Lab 3

Objectives

- Use a CSS flexbox to create a responsive webpage
- Write advanced selectors in CSS
- Write CSS using the **`@media`** rule

Associated Lecture Videos

- Flexbox introduction
- Flexbox example
- Selector patterns
- The @media rule

Additional References / Resources

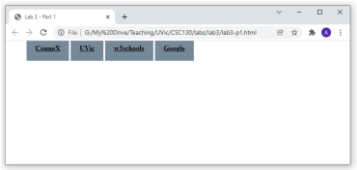
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/: a guide for flexbox specifically, including some really good illustrations and descriptions. The three exercises for lab3 are taken from here and modified slightly.
- Flexbox Froggy: a fun game to practice Flexbox: https://flexboxfroggy.com/
- w3schools @media page: https://www.w3schools.com/cssref/css3_pr_mediaquery.asp

SETUP – Download the necessary files

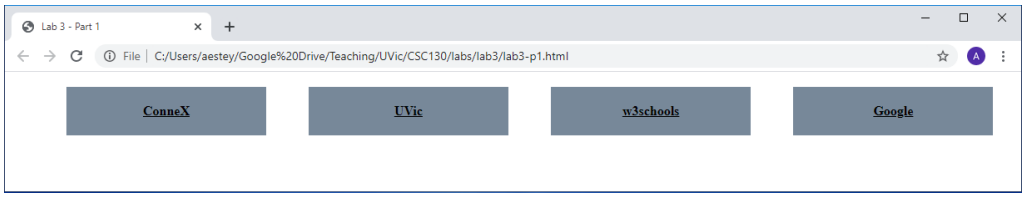
1. Make a lab 3 folder to save the files for this lab into.
2. Go to the BrightSpace course page, and download all of the lab files. There are three .html files and the three .css files. Save them all into your lab 3 folder.
    - Another option is to download the lab3-files.zip. It is a compressed file that contains all of the necessary lab files in one place.
3. You should now be able to open up the lab3-p1.html file in a text editor to edit the HTML, and also open it in a browser to view the web page.

Part I – Using flexbox [lab3-p1.html and lab3-p1.css]

1. Open up the lab3-p1.html and lab3-p1.css files in a text editor.
2. If you look at the lab3-p1.html file, you see there is not a lot of code in the HTML document. There is a single div, with four links inside of it. The important things to observe are that I have given the div containing the links a class name **`container`**, and the links inside the div all the class name **`item`**. This will allow us to select these different elements with CSS.
3. If you look at the corresponding CSS file lab3-p1.css, you'll see there is some styling for the **`item`** class and font color rules for links. At this point, if you open up the lab1-p1.html file in a browser it should look like the following image:



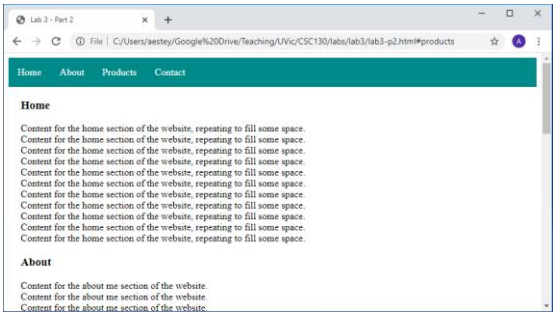
4. The goal is that to make the list of links dynamically change depending on the size of the browser window. To do so, you will need to create some style rules in the .css file for the **`container`** class:
    a. First of all, set the **`display`** property to **`flex`**, and have the items in the flex container displayed horizontally across the screen. Although it may do this by default, declare the **`flex-direction`** property so that it is set to **`row`**.
    b. Make it so that if the window is too small to fit all of the links, the links will drop down to a create another row. Set the **`flex-wrap`** property to **`wrap`**.
    c. Save the .css file and open/refresh your web page. Does it look different? Try combining the **`flex-direction`** and **`flex-wrap`** values into one rule using **`flex-flow`**, as shown in lecture.
    d. Update so the **`justify-content`** property so that the links are spaced out equals across the flex container.
    e. Save the .css file and open up your lab3-p1.html file in a browser. Try resize the window and see if the layout responds.



