Styling Web Pages with CSS

1 Introduction

Web pages are plain text documents formatted with ***HTML***. Different HTML elements -- or tags -- format plain text to change their appearance. For instance, heading tags format their text to be much larger than their surrounding plain text. Paragraph tags add vertical spacing. List tags enumerate or bullet the line items they surround. Table tags can format data into rows and columns. The default styling of Web pages is determined by the browser's ***style sheet*** written in ***CSS*** (***Cascade Style Sheets***), a declarative language that ***declares*** how each ***HTML*** element should look like.

To make Web pages stand out, we are going to learn how to use ***CSS*** to override the default styling to customize the Web pages look and feel. ***CSS*** works by referring to different parts of the ***DOM*** and configuring various appearance attributes such as foreground and background color, font, alignment, spacing, borders, paddings, etc. This assignment will give us a chance to learn about how to use CSS to style Web pages. CSS is a powerful language allowing complex Web page styling and layout. Various patterns and best practices have evolved over time in the industry that have become popular. Some of these have been collected into commercial and open source libraries such as:

- Bootstrap
- Foundation
- Tailwind
- Material Design
- Bulma

All these libraries define a set of ***CSS rules*** that can be readily applied to achieve a professional look and feel, powerful layouts, and ***responsive designs***. Using a library consists of becoming familiar with the CSS rules and applying them appropriately to your HTML to achieve a particular purpose. For this course we are going to be using ***Bootstrap*** throughout our assignments. Feel free to explore and use other libraries for your ***final project***.

The next section will give you an opportunity to practice various ***CSS*** concepts and the ***Bootstrap*** library. Once you've had a chance to practice, the ***Kanbas*** section will ask you to apply what you've learned to style the Web application started in previous assignments. Create a new branch called ***a2*** and do all your work there. When done, add, commit and push the branch to GitHub. Deploy the new branch to ***Netlify*** and confirm it's available in a new URL based on the branch name ***a2***.

2 Learning objectives

- Styling Web content with ***Cascading Style Sheets*** (***CSS***)
- Layout and responsive design with ***Bootstrap***
- Laying out Webpages with ***Bootstrap***
- Use ***React Icons***

3 Labs

This section presents several CSS, Bootstrap, and React Icons exercises to practice and learn how to style HTML documents. Use the same project you created last assignment. After you work through the exercises you will apply the skills to create ***Kanbas*** on your own. Using ***IntelliJ*** or ***VS Code***, open the project you created in the previous assignment (***kanbas-react-web-app***), and do all your work under the ***src*** directory of your project. Under the ***src/Labs*** directory, create a new directory called ***Lab2*** and create ***index.tsx*** under ***src/Labs/Lab2*** and do all your work in a new ***index.tsx*** file. Make sure TAs can navigate to the lab from the root of your application.

3.1 Styling Webpages with CSS (Cascading Style Sheets)

3.1.1 Styling elements with the style attribute

An HTML element's ***style*** attribute can configure the look and feel of the element by changing the values of its style ***properties*** as shown below. The value of the ***style*** attribute is an object in ***JSON*** format (***JavaScript Object Notation***).

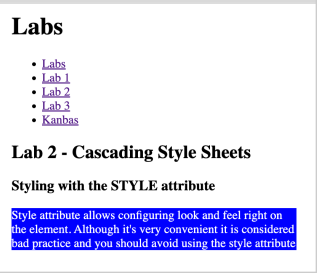
```
<element style={{property1: "value1", property2: "value2"}}>
  element body
</element>
```

Examples of properties ***property1*** and ***property2*** are foreground color, background color, font size, etc. The value of the properties are strings or numbers. To practice using the ***style*** attribute, copy and paste the example below into ***src/Labs/Lab2/index.tsx***

src/Labs/Lab2/index.tsx

```
export default function Lab2() {
  return (
    <div id="wd-lab2">
      <h2>Lab 2 - Cascading Style Sheets</h2>
      <h3>Styling with the STYLE attribute</h3>
      <p style={{ backgroundColor: "blue",
                  color: "white" }}>
        Style attribute allows configuring look and feel
        right on the element. Although it's very convenient
        it is considered bad practice and you should avoid
        using the style attribute
      </p>
    </div>
  );
}
```

Browser



In the exercise above we styled the paragraph element with its ***style*** attribute. We changed the color of its background by setting the ***backgroundColor*** property to ***blue*** and also changing the foreground color to white by setting the ***color*** property to ***white***. There are 100s of style attributes of which we'll cover the most relevant.

3.1.2 Importing CSS documents

Instead of changing styles within HTML, it is a ***best practice*** to do all styling configuration in separate CSS files and then import the files. To practice importing ***CSS*** files, create a brand new file called ***src/Labs/Lab2/index.css*** in the same directory of the ***src/Labs/Lab2/index.tsx*** document, and copy the following content.

src/Labs/Lab2/index.css

```
p {
   background-color: blue;
   color: white;
}
```

Then, as shown below, comment out the ***style*** attribute in ***index.tsx*** highlighted in red, since we won't be using it anymore. Instead, import the ***index.css*** file created earlier, as highlighted in green. Confirm that the browser renders the same.

src/Labs/Lab2/index.tsx

```
import "./index.css";
export default function Lab2() {
  return (
    <div id="wd-lab2">
      <h2>Lab 2 - Cascading Style Sheets</h2>
      <h3>Styling with the STYLE attribute</h3>
```

```
      <p style={{ backgroundColor: "blue", color: "white" }}>
        Style attribute allows configuring look and feel
        right on the element. Although it's very convenient
        it is considered bad practice and you should avoid
        using the style attribute
      </p>
    </div>
  );
}
```

3.1.3 Selecting content with ID selectors

The CSS rules in previous exercises styled all paragraphs at once by using the name of the tag ***p*** and then specifying the style property values. Instead of changing the look and feel of all the elements of the same name, e.g., ***p***, we can refer to a specific element by their ID, an attribute specifying a unique identifier. To practice using ID ***selectors***, in ***index.css***, comment out the highlighted paragraph CSS rule as shown and add the two CSS rules referring to paragraphs with IDs ***id-selector-1*** and ***id-selector-2***. Add the below code highlighted in green to ***index.tsx***, and confirm it renders as shown in the below on the right.

src/Labs/Lab2/index.css

```
/*p {
 background-color: blue;
 color: white;}*/
p#wd-id-selector-1 {
 background-color: red;
 color: white;
}
p#wd-id-selector-2 {
 background-color: yellow;
 color: black;
}
```

src/Labs/Lab2/index.tsx

```
import "./index.css";
export default function Lab2() {
  return (
    <div id="wd-lab2">
      <h2>Lab 2 - Cascading Style Sheets</h2>
      <h3>Styling with the STYLE attribute</h3>
      ...
      <div id="wd-css-id-selectors">
        <h3>ID selectors</h3>
        <p id="wd-id-selector-1">
Instead of changing the look and feel of all the
elements of the same name, e.g., P, we can refer
to a specific element by its ID
        </p>
        <p id="wd-id-selector-2">
Here's another paragraph using a different ID and
a different look and
          feel
        </p>
      </div>
    </div>
  );
}
```

Browser

ID selectors

Instead of changing the look and feel of all the elements of the same name, e.g., P, we can refer to a specific element by its ID

Here's another paragraph using a different ID and a different look and feel

3.1.4 Selecting content with class selectors

Instead of using IDs to refer to specific elements, you can use an element's ***class*** attribute instead, or a combination of both. ***Class selectors*** can be used just like ID selectors if you keep them unique, but can also be used to apply the same style to several elements, even if they are different types of elements. To practice using ***class selectors***, copy the CSS rule below into ***index.css***, and the HTML at the end of ***index.tsx***. The ellipses below (...) means that there's code above and/or below that we are not replicating here for brevity. Do not include the ellipses in your code.

src/Labs/Lab2/index.css

```
...
.wd-class-selector {
 background-color: yellow;
 color: blue;
}
```

src/Labs/Lab2/index.tsx

```
...
<div id="wd-css-class-selectors">
  <h3>Class selectors</h3>
  <p className="wd-class-selector">
Instead of using IDs to refer to elements, you
can use an element's CLASS attribute
  </p>
  <h4 className="wd-class-selector">
This heading has same style as paragraph above
  </h4>
</div>
...
```

Browser

Class selectors

Instead of using IDs to refer to elements, you can use an element's CLASS attribute

This heading has same style as paragraph above

The example above declares a selector that declares a style that transforms the background and foreground color. We can then use the selector to apply the transformation to several elements. The above example applies the style to two elements, the paragraph and the heading.

