" type="submit">Search</button>
</form>
</div>
</div>
</nav>

```

Figure 10.4  
Search Bar Addition

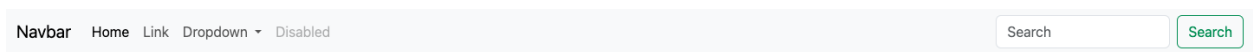


Figure 10.5  
Output

A search bar or a form can also be added in our navbar with the necessary classes to design our search bar and aria labels for screen readers.

That concludes our lesson for Week 10. There will be no assessments this week, so I trust that you will use this week to finish some of your assessments in Neo.

I trust that you will also read and study this document as I have created it in a way wherein I am sort of speaking. So it resembles a lesson. May you read this as I will no longer be repeating this lesson inside our classroom next week, Week 11, for we will be proceeding with the next part of the lesson.