CHECK POINT 1

Part II – Predefined layouts for devices with different screen sizes

1. Open up the lab3-p2.html and lab3-p2.css files in a text editor.
2. If you look at the lab3-p2.html file, you'll see this example is similar to one of the lecture examples, where there are some links at the top of the document that jump to sections further down the page. The .css file adds left margins to the main content of the page, and some style rules for the flexbox navigation bar at the top of the page, with white font on a colored background:

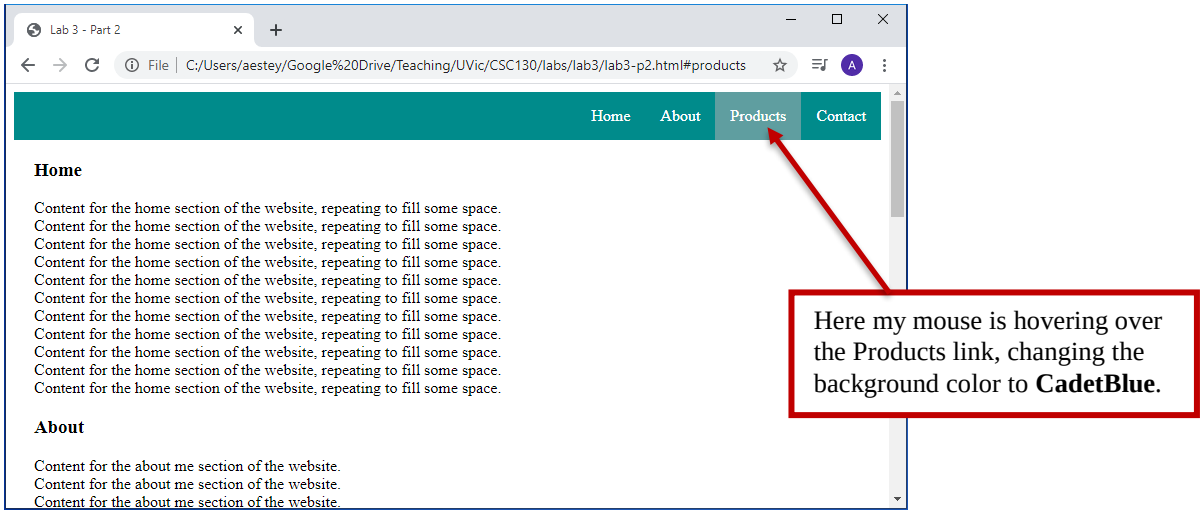


3. Add a style rule in the .css file so that when you put your mouse over one of the links in the navigation bar at the top, the **background-color** changes to **cadetblue**.

   Hint: the CSS rule to the right changes paragraph color to red when the mouse hovers over it. The difference is that you want to change the background color of links.

```
p:hover {
      color: red;
}
```

4. Update the flex container so that items are aligned on the right side of the screen:



5. Look in the lab3-p2.css file. You will notice there are two different CSS rules that change the color of links. Regular links are displayed in a dark green colour, as you can see with the four links in the Products section, and the link in the Contact section. On the other hand, the links in the navigation are coloured white. That is because the following CSS rule selects only links INSIDE the container class:

```
.container a {
     text-decoration: none;
     padding: 15px;
     color: white;
}
```

A CSS rule-set consists of a **selector** (what you want to apply the style to), and a **declaration block** (where the style rules are written). The **selector** can be written with a space between two elements, which allows us to select only elements on the right side of the space that are *inside* of the element on the left side of the space.

```
li {
     color: red
}
```

```
ol li {
     color: red;
}
```

For example, the rule-set above on the left applies to all list item (`li`) elements.