3.1.5 Selecting content based on the document structure

Selectors can be combined to refer to elements in particular places in the document. A set of selectors separated by a space can refer to elements in a hierarchy. For instance: ***.selector1 .selector2 { … }*** refers to an element whose class is ***.selector2*** and is inside some ***descendant*** of another element whose class is ***.selector1***. If we use ">" instead to separate the classes, then we can establish a direct parent/child relationship. To practice selecting elements using a set of selectors, copy the following content in ***index.tsx***. The code below does not show ellipses as previous examples, but the code is intended to be copied at the end of the file shown. Be sure to include the code within the function's return and within the wrapping *** ***.

src/Labs/Lab2/index.tsx

```
<div id="wd-css-document-structure">
  <div className="wd-selector-1">
    <h3>Document structure selectors</h3>
    <div className="wd-selector-2">
      Selectors can be combined to refer elements in particular
      places in the document
      <p className="wd-selector-3">
        This paragraph's red background is referenced as
        <br />
        .selector-2 .selector3<br />
        meaning the descendant of some ancestor.<br />
        <span className="wd-selector-4">
          Whereas this span is a direct child of its parent
        </span><br />
          You can combine these relationships to create specific
          styles depending on the document structure
      </p>
    </div>
  </div>
</div>
```

```
<!-- this is parent element with selector
  .wd-selector-1
  .wd-selector-2 is a direct child of
  .wd-selector-1
  .wd-selector-3 is a descendant of
  .wd-selector-1 and a direct child of
  .wd-selector-2


  this is a descendant of .selector-1 and
  .wd-selector-2 and a direct child of
  .wd-selector-3
-->
```

Let's now style elements ***.wd-selector-3*** and ***.wd-selector-4***. Copy the CSS below into ***index.css***.

src/Labs/Lab2/index.css

```
.wd-selector-1 .wd-selector-3 {
   background-color: red;
   color: white;
}
.wd-selector-2 > .wd-selector-3 > .wd-selector-4 {
   background-color: yellow;
   color: blue;
}
```

```
/* refers to .wd-selector-3 as a descendant of .wd-selector-1   */


/* refers to .wd-selector-4 as a direct child of .wd-selector-3 */
/* which is a direct child of .wd-selector-2                    */
```

3.1.6 CSS Rule Mechanism

The CSS rules we are writing are overriding the default styling of the elements the rules refer to. That means that there are at least two rules that apply to the same element, the default rule declared by the browser, and then our rule that overrides the default rule. Our rules wins out over the default rule, but why? The ***Cascading*** in ***Cascading Style Sheets*** refer to the way CSS rules are applied to HTML elements based on a hierarchy of specificity and origin. This mechanism determines how styles are applied when there are multiple style rules that could affect an element. Here's how it works:

1. ***Specificity***: CSS rules are applied based on their specificity. This is a measure of how precise a selector is. More specific selectors override more general ones. For example, an ID selector is more specific than a class selector, and a class selector is more specific than a tag name selector.
2. ***Source Order***: If multiple rules have the same specificity, the last rule defined in the CSS will take precedence.

3. ***Inheritance***: Some styles are inherited by child elements from their parent elements, such as font styles. However, properties like width and margin are not inherited.

3.1.7 Styling the foreground color

Foreground colors can be configured using the CSS ***color*** property as follows

```
.some-css-selector {                    /* selects some DOM element    */
   color: blue;                         /* sets color property to blue */
}
```

Colors can be defined as follows

- As strings, e.g., white, red, blue, etc
- As hexadecimals, e.g., #ABCDEF
- As RGB, e.g., rgb(12, 34, 56)

Here are a couple examples:

`.the-sun { color: rgb(255,255,0); }`	`.the-sky { color: blue; }`	`.ketchup { color: #FF0000; }`

To practice working with foreground colors, copy the CSS rules below into ***index.css*** to declare several useful color classes. Copy the HTML code below into ***ForegroundColors.tsx***. Import the file into ***Lab2*** and confirm it renders as shown.

src/Labs/Lab2/index.css

```
.wd-fg-color-black { color: black; }
.wd-fg-color-white { color: white; }
.wd-fg-color-blue { color: #7070ff; }
.wd-fg-color-red { color: #ff7070; }
.wd-fg-color-green { color: green; }
```

src/Labs/Lab2/ForegroundColors.tsx

```
<div id="wd-css-colors">
  <h2>Colors</h2>
  <h3 className="wd-fg-color-blue">Foreground color</h3>
  <p className="wd-fg-color-red">
    The text in this paragraph is red but
    <span className="wd-fg-color-green">this text is green</span>
  </p>
</div>
```

Foreground color

The text in this paragraph is red but this text is green

3.1.8 Styling the background color

To practice working with background colors, copy the CSS rules shown below into ***index.css*** and the HTML code into ***BackgroundColors.tsx***. Import the file into ***Lab2*** and confirm the browser renders as shown.

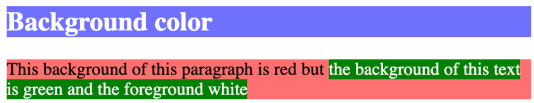
src/Labs/Lab2/index.css

```
.wd-bg-color-yellow {
   background-color: #ffff07;
}
.wd-bg-color-blue {
   background-color: #7070ff;
}
.wd-bg-color-red {
   background-color: #ff7070;
}
.wd-bg-color-green {
```

src/Labs/Lab2/BackgroundColors.tsx

```
<div id="wd-css-background-colors">
  <h3 className="wd-bg-color-blue wd-fg-color-white">Background color</h3>
  <p className="wd-bg-color-red wd-fg-color-black">
    This background of this paragraph is red but
    <span className="wd-bg-color-green wd-fg-color-white">
      the background of this text is green and the foreground white
    </span>
  </p>
</div>
```

```
   background-color: green;
}
.wd-bg-color-gray {
   background-color: lightgray;
}
```



3.1.9 Styling borders

Use CSS border properties to configure the look and feel of the border around content. Here's a sample of the properties that can be configured.

```
.some-selector {
  border-width: 10px;
  border-style: solid|dotted|dashed|double;
  border-color: red | blue ...;
}
```

```
/* configure border with several properties*/
/* border's width. Can also provide per border*/
/* the style of the border*/
/* the color of the border     */
```

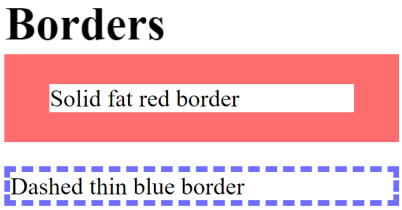
To practice styling borders, copy the CSS code below into ***index.css*** and the HTML code into ***Borders.tsx*** and confirm the browser renders as shown.

src/Labs/Lab2/index.css

```
.wd-border-fat {
   border-width: 20px 30px 20px 30px; }
.wd-border-thin {
   border-width: 4px; }
.wd-border-solid {
   border-style: solid; }
.wd-border-dashed {
   border-style: dashed; }
.wd-border-yellow {
   border-color: #ffff07; }
.wd-border-red {
   border-color: #ff7070; }
.wd-border-blue {
   border-color: #7070ff; }
```

src/Labs/Lab2/Borders.tsx

```
<div id="wd-css-borders">
  <h2>Borders</h2>
  <p className="wd-border-fat wd-border-red wd-border-solid">
    Solid fat red border</p>
  <p className="wd-border-thin wd-border-blue wd-border-dashed">
    Dashed thin blue border</p>
</div>
```



3.1.10 Styling margins and paddings

You can also configure the spacing between elements. To practice with ***padding***s, copy the CSS code below into ***index.css*** and the HTML into ***Padding.tsx***. Confirm browser renders as shown.

src/Labs/Lab2/index.css

```
.wd-padded-top-left {
  padding-top: 50px;
  padding-left: 50px;
}
```

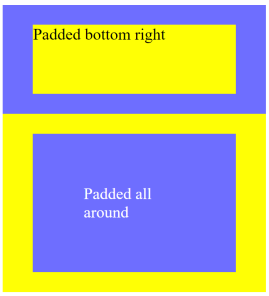
src/Labs/Lab2/Padding.tsx

```
<div id="wd-css-paddings">
  <h2>Padding</h2>
  <div className="wd-padded-top-left wd-border-fat
      wd-border-red wd-border-solid
      wd-bg-color-yellow">
    Padded top left
  </div>
```

Browser

```
.wd-padded-bottom-right {
  padding-bottom: 50px;
  padding-right: 50px;
}


.wd-padding-fat {
  padding: 50px;
}
```

```
  <div className="wd-padded-bottom-right wd-border-fat
      wd-border-blue wd-border-solid
      wd-bg-color-yellow">
    Padded bottom right
  </div>

  <div className="wd-padding-fat wd-border-fat
      wd-border-yellow wd-border-solid
      wd-bg-color-blue wd-fg-color-white">
    Padded all around
  </div>
</div>
```



To practice with margins, copy the CSS code below into ***index.css*** and the HTML into ***Margins.tsx***. Confirm browser renders as shown.

