Activity 1 - Flexbox implementation

Activity 1

- This is an **individual** activity and will **not** be graded.
- Time: **120 minutes**

Now that you've learnt a bit about what a flexbox is, let's create one!

Instructions

1. Create a new Atom project called flexbox to house all the example files for this topic. Then, create a file called flexbox.html and add the following markup:

```
<!DOCTYPE html>
<html lang='en'>
  <head>
    <meta charset='UTF-8'/>
    <title>Some Web Page</title>
    <link rel='stylesheet' href='style.css'/>
  </head>
  <body>
    <div class='menu-container'>
    <div class='menu'>
    <div class='date'>Aug 14, 2016</div>
    <div class='signup'>Sign Up</div>
    <div class='login'>Login</div>
    </div>
```

```
    </div>
  </body>
</html>
```

NB: In the code above, this line of code <link rel='stylesheet' href='style.css'/> makes reference to a css file called style.css . You need to create this css file next

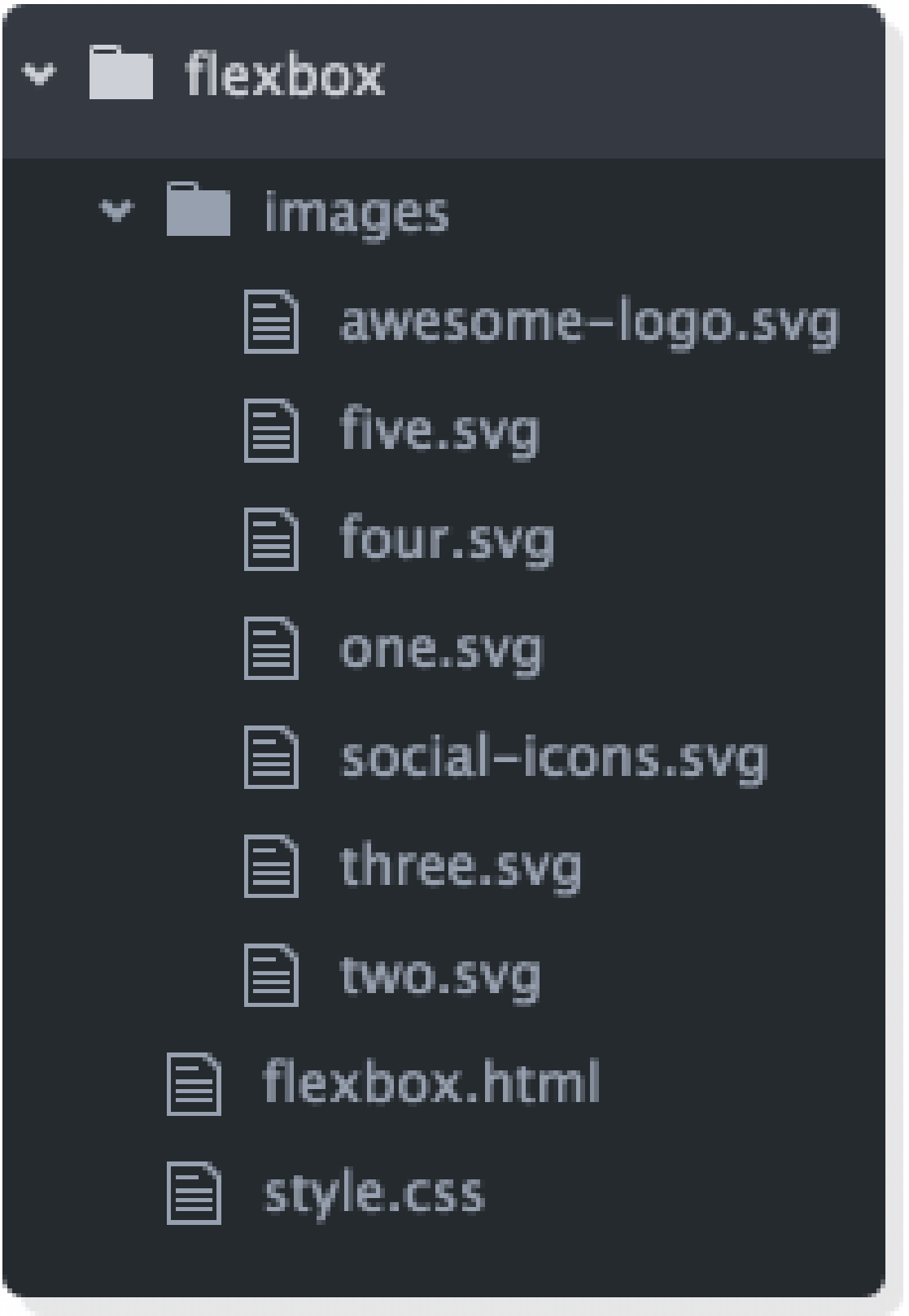
2. Create the corresponding style.css stylesheet, using the following css code:

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  }

.menu-container {
  color: #fff;
  background-color: #5995DA;  /* Blue */
  padding: 20px 0;
  }

.menu {
  border: 1px solid #fff;  /* For debugging */
  width: 900px;
  }
```

3. Finally, download these images (/assets/courseware/v1/4497057de3d05866452065ac4bbf873b/asset-v1:CITI+CAP001+2019+type@asset+block/flexbox-images-449705.zip) to use in your example web page. Unzip them into your flexbox project, keeping the parent images directory. Your project should look like this before moving on:



4. Add the following line to your .menu-container rule to turn it into a flex container:

```
.menu-container {
   /* ... */
   display: flex;
   }
```

NB: The above code tells the browser that instead of using the normal box model, it must use the flexbox model.

5. Define the horizontal alignment of the box's items, using the justify-content property. You can use this property to center your .menu, as shown here:

```
menu-container {
 /* ... */
   display: flex;
   justify-content: center;    /* Add this */
   }
```

NB: This has the same effect as adding a margin: 0 auto declaration to the .menu element.

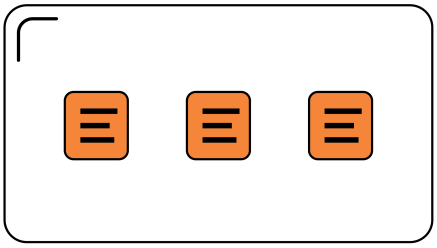
Other values for justifying content are:

- center
- flex-start
- flex-end
- space-around
- space-between

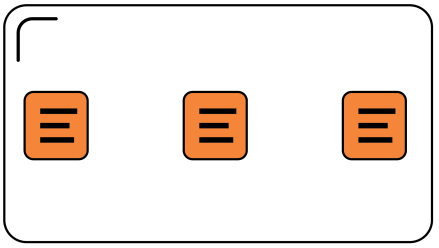
6. Change your .menu rule to match the following:

```
.menu {
   border: 1px solid #fff;
   width: 900px;
   display: flex;
   justify-content: space-around;
   }
```

NB: The justify-content property also lets you distribute items equally inside a container, as shown in the image below:

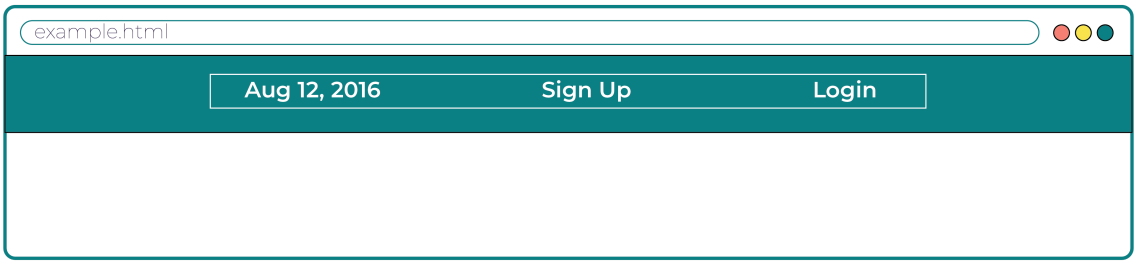


Space-around



Space-between

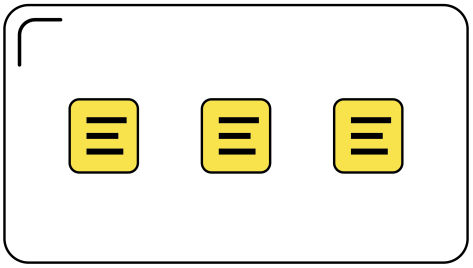
7. You should see something like this:



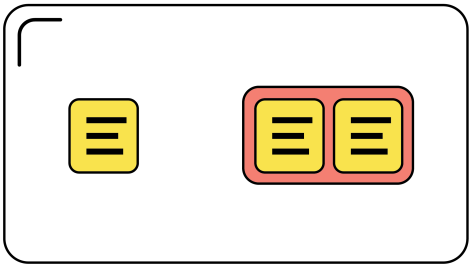
8. You want both the Sign-Up and Login links to be on the right side of the page, as in the screenshot below. To do this, place them in another , using the code below:

```
<div class='menu'>
  <div class='date'>Aug 14, 2016</div>
    <div class='links'>
    <div class='signup'>Sign Up</div>    <!-- This is nested now -->
    <div class='login'>Login</div>      <!-- This one too! -->
  </div>
</div>
```

NB: Flex containers only know how to position elements that are one level deep (i.e., their child elements). They do not consider what's inside their flex items. This means that grouping flex items is another weapon in your layout-creation arsenal. Wrapping a bunch of items in an extra results in a totally different web page.



No grouping
(3 flex items)



Grouped items
(2 flex items)

For example, instead of having three items, your .menu flex container now has only two (.date and .links). Under the existing space-between behaviour, they'll snap to the left and right side of the page.

9. Next, add a new rule to your styles.css file that turns the .links element into a flex container:

```
.links {
   border: 1px solid #fff;  /* For debugging */
   display: flex;
   justify-content: flex-end;
   }

.login {
   margin-left: 20px;
   }
```

10. Now add the following markup to flexbox.html after the .menu-container element:

```
<div class='header-container'>
   <div class='header'>
      <div class='subscribe'>Subscribe &#9662;</div>
      <div class='logo'><img src='images/awesome-logo.svg'/></div>
      <div class='social'><img src='images/social-icons.svg'/></div>
   </div>
</div>
```

NB: So far, you've been manipulating horizontal alignment, but flex containers can also define the vertical alignment of their items. The code above does the vertical alignment.

11. Next, add some base styles to get it aligned with our .menu element:

```
.header-container {
   color: #5995DA;
   background-color: #D6E9FE;
   display: flex;
   justify-content: center;
   }

.header {
   width: 900px;
   height: 300px;
   display: flex;
   justify-content: space-between;
   }

.header {
   /* ... */
   align-items: center;  /* Add this */
   }
```

12. Save your work

Daily Notes - Flexbox implementation

Please enter your daily notes