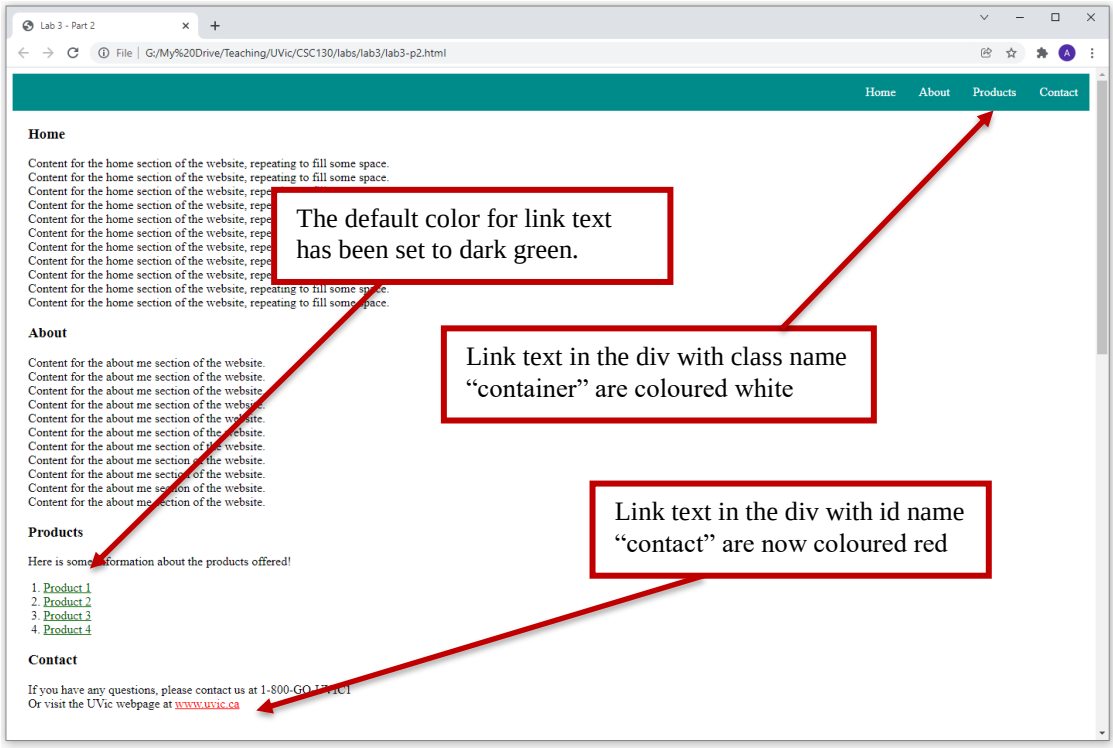
The selector for the rule-set above on the right has a space between two elements, which means that only list items inside of an ordered list would be coloured red.

What does the following rule-set do?

```
table img {
     border: solid 2px black;
}
```

Answer: It selects only images that are inside a table, and gives those images a solid black border with a width of 2px.

Write a CSS rule-set that selects only links that are ***inside*** the div with id = “contact” and changes the color of the link text to red. The lab3-p2.html page should then look like the following:

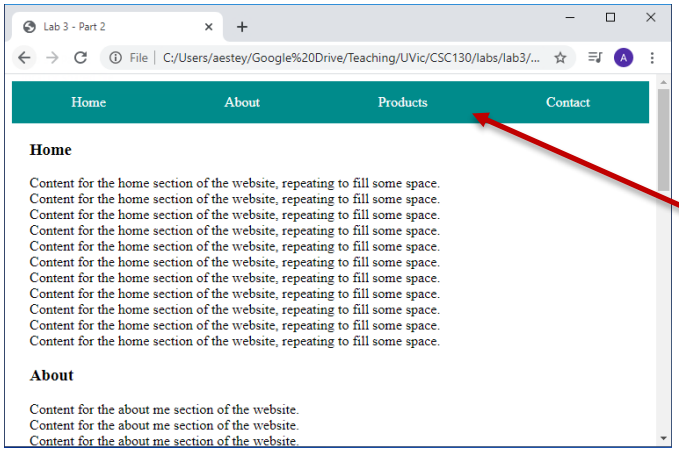


CHECK POINT 2

6. Now, you will use the @media rule shown in lecture to apply rules based on the type of device viewing the web page. For this lab, we will make one set of rules for all devices with a maximum width of 800px, and another for devices with a maximum width of 600px. To do this the syntax in the .css file is:

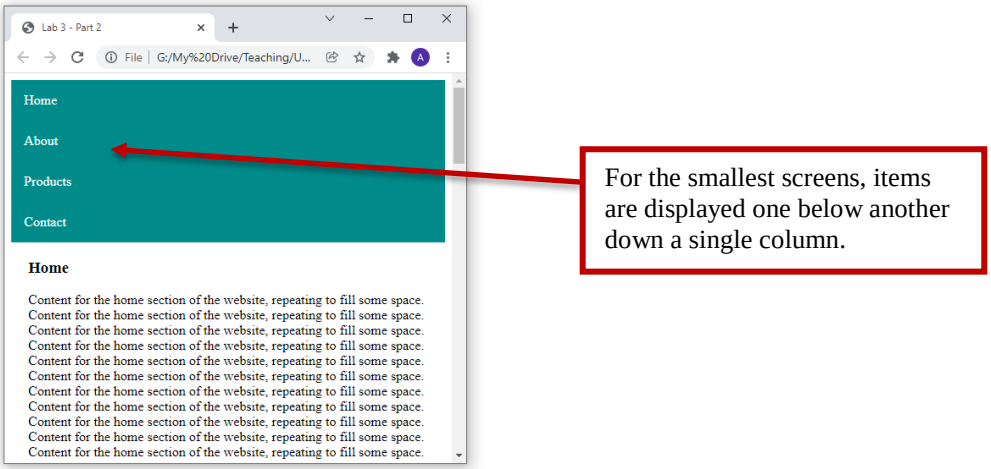
```
@media all and (max-width: 800px) {
      ***PUT CSS RULES IN HERE***
}

@media all and (max-width: 600px) {
      ***PUT CSS RULES IN HERE***
}
```

   a. Within the declaration block for 800px widths, update the layout of the flexbox container so that the links at the top are equally spaced across the navigation bar *(Hint: we did this in Part I)*. When complete, save your .css and refresh your web page to see if resizing the window has any effect.

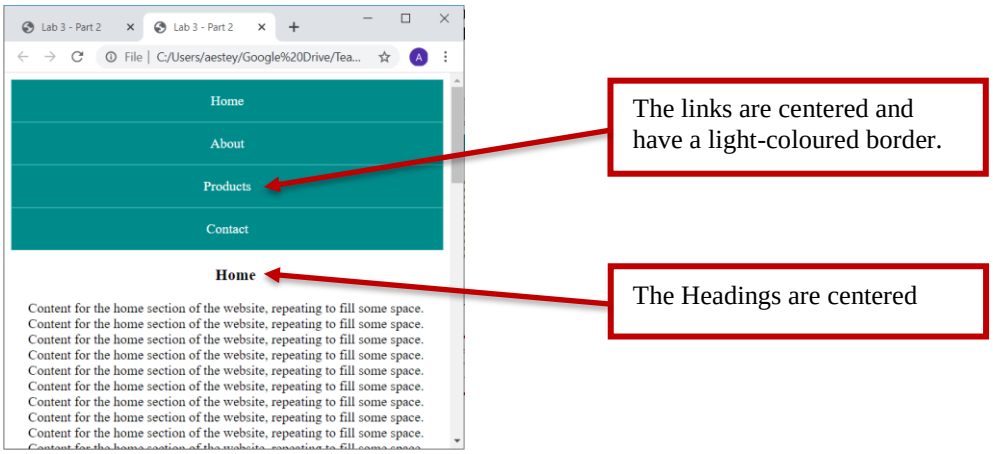


For slightly smaller screens, the items in the container are evenly spaced across the top.

b. Within the 600px declaration block, update the layout of the flexbox containers so that it displays the items down a column instead of along a row. *(Hint: we are changing the direction the items are displayed in)*.



c. Now, within the 600px declaration block, add rules so that headings (h3) are centered, and the links inside the flexbox container are centered and have a border *(the border in the image below has color RGBA(200,255,255,0.25).*



d. You should now be able to resize the width of your browser window and the layout of content on the page will update dynamically.

CHECK POINT 3 – MAIN PART OF LAB COMPLETE

Extra Practice (BONUS – you do not need to complete this as part of the lab)

I have created a flexbox template that you might find useful to use on your projects

1. Open up the lab3-p3.html and lab3-p3.css files in a text editor.
2. If you look at the lab3-p3.html file, you'll see this example is similar to one of the lecture examples where I had a number of side bars positioned to the left and right side of the screen. This example uses a flexbox container to modify how the side bars are displayed based on the width of the screen.
3. You do not need to modify the .html or .css files in any way, instead we will upload these files so they can be viewed online by anyone.
4. I have added some **comments** to the .css file to help explain some of the style rules I declared. Comments are enclosed in /* and */ and are meant to be read by other programmers, without affecting the execution of the code in any way. Depending on how much time you have, feel free to tweak the style rules in the .css file, but only do so after completing the rest of the steps in Part III.