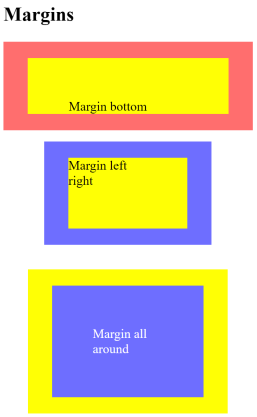
src/Labs/Lab2/index.css

```
.wd-margin-bottom {
  margin-bottom: 50px;
}

.wd-margin-right-left {
  margin-left: 50px;
  margin-right: 50px;
}

.wd-margin-all-around {
  margin: 30px;
}
```

src/Labs/Lab2/Margins.tsx

```
<div id="wd-css-margins">
  <h2>Margins</h2>
  <div
    className="wd-margin-bottom
      wd-padded-top-left
      wd-border-fat wd-border-red
      wd-border-solid
      wd-bg-color-yellow">
    Margin bottom
  </div>
  <div
    className="wd-margin-right-left
      wd-padded-bottom-right
      wd-border-fat wd-border-blue
      wd-border-solid
      wd-bg-color-yellow">
    Margin left right
  </div>
  <div
    className="wd-margin-all-around
      wd-padding-fat wd-border-fat
      wd-border-yellow
      wd-border-solid
      wd-bg-color-blue
      wd-fg-color-white">
    Margin all around
  </div>
</div>
```

Browser



3.1.11 Styling corners

You can configure the corners of element borders to be rounded. Either all of them at once or specific corners. You can do this by configuring a border's radius. To practice rounding some corners, copy the CSS and HTML below into ***index.css*** and ***Corners.tsx*** and confirm the browser renders as shown.

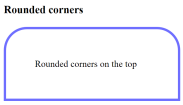
src/Labs/Lab2/index.css

```
.wd-rounded-corners-top {
  border-top-left-radius: 40px;
  border-top-right-radius: 40px;
}
```

src/Labs/Lab2/Corners.tsx

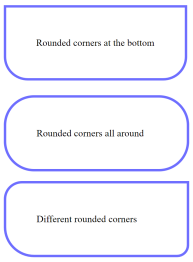
```
<div id="wd-css-borders">
  <h3>Rounded corners</h3>
  <p className="wd-rounded-corners-top wd-border-thin
     wd-border-blue wd-border-solid wd-padding-fat">
    Rounded corners on the top
  </p>
```

Browser



```
.wd-rounded-corners-bottom {
 border-bottom-left-radius: 40px;
 border-bottom-right-radius: 40px;
}

.wd-rounded-corners-all-around {
 border-radius: 50px;
}

.wd-rounded-corners-inline {
 border-radius: 30px 0px 20px 50px;
}
```

```
  <p className="wd-rounded-corners-bottom
     wd-border-thin wd-border-blue wd-border-solid
     wd-padding-fat">
    Rounded corners at the bottom
  </p>
  <p className="wd-rounded-corners-all-around
     wd-border-thin wd-border-blue wd-border-solid
     wd-padding-fat">
    Rounded corners all around
  </p>
  <p className="wd-rounded-corners-inline
     wd-border-thin wd-border-blue wd-border-solid
     wd-padding-fat">
    Different rounded corners
  </p>
</div>
```



3.1.12 Styling dimensions

You can configure an element's dimensions with ***width*** and ***height*** properties. To practice setting element's dimensions, copy the CSS and HTML below into ***index.css*** and ***Dimensions.tsx*** and confirm the browser renders as shown.

src/Labs/Lab2/index.css

```
.wd-dimension-portrait {
   width: 75px;
   height: 100px;
}

.wd-dimension-landscape {
   width: 100px;
   height: 75px;
}

.wd-dimension-square {
   width: 75px;
   height: 75px;
}
```

src/Labs/Lab2/Dimensions.tsx

```
<div id="wd-css-dimensions">
  <h2>Dimension</h2>
  <div>
    <div className="wd-dimension-portrait wd-bg-color-yellow">
      Portrait
    </div>
    <div className="wd-dimension-landscape wd-bg-color-blue
                     wd-fg-color-white">
      Landscape
    </div>
    <div className="wd-dimension-square wd-bg-color-red">
      Square</div>
  </div>
</div>
```

Browser

3.1.13 Styling relative position

You can configure an element's position with the ***position*** property. The property has many possible values, but we'll explore ***relative***, ***absolute***, and ***static***. Setting ***position*** property to ***relative*** allows moving the element relative to its original position. To practice setting element's relative position, copy the CSS and HTML below into ***index.css*** and ***Positions.tsx*** and confirm the browser renders as shown.

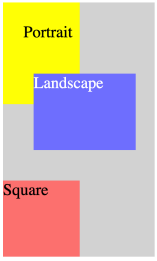
src/Labs/Lab2/index.css

```
.wd-pos-relative-nudge-up-right {
   position: relative;
   bottom: 30px;
   left: 30px;
}
.wd-pos-relative-nudge-down-right {
   position: relative;
   top: 20px;
   left: 20px;
}
.wd-pos-relative {
   position: relative;
}
```

src/Labs/Lab2/Positions.tsx

```
<div id="wd-css-position-relative">
  <h2>Relative</h2>
  <div className="wd-bg-color-gray">
    <div className="wd-bg-color-yellow
                     wd-dimension-portrait">
      <div className="wd-pos-relative-nudge-down-right">
        Portrait</div></div>
    <div className="wd-pos-relative-nudge-up-right
        wd-bg-color-blue wd-fg-color-white
        wd-dimension-landscape">
      Landscape</div>
    <div className="wd-bg-color-red wd-dimension-square">
      Square</div>
  </div>
</div>
```

Browser



3.1.14 Styling absolute position

Setting ***position*** property to ***absolute*** allows moving the element relative to the position of its parent. To practice setting element's absolute position, copy the CSS and HTML below into ***index.css*** and ***Positions.tsx***. Notice several ***
*** elements were added at the end of the example to make room for the next exercise.

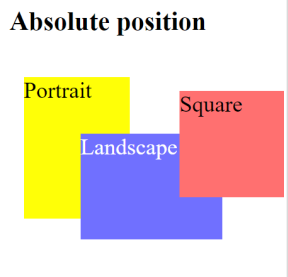
src/Labs/Lab2/index.css

```
.wd-pos-absolute-10-10 {
   position: absolute;
   top: 10px;
   left: 10px;
}
.wd-pos-absolute-50-50 {
   position: absolute;
   top: 50px;
   left: 50px;
}
.wd-pos-absolute-120-20 {
   position: absolute;
   top: 20px;
   left: 120px;      }
```

src/Labs/Lab2/Positions.tsx

```
<div id="wd-css-position-absolute">
  <h2>Absolute position</h2>
  <div className="wd-pos-relative">
    <div className="wd-pos-absolute-10-10
         wd-bg-color-yellow wd-dimension-portrait">
      Portrait</div>
    <div className="wd-pos-absolute-50-50
         wd-bg-color-blue wd-fg-color-white
         wd-dimension-landscape">
      Landscape</div>
    <div className="wd-pos-absolute-120-20
         wd-bg-color-red wd-dimension-square">
      Square</div>
  </div><br /><br /><br /><br /><br /><br /><br />
</div>
```

Browser



3.1.15 Styling fixed position

Setting ***position*** property to ***fixed*** allows setting the element relative to the window. That means that if you scroll the content of the page, the element will not scroll with it. To practice setting element's fixed position, copy the CSS and HTML below into ***index.css*** and ***Positions.tsx*** and confirm the browser renders as shown. Your display may be different depending on the actual size of the screen and scrolling.

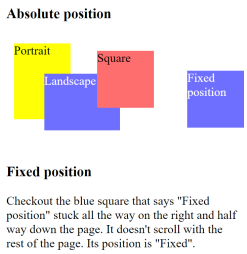
src/Labs/Lab2/index.css

```
.wd-pos-fixed {
  position: fixed;
  right: 0px;
  bottom: 50%;
}
```

src/Labs/Lab2/Positions.tsx

```
<div id="wd-css-position-fixed">
  <h2>Fixed position</h2>
  Checkout the blue square that says "Fixed
position" stuck all the way on the right
and half way down the page. It doesn't
scroll with the rest of the page. Its
position is "Fixed".
  <div className="wd-pos-fixed
    wd-dimension-square wd-bg-color-blue
    wd-fg-color-white">
    Fixed position
  </div>
</div>
```

Browser



3.1.16 Styling z-index

When the browser renders content declared in HTML documents, it calculates positions and dimensions so every element has a dedicated rectangle on the window. Typically elements don't fall on top of each other. When you start moving elements with ***position***, then overlapping elements are possible. By default elements are rendered in the order declared in HTML documents. Elements declared later render on top of elements declared earlier. The ***z-index*** CSS property overrides this behavior. Default value of ***z-index*** is ***auto***, which corresponds to 0. Increasing z-index can make elements render later, or on top of, others. To practice setting an element's ***z-index***, copy the CSS and HTML below into ***index.css*** and ***Zindex.tsx***.

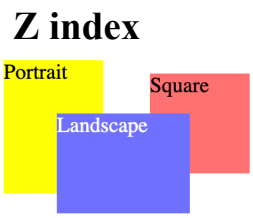
src/Labs/Lab2/index.css

```
.wd-zindex-bring-to-front {
   z-index: 10;
}
```

src/Labs/Lab2/Zindex.tsx

```
<div id="wd-z-index">
  <h2>Z index</h2>
  <div className="wd-pos-relative">
    <div className="wd-pos-absolute-10-10
         wd-bg-color-yellow wd-dimension-portrait">
      Portrait
    </div>
    <div className="wd-zindex-bring-to-front
         wd-pos-absolute-50-50 wd-dimension-landscape
         wd-bg-color-blue wd-fg-color-white">
      Landscape
    </div>
    <div className="wd-pos-absolute-120-20
         wd-bg-color-red wd-dimension-square">
      Square
    </div>
  </div><br /><br /><br /><br /><br />
</div>
```

Browser



3.1.17 Floating Images and Content

HTML does not support laying out content horizontally. The CSS float property allows fixing that. To practice laying out content horizontally, copy the CSS and HTML below into ***index.css*** and ***Float.tsx*** and confirm the browser renders as shown.

src/Labs/Lab2/index.css

```
.wd-float-left {
   float: left;
}

.wd-float-right {
   float: right;
   height: 100px;
}

.wd-float-done {
   clear: both;
}
```

src/Labs/Lab2/Float.tsx

```
<div id="wd-float-divs">
 <h2>Float</h2>
 <div>
   <div className="wd-float-left wd-dimension-portrait wd-bg-color-yellow">
     Yellow </div>
   <div className="wd-float-left wd-dimension-portrait wd-bg-color-blue wd-fg-color-white">
     Blue </div>
   <div className="wd-float-left wd-dimension-portrait wd-bg-color-red">
     Red </div>
   <img className="wd-float-right"
     src="https://www.staradvertiser.com/wp-content/uploads/2021/08/web1_Starship-gap2.jpg"/>
   <div className="wd-float-done"></div>
 </div>
</div>
```

3.1.18 Laying out content in a grid

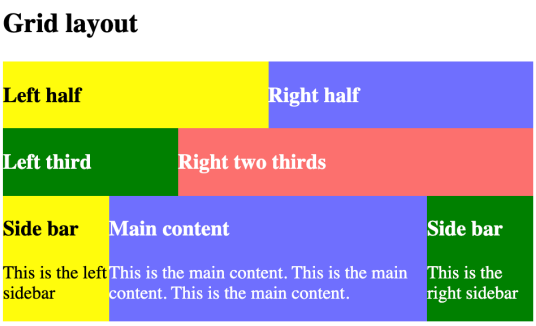
Using float we can implement a grid layout made up of rows and columns. To practice laying out content in a grid, copy the CSS and HTML below into ***index.css*** and ***GridLayout.tsx*** and confirm the browser renders as shown.

src/Labs/Lab2/index.css

```css
.wd-grid-row {
   clear: both;
}

.wd-grid-col-half-page {
   width: 50%;
   float: left;
}

.wd-grid-col-third-page {
   width: 33%;
   float: left;
}

.wd-grid-col-two-thirds-page
{
   width: 67%;
   float: left;
}

.wd-grid-col-left-sidebar {
   width: 20%;
   float: left;
}

.wd-grid-col-main-content {
   width: 60%;
   float: left;
}

.wd-grid-col-right-sidebar {
   width: 20%;
   float: left;
}
```

src/Labs/Lab2/GridLayout.tsx

```tsx
<div id="wd-css-grid-layout">
  <div id="wd-css-left-right-layout">
    <h2>Grid layout</h2>
    <div className="wd-grid-row">
      <div className="wd-grid-col-half-page wd-bg-color-yellow">
        <h3>Left half</h3>
      </div>
      <div className="wd-grid-col-half-page wd-bg-color-blue wd-fg-color-white">
        <h3>Right half</h3>
      </div>
    </div>
  </div>
  <div id="wd-css-left-third-right-two-thirds" className="wd-grid-row">
    <div className="wd-grid-col-third-page wd-bg-color-green
        wd-fg-color-white">
      <h3>Left third</h3>
    </div>
    <div className="wd-grid-col-two-thirds-page wd-bg-color-red wd-fg-color-white">
      <h3>Right two thirds</h3>
    </div>
  </div>
  <div id="wd-css-side-bars" className="wd-grid-row">
    <div className="wd-grid-col-left-sidebar wd-bg-color-yellow">
      <h3>Side bar</h3>
      <p>This is the left sidebar</p>
    </div>
    <div className="wd-grid-col-main-content wd-bg-color-blue wd-fg-color-white">
      <h3>Main content</h3>
      <p>
        This is the main content. This is the main content. This is the
        main content.
      </p>
    </div>
    <div className="wd-grid-col-right-sidebar wd-bg-color-green wd-fg-color-white">
      <h3>Side bar</h3>
      <p>This is the right sidebar</p>
    </div>
  </div>
</div>
```



3.1.19 Flex

Flexbox Layout (***Flexible Box***, or just ***Flex***) provides a simpler way to layout and distribute content in an HTML document. It all starts with creating a container element that configures the layout and behavior of its child elements. To illustrate some of the features of flex, let's create create a container with display configured to flex as shown below.

src/Labs/Lab2/Flex.tsx	src/Labs/Lab2/index.css	Browser

`<div id="wd-css-flex">`
`  <h2>Flex</h2>`
`  <div className="wd-flex-row-container">`
`  <div className="wd-bg-color-yellow">Column 1</div>`
`  <div className="wd-bg-color-blue">Column 2</div>`
`  <div className="wd-bg-color-red">Column 3</div>`
`  </div>`
`</div>`
	`.wd-flex-row-container {`
	`  display: flex;`
	`  flex-direction: row;`
	`}`
		Flex
		Column 1 Column 2 Column 3

Note how DIV elements inside the container render horizontally as a row of element instead of stacking the elements vertically. Flex simplifies laying out content horizontally. Flex child elements can also be configured to grow and expand to fill into empty spaces. The styling below illustrates how the last column can expand into the empty space to its right.

src/Labs/Lab2/Flex.tsx	src/Labs/Lab2/index.css	Browser

`<div id="wd-css-flex">`
`  <h2>Flex</h2>`
`  <div className="wd-flex-row-container">`
`  <div className="wd-bg-color-yellow">`
`  Column 1</div>`
`  <div className="wd-bg-color-blue">`
`  Column 2</div>`
`  <div className="wd-bg-color-red`
`  wd-flex-grow-1">`
`  Column 3</div>`
`  </div>`
`</div>`
	`.wd-flex-row-container {`
	`  display: flex;`
	`  flex-direction: row;`
	`}`
	`.wd-flex-grow-1 {`
	`  flex-grow: 1;`
	`}`
		Flex
		Column 1 Column 2 Column 3

The rest of the flex child elements can be configured independently to have specific widths to fit whatever content is needed as shown below.

src/Labs/Lab2/Flex.tsx	src/Labs/Lab2/index.css	Browser

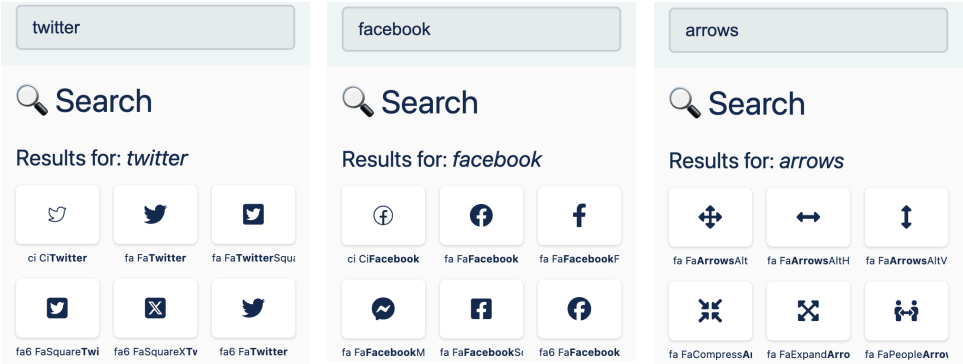
`<div id="wd-css-flex">`
`  <h2>Flex</h2>`
`  <div className="wd-flex-row-container">`
`  <div className="wd-bg-color-yellow`
`  wd-width-75px">`
`  Column 1</div>`
`  <div className="wd-bg-color-blue">`
`  Column 2</div>`
`  <div className="wd-bg-color-red`
`  wd-flex-grow-1">`
`  Column 3</div>`
`  </div>`
`</div>`
	`.wd-flex-row-container {`
	`  display: flex;`
	`  flex-direction: row;`
	`}`
	`.wd-flex-grow-1 {`
	`  flex-grow: 1;`
	`}`
	`.wd-width-75px {`
	`  width: 75px;`
	`}`
		Flex
		Column 1 Column 2 Column 3

3.2 Decorating Documents with React Icons

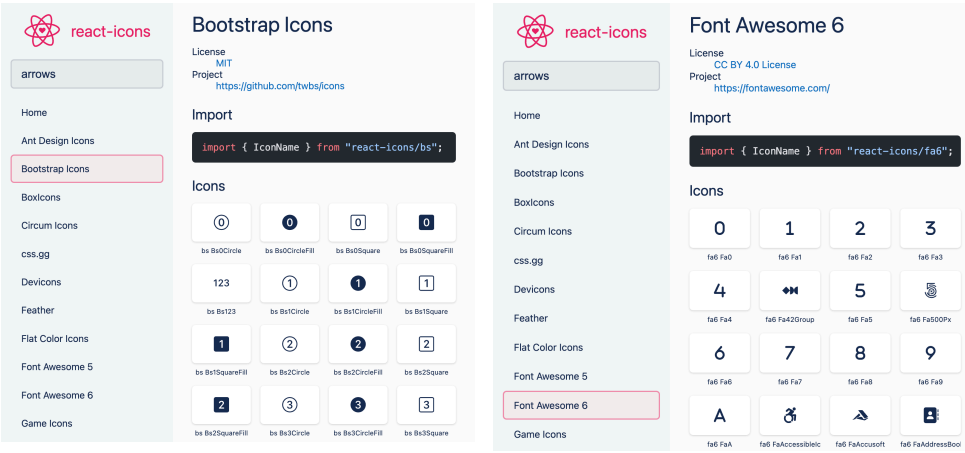
React icons is a CSS library that aggregates icons from various sources including Fontawesome, Bootstrap, and Heroicons. Install React icons from the root of your project as shown below.

```
$ npm install react-icons                # install react icons from the root of your project
```

Head over to https://react-icons.github.io/react-icons to search for icons of interest. Search for icons by typing a topic in the search box. Here are a few example searches found from several sources.



You can also navigate through the icons by their vendors as shown below, e.g, ***Bootstrap*** and ***Fontawesome***.



The icons are made available as ***React.js*** components you can import and use as tags within your HTML code. To practice using ***React Icons***, create the ***ReactIconSampler*** component as shown below. Import the component at the end of ***Lab2*** component and confirm it renders as shown below on the right.

src/Labs/Lab2/ReactIcons.tsx

```
import { FaCalendar, FaEnvelopeOpenText, FaRegClock }
  from "react-icons/fa";
import { AiOutlineDashboard } from "react-icons/ai";
import { FaBookBible } from "react-icons/fa6";
import { VscAccount } from "react-icons/vsc";
export default function ReactIconsSampler() {
  return (
    <div id="wd-react-icons-sampler" className="mb-4">
      <h3>React Icons Sampler</h3>
      <div className="d-flex">
        <VscAccount className="fs-3 text" />
```

```
        <AiOutlineDashboard className="fs-3 text" />
        <FaBookBible className="fs-3 text" />
        <FaCalendar className="fs-3 text" />
        <FaEnvelopeOpenText className="fs-3 text" />
        <FaRegClock className="fs-3 text" />
      </div>
    </div>
  );
}
```

3.3 Styling Webpages with the Bootstrap CSS Library

Bootstrap is a ***CSS*** library containing a plethora of useful CSS rules that implement various widgets, layouts, and responsive design. This section presents exercises of how to use the ***Bootstrap*** CSS library to style and layout Web pages.

3.3.1 Installing bootstrap

Bootstrap can be installed with ***npm*** from the root of you project as follows.

```
$ npm install bootstrap                    # install bootstrap from the root of your project
```

The ***bootstrap*** CSS library will be downloaded from ***npm*** and installed in you ***node_modules*** directory located at the root of your project. Once installed, import the library from ***src/index.tsx*** as shown below.

src/index.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import "bootstrap/dist/css/bootstrap.min.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
const root = ReactDOM.createRoot(document.getElementById("root") as HTMLElement);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

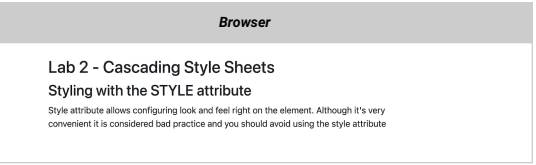
3.3.2 Laying out content with containers

Bootstrap containers establish the root of your HTML document providing a basis of default styles such as the overall margins, paddings, and font of your document. There are two main classes that control container elements: ***.container*** and ***.container-fluid***. The ***.container*** class centers the content with margins on either side and defines several responsive design thresholds. The ***.container-fluid*** class just defines a constant thin margin all around the document. To practice with containers, add ***container*** to the root ***div*** that contains the ***Lab2*** content as shown below. Refresh the browser and confirm it looks similar to image shown. The heading is not flush with the left hand side and the font is not the default browser font.

src/Labs/Lab2/index.tsx

```
import "./index.css";
export default function Lab2() {
  return (
    <div className="container">
      <h2>Lab 2 - Cascading Style Sheets</h2>
      <h3>Styling with the STYLE attribute</h3>
```

Browser



3.3.3 Laying out content with grids

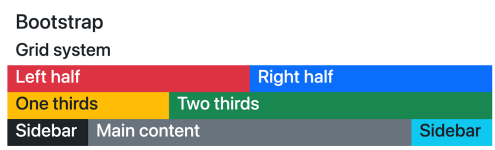
It's easy to break a page vertical in HTML with the ***p*** and ***div*** tags. It's a little harder to layout content horizontally. Some resort to HTML tables to layout content horizontally using table ***rows*** and ***columns***, but this is generally considered a bad practice. HTML tables should be used for displaying tabular content only, not laying out HTML content. Nevertheless, laying out content like a table is convenient, so to achieve the same functionality as tables, but without tables, we can use CSS instead. Bootstrap provides classes such as ***.row*** and ***.col*** to layout content in a ***grid***. To practice with ***Bootstrap grids***, create a ***BootstrapGrids*** component using the code below, and import it to ***Lab2***. Refresh the browser and confirm it looks similar to image shown.

src/Labs/Lab2/BootstrapGrids.tsx

```
<h2>Bootstrap</h2>
<div id="wd-bs-grid-system">
  <h2>Grid system</h2>
  <div className="row">
    <div className="col bg-danger text-white">
      <h3>Left half</h3>
    </div>
    <div className="col bg-primary text-white">
      <h3>Right half</h3>
    </div>
  </div>
  <div className="row">
    <div className="col-4 bg-warning">
      <h3>One thirds</h3>
    </div>
    <div className="col-8 bg-success text-white">
      <h3>Two thirds</h3>
    </div>
  </div>
  <div className="row">
    <div className="col-2 bg-dark text-white">
      <h3>Sidebar</h3>
    </div>
    <div className="col-8 bg-secondary text-white">
      <h3>Main content</h3>
    </div>
    <div className="col-2 bg-info">
      <h3>Sidebar</h3>
    </div>
  </div>
</div>
```

Comments / Browser

```
<!-- a row containing
   two columns applying class col with default
   width evenly spread over all columns


   another row with
   two columns applying classes col-4 and col-8
   where 4 + 8 = 12, the total number of columns
   so 4/12 is ⅓ and 8/12 is ⅔ of the screen
-->
```



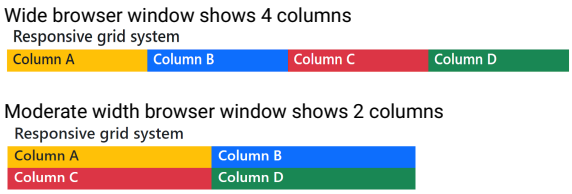
3.3.4 Responsive Grids

Bootstrap grids can adapt to the size of the screen, that is, they can be responsive. We can achieve this by applying more than one ***.col*** class which only applies for a given window size. To practice with ***Bootstrap responsive*** grids, create a ***BootstrapGrids*** component as shown below. Confirm it looks similar to image shown. Resize the browser and confirm that the grid shows 4 columns, then 2 and then just 1.

src/Labs/Lab2/BootstrapGrids.tsx

```
<div id="wd-bs-responsive-grids">
  <h2>Responsive grid system</h2>
  <div className="row">
    <div className="col-12 col-md-6 col-xl-3
                    bg-warning">
      <h3>Column A</h3>
    </div>
    <div className="col-12 col-md-6 col-xl-3
                    bg-primary text-white">
      <h3>Column B</h3>
    </div>
    <div className="col-12 col-md-6 col-xl-3
                    bg-danger text-white">
      <h3>Column C</h3>
```

Comments / Browser



```
    </div>
    <div className="col-12 col-md-6 col-xl-3
                    bg-success text-white">
      <h3>Column D</h3>
    </div>
  </div>
</div>
```

Thin browser window shows only 1 column

Responsive grid system

Column A

Column B

Column C

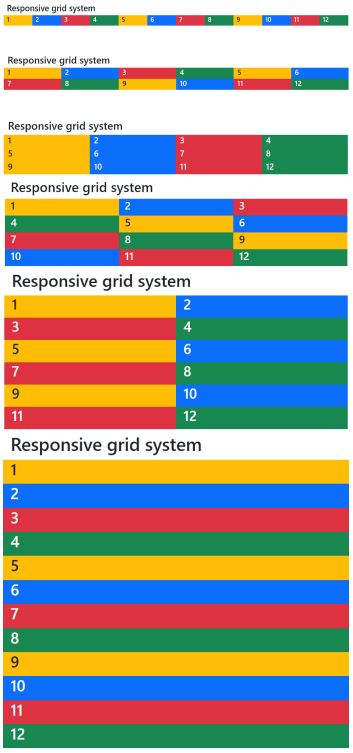
Column D

Let's try a more dramatic example by adding more content spread over more columns and rows. Copy the HTML code below to the end of ***BootstrapGrids.tsx***, and save. Refresh the browser and confirm that the column layout changes as you resize the browser window.

src/Labs/Lab2/BootstrapGrids.tsx

```
<div id="wd-bs-responsive-dramatic">
  <h2>Responsive grid system</h2>
  <div className="row">
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-warning">
      <h4>1</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-primary text-white">
      <h4>2</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-danger text-white">
      <h4>3</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-success text-white">
      <h4>4</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-warning">
      <h4>5</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-primary text-white">
      <h4>6</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-danger text-white">
      <h4>7</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-success text-white">
      <h4>8</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-warning">
      <h4>9</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-primary text-white">
      <h4>10</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-danger text-white">
      <h4>11</h4>
    </div>
    <div className="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 col-xxl-1
                    bg-success text-white">
      <h4>12</h4>
    </div>
  </div>
</div>
```

Browser



3.3.5 Hiding and showing responsive content

As users shrink or widen the browser window, there may be more or less space to show some content. Bootstrap can configure content to show or hide depending on the width of the screen. As described earlier in Responsive grids, Bootstrap breaks up the screen into 5 sizes: ***extra small***, ***small***, ***medium***, ***large***, ***extra large***, and ***extra extra large***. Create a new component called ***ScreenSizeLabel*** as shown below which displays and hides different labels at different screen sizes. Add the styling to ***index.css*** to position the label at the top left corner. Create a component as shown below and confirm that the label displays the current screen size when you resize the window.

src/Labs/Lab2/ScreenSizeLabel.tsx

```
export default function ScreenSizeLabel() {
  return (
    <div id="wd-screen-size-label">
      <div className="d-block d-sm-none">
        XS - Extra Small (&lt;576px)
      </div>
      <div className="d-none d-sm-block d-md-none">
        S - Small (≥576px)
      </div>
      <div className="d-none d-md-block d-lg-none">
        M - Medium (≥768px)
      </div>
      <div className="d-none d-lg-block d-xl-none">
        L - Large (≥992px)
      </div>
      <div className="d-none d-xl-block d-xxl-none">
        XL - Extra Large (≥1200px)
      </div>
      <div className="d-none d-xxl-block">
        XXL - Extra Extra Large (≥1400px)
      </div>
    </div>
  );
}
```

src/Labs/Lab2/index.css

```
#wd-screen-size-label {
  position: fixed;
  top: 0;
  left: 0;
  background-color: black;
  color: white;
  padding: 5px;
  font-size: 12px;
  z-index: 1000;
  width: 220px;
  text-align: center;
}
```

3.3.6 Styling tables

Bootstrap provides several classes that enhance the look and feel of common HTML widgets such as tables, lists, and form elements. Let's start with tables. To practice with styling ***HTML tables***, create a component with the code shown below. Refresh the browser and confirm it looks similar to image shown.

src/Labs/Lab2/BootstrapTables.tsx

```
<div id="wd-css-styling-tables">
  <h2>Tables</h2>
  <table className="table">
    <thead>
      <tr className="table-dark"><th>Quiz</th><th>Topic</th><th>Date</th><th>Grade</th></tr>
    </thead>
    <tbody>
      <tr className="table-warning"><td>Q1</td><td>HTML</td><td>2/3/21</td><td>85</td></tr>
      <tr className="table-danger"><td>Q2</td><td>CSS</td><td>2/10/21</td><td>90</td></tr>
      <tr className="table-primary"><td>Q3</td><td>JavaScript</td><td>2/17/21</td><td>90</td></tr>
    </tbody>
    <tfoot>
      <tr className="table-success"><td colSpan={3}>Average</td><td>90</td></tr>
    </tfoot>
  </table>
</div>
```

Tables

Quiz	Topic	Date	Grade
Q1	HTML	2/3/21	85
Q2	CSS	2/10/21	90
Q3	JavaScript	2/17/21	90
Average			90

3.3.7 Making tables responsive

In general it is a good practice to minimize the number of scrollbars shown at any one time in a browser screen. In browsers large amounts of content extends vertically beyond the height of the window, and then scrollbars allow you to access that extra content. Sometimes it is necessary to use additional scrollbars when appropriate such as tables that might be too wide to fit horizontally. Bootstrap provides tables that can show scrollbars when they don't fit in their parent window. To practice with ***Bootstrap responsive tables***, create a component with the code below and confirm it looks similar to image shown. Resize the window and confirm that the table shows a horizontal scroll bar when the screen is too small to fit the table comfortably.

src/Labs/Lab2/BootstrapTables.tsx

```
<div id="wd-css-responsive-tables">
  <h2>Responsive tables</h2>
  <div className="table-responsive">
    <table className="table">
      <thead>
        <tr><th>Very</th><th>long</th><th>set</th><th>of</th><th>columns</th>
            <th>Very</th><th>long</th><th>set</th><th>of</th><th>columns</th>
            <th>Very</th><th>long</th><th>set</th><th>of</th><th>columns</th>
        </tr>
      </thead>
      <tbody>
        <tr>
           <td>Very</td><td>long</td><td>set</td><td>of</td><td>columns</td>
           <td>Very</td><td>long</td><td>set</td><td>of</td><td>columns</td>
           <td>Very</td><td>long</td><td>set</td><td>of</td><td>columns</td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

3.3.8 Styling Lists

Another set of Bootstrap classes can make simple HTML lists look more presentable. The ***.list-group*** and ***.list-group-item*** classes can be applied to ***ul*** and ***li*** tags correspondingly to make list stand out. The ***.active*** class can be applied to an ***li*** tag to highlight it. To practice with ***Bootstrap lists***, create a component with the code below and confirm it looks similar to image shown.

src/Labs/Lab2/BootstrapLists.tsx

```
<div id="wd-css-styling-lists">
  <h2>Favorite movies</h2>
  <ul className="list-group">
    <li className="list-group-item active">Aliens</li>
    <li className="list-group-item">Terminator</li>
    <li className="list-group-item">Blade Runner</li>
    <li className="list-group-item">Lord of the Ring</li>
    <li className="list-group-item disabled">Star Wars</li>
  </ul>
</div>
```

Browser

Favorite movies

- Aliens
- Terminator
- Blade Runner
- Lord of the Ring
- Star Wars

3.3.9 Styling a List of Hyperlinks

The same ***.list-group*** and ***.list-group-item*** Bootstrap classes can be applied to ***div*** and ***a*** tags to implement a list of anchor links. To practice with ***Bootstrap hyperlink lists***, create a component with the code below and confirm it looks similar to image shown and that the links work.

src/Labs/Lab2/BootstrapLists.tsx

```
<div id="wd-css-hyperlink-list">
  <h3>Favorite books</h3>
  <div className="list-group">
    <a href="https://en.wikipedia.org/wiki/Dune_(novel)"
       className="list-group-item list-group-item-action active">
       Dune</a>
    <a href="https://en.wikipedia.org/wiki/The_Lord_of_the_Rings"
       className="list-group-item list-group-item-action">
       Lord of the Rings</a>
    <a href="https://en.wikipedia.org/wiki/The_Forever_War"
       className="list-group-item list-group-item-action">
       The Forever War</a>
    <a
href="https://en.wikipedia.org/wiki/2001:_A_Space_Odyssey_(novel)"
       className="list-group-item list-group-item-action">
       2001 A Space Odyssey</a>
    <a href="https://en.wikipedia.org/wiki/Ender%27s_Game"
       className="list-group-item list-group-item-action disabled">
       Ender's Game</a>
  </div>
</div>
```

Browser

Favorite books

- Dune
- Lord of the Rings
- The Forever War
- 2001 A Space Odyssey
- Ender's Game

3.3.10 Basic Form Styling

Bootstrap has tons of classes to style form elements especially to make them friendly for mobile Web applications. To practice with ***Bootstrap form classes***, create a component with the code below and confirm it renders similar to the image shown.

src/Labs/Lab2/BootstrapForms.tsx

```
<div id="wd-css-styling-forms">
  <h2>Forms</h2>
  <div className="mb-3">
    <label htmlFor="input1" className="form-label">
      Email address</label>
    <input type="email" className="form-control"
      id="input1" placeholder="name@example.com"/>
  </div>
  <div className="mb-3">
    <label htmlFor="textarea1" className="form-label">
      Example textarea</label>
    <textarea className="form-control" id="textarea1"
              rows={3}></textarea>
  </div></div>
```

Browser

Forms

Email address

name@example.com

Example textarea

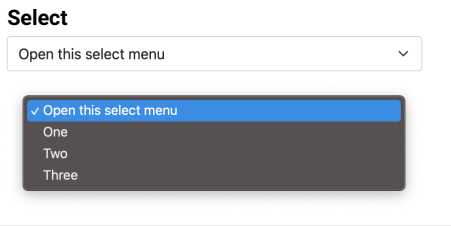
3.3.11 Styling Dropdowns

Dropdowns can also be styled professionally. To practice with ***Bootstrap dropdown styling***, create a component with the code below and confirm it looks similar to image shown.

src/Labs/Lab2/BootstrapForms.tsx

```
<div id="wd-css-styling-dropdowns">
  <h3>Dropdowns</h3>
  <select className="form-select">
     <option selected>Open this select menu</option>
     <option value="1">One</option>
     <option value="2">Two</option>
     <option value="3">Three</option>
  </select>
</div>
```

Browser



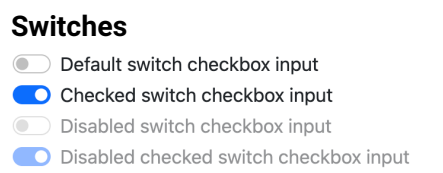
3.3.12 Styling Switches

Checkboxes can be styled to look like switches with Bootstrap classes ***.form-check*** and ***.form-switch***. To practice with ***Bootstrap form switches***, create a component with the code below and confirm it looks similar to image shown.

src/Labs/Lab2/BootstrapForms.tsx

```
<div id="wd-css-styling-switches">
  <h3>Switches</h3>
  <div className="form-check form-switch">
    <input className="form-check-input" type="checkbox"
           id="switch1" />
    <label className="form-check-label" htmlFor="switch1">
      Default switch checkbox input
    </label>
  </div>
  <div className="form-check form-switch">
    <input className="form-check-input" type="checkbox"
      id="switch2" checked />
    <label className="form-check-label" htmlFor="switch2">
      Checked switch checkbox input
    </label>
  </div>
  <div className="form-check form-switch">
    <input className="form-check-input" type="checkbox"
      id="switch3" disabled />
    <label className="form-check-label" htmlFor="switch3">
      Disabled switch checkbox input
    </label>
  </div>
  <div className="form-check form-switch">
    <input className="form-check-input" type="checkbox"
      id="switch4" checked disabled />
    <label className="form-check-label" htmlFor="switch4">
      Disabled checked switch checkbox input
    </label>
  </div>
</div>
```

Browser



3.3.13 Styling Range and Sliders

Range input fields render as sliders. To practice with ***Bootstrap sliders***, create a component with the code below and confirm it renders similar to image shown.

src/Labs/Lab2/BootstrapForms.tsx ***Browser***

```
<div id="wd-css-styling-range-and-sliders">
  <h3>Range</h3>
  <label htmlFor="range1" className="form-label">
    Example range
  </label>
  <input type="range" className="form-range"
    min="0" max="5" step="0.5" id="range1" />
</div>
```

Range

Example range

3.3.14 Styling Addons

Addons decorate input fields to give some context on the type and formate of the information to type in the input field. To practice with ***Bootstrap addons***, create a component with the code below and confirm it renders similar to image shown.

src/Labs/Lab2/BootstrapForms.tsx

```
<div id="wd-css-styling-addons">
  <h3>Addons</h3>
  <div className="input-group mb-3">
    <span className="input-group-text">$</span>
    <span className="input-group-text">0.00</span>
    <input type="text" className="form-control" />
  </div>
  <div className="input-group">
    <input type="text" className="form-control" />
    <span className="input-group-text">$</span>
    <span className="input-group-text">0.00</span>
  </div></div>
```

Browser

Addons

$ 0.00

$ 0.00

3.3.15 Responsive Forms

Forms can be configured to display either horizontally or vertically depending on the size of the containing element. To practice with Bootstrap responsive forms, create a component with the code below and confirm it renders similar to image shown. Resize the window to show how the form changes layout as the window resizes.

src/Labs/Lab2/BootstrapForms.tsx

```
<div id="wd-css-responsive-forms-1">
  <h3>Responsive forms</h3>
  <div className="mb-3 row">
    <label htmlFor="email1"
      className="col-sm-2 col-form-label">
      Email </label>
    <div className="col-sm-10">
      <input type="text" className="form-control"
        id="email1" value="email@example.com" />
    </div> </div>
  <div className="mb-3 row">
    <label htmlFor="password1"
          className="col-sm-2 col-form-label">
      Password </label>
    <div className="col-sm-10">
      <input type="password" id="password1"
            className="form-control" />
    </div> </div>
  <div className="mb-3 row">
    <label htmlFor="textarea2"
          className="col-sm-2 col-form-label">
      Bio </label>
    <div className="col-sm-10">
      <textarea className="form-control"
        id="textarea2" rows={3}></textarea>
    </div>
  </div>
</div>
```

Browser

Responsive forms

Email email@example.com

Password

Bio

Email

email@example.com

Password

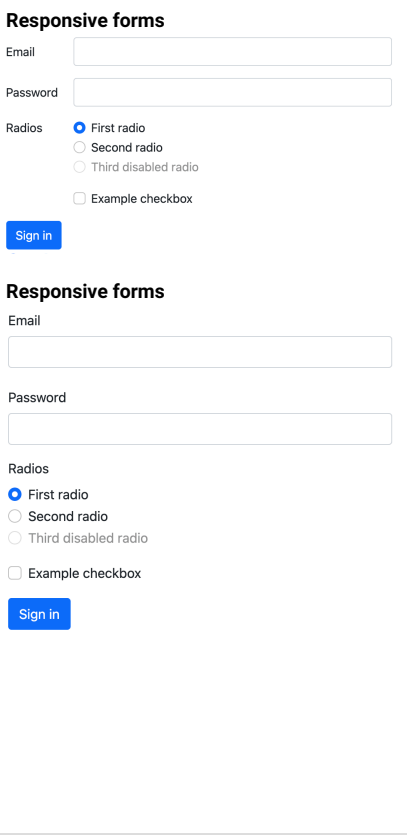
Bio

Here's another example. Create a component with the code below and confirm it renders similar to image shown. Resize the window to show how the form changes layout as the window resizes.

src/Labs/Lab2/BootstrapForms.tsx

```
<div id="wd-css-responsive-forms-2">
  <h3>Responsive forms</h3>
  <form>
    <div className="row mb-3">
      <label htmlFor="r1" className="col-sm-2 col-form-label">
        Email </label>
      <div className="col-sm-10">
        <input type="email" className="form-control" id="r1" />
      </div> </div>
    <div className="row mb-3">
      <label htmlFor="r2" className="col-sm-2 col-form-label">
        Password </label>
      <div className="col-sm-10">
        <input type="password" className="form-control" id="r2" />
      </div>
    </div>
    <fieldset className="row mb-3">
      <legend className="col-form-label col-sm-2 pt-0">
        Radios</legend>
      <div className="col-sm-10">
        <div className="form-check">
          <input className="form-check-input" type="radio"
            name="gridRadios" id="r3" value="option1" checked />
          <label className="form-check-label" htmlFor="r3">
            First radio </label> </div>
        <div className="form-check">
          <input className="form-check-input" type="radio"
            name="gridRadios" id="r4" value="option2" />
          <label className="form-check-label" htmlFor="r4">
            Second radio </label> </div>
        <div className="form-check disabled">
          <input className="form-check-input" type="radio"
            name="gridRadios" id="r5" value="option3" disabled />
          <label className="form-check-label" htmlFor="r5">
            Third disabled radio </label> </div>
      </div>
    </fieldset>
    <div className="row mb-3">
      <div className="col-sm-10 offset-sm-2">
        <div className="form-check">
          <input className="form-check-input" type="checkbox"
                 id="r6" />
          <label className="form-check-label" htmlFor="r6">
            Example checkbox </label> </div> </div>
    </div>
    <button type="submit" className="btn btn-primary">
      Sign in </button>
  </form>
</div>
```

Browser



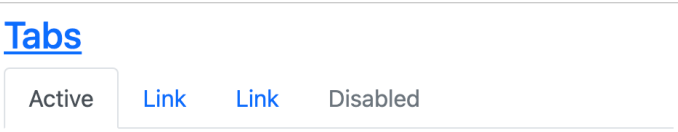
3.3.16 Navigating with Tabs

Bootstrap provides several common navigation widgets such as tabs, menus, and pills. Let's take a look at tabs first. To practice with Bootstrap tabs, create a component with the code below and confirm it renders similar to image shown.

src/Labs/Lab2/BootstrapNavigation.tsx

```
<div id="wd-css-navigating-with-tabs">
  <h2>Tabs</h2>
  <ul className="nav nav-tabs">
    <li className="nav-item">
        <a className="nav-link active" href="#">Active</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
    </li>
```

```
    <li className="nav-item">
        <a className="nav-link" href="#">Link</a>
    </li>
    <li className="nav-item">
        <a className="nav-link disabled" href="#">Disabled</a>
    </li>
  </ul>
</div>
```



3.3.17 Navigating with Pills

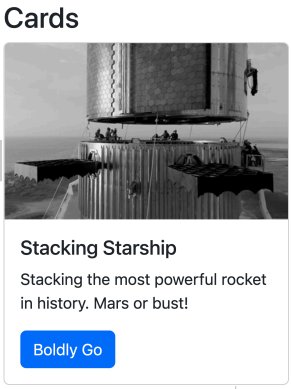
Pills are another navigation widget listing several links. Here's an example of using the ***Bootstrap Pills*** to refactor the ***TOC*** component and use ***JavaScript*** to highlight the ***Lab*** we are currently looking at. Confirm the ***TOC*** highlights the corresponding link as you navigate between the labs.

src/Labs/TOC.tsx

```
import { useLocation } from "react-router";
export default function TOC() {
  const { pathname } = useLocation();
  return (
    <ul className="nav nav-pills">
      <li className="nav-item">
        <a id="wd-a" href="#/Labs" className="nav-link">
          Labs
        </a>
      </li>
      <li className="nav-item">
        <a id="wd-a1" href="#/Labs/Lab1"
          className={`nav-link ${pathname.includes("Lab1") ? "active" : ""}`}>
          Lab 1
        </a>
      </li>
      <li className="nav-item">
        <a id="wd-a2" href="#/Labs/Lab2"
          className={`nav-link ${pathname.includes("Lab2") ? "active" : ""}`}>
          Lab 2
        </a>
      </li>
      <li className="nav-item">
        <a id="wd-a3" href="#/Labs/Lab3"
          className={`nav-link ${pathname.includes("Lab3") ? "active" : ""}`}>
          Lab 3
        </a>
      </li>
      <li className="nav-item">
        <a id="wd-k" href="#/Kanbas" className="nav-link">
          Kanbas
        </a>
      </li>
      <li className="nav-item">
        <a id="wd-k" href="https://github.com/jannunzi" className="nav-link">
          My GitHub
        </a>
      </li>
    </ul>
  );
}
```

3.3.18 Navigating with cards

Cards combine images, titles, paragraphs and buttons into a reusable component that can quickly summarize a topic. To practice with ***Bootstrap cards***, create a component with the code below and confirm it renders similar to image shown. Use an image of your own, or download one from my Flickr account and save it to ***public/images/stacked.jpg***.



src/Labs/Lab2/BootstrapNavigation.tsx

```
<div id="wd-css-navigating-with-cards">
  <h2>
    Cards
  </h2>
  <div className="card"
       style={{ width: "18rem" }}>
    <img src="images/stacked.jpg"
         className="card-img-top" />
    <div className="card-body">
      <h5 className="card-title">
          Stacking Starship
      </h5>
      <p className="card-text">
        Stacking the most powerful rocket in history. Mars or bust!
      </p>
      <a href="#" className="btn btn-primary">
        Boldly Go
      </a>
    </div>
  </div>
</div>
```

4 Styling Kanbas with CSS and Bootstrap

This section revisits the ***Kanbas*** screens implemented in previous assignments where the HTML rendered in its natural, unstyled, default look and feel. This section uses ***CSS*** and ***Bootstrap*** to layout and color the screens so they look more like the screenshots provided. Make an effort to style the HTML code to make the screens look as close as possible to their intended look and feel, but it is not required that the resulting screens are pixel perfect matches. Instead guidelines and requirements are provided which should adhered to. Previous assignments used ***table***, ***tr***, and ***td*** elements to layout screens horizontally. In general this is considered a bad practice and CSS alternatives are preferred. Remove the ***table*** elements in preparation of using ***CSS*** instead as shown below.

src/Kanbas/index.tsx

```
<div id="wd-kanbas">
  <table width="100%">
    <tr><td valign="top">
      <KanbasNavigation />
      </td><td valign="top">
    <div>
      <Routes>
        <Route path="/" element={<Navigate to="Account" />} />
        <Route path="/Account/*" element={<Account />} />
        <Route path="/Dashboard" element={<Dashboard />} />
        <Route path="/Courses/:cid/*" element={<Courses />} />
      </Routes>
    </div>
      </td></tr>
  </table>
</div>
```

Do the same for the ***Courses*** and ***Home*** screens replacing the ***table*** elements with ***div*** elements styled with ***Bootstrap flex classes*** and ***display classes***. Style the ***Course*** header as shown below and confirm the ***CoursesNavigation*** sidebar and the ***Course Status*** sidebar appears and hides as you resize the window. Add any missing imports as needed.

src/Kanbas/Courses/index.tsx

```
<div id="wd-courses">
  <h2 className="text-danger">
      <FaAlignJustify className="me-4 fs-4 mb-1" />
      Course 1234 </h2> <hr />
  <table><tr><td valign="top">
  <div className="d-flex">
     <div className="d-none d-md-block">
       <CoursesNavigation />
     </td><td valign="top">
     </div>
     <div className="flex-fill">
     <Routes>
       <Route path="Home" element={<Home />} />
       <Route path="Modules" element={<Modules />} />
       <Route path="Assignments" element={<Assignments />} />
       <Route path="Assignments/:aid" element={<AssignmentEditor />} />
     </Routes>
     </div></div>
     </td></tr>
  </table>
</div>
```

src/Kanbas/Courses/Home/index.tsx

```
<table id="wd-home"><tr><td valign="top">
<div className="d-flex" id="wd-home">
  <div className="flex-fill">
    <Modules />
  </div>
</td><td valign="top">
  <div className="d-none d-md-block">
    <CourseStatus />
  </div>
</div>
</td></tr></table>
```

4.1 Styling the Kanbas Navigation Sidebar

The ***Kanbas*** Web application has several screens implementing different features. Figure 4.1a bellow illustrates the ***Kanbas Navigation*** sidebar with the ***Dashboard*** link selected displaying a grid of courses that allow navigating to different courses. In This section demonstrates how to style the ***Kanbas Navigation*** as shown below and configure it to navigate to the ***Kanbas Dashboard*** and ***Courses Home*** screens.

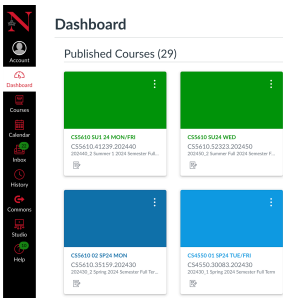


Figure 4.1a - Canvas Dashboard

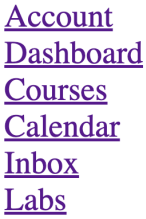


Figure 4.1b - Kanbas Navigation

Figure 4.1c - Our Kanbas Navigation

Previous assignments implemented the ***Kanbas Navigation*** sidebar in ***src/Kanbas/Navigation.tsx*** as rendered above in Figure 4.1b. Using ***Bootstrap's Links and Buttons*** to ***Navigation.tsx***, style the sidebar with ***CSS*** so that it looks more like the sidebar in the screenshot in Figures 4.1c. Use the code below as a guide to apply the ***Bootstrap*** classes and ***React Icons*** to render the ***Dashboard*** and ***Calendar*** as icons. Explore other icons for the rest of the links. The icons don't have to match the ones used in Figures 4.1a or c, but should reflect the meaning and intention of the link's text and target screen. The ***Northeastern*** logo can be found in the ***Brand Center*** website.

src/Kanbas/Navigation.tsx

```
import { AiOutlineDashboard } from "react-icons/ai";
import { IoCalendarOutline } from "react-icons/io5";
import { LiaBookSolid, LiaCogSolid } from "react-icons/lia";
import { FaInbox, FaRegCircleUser } from "react-icons/fa6";
export default function KanbasNavigation() {
  return (
    <div id="wd-kanbas-navigation" style={{ width: 120 }}                    {/* add list-group class     */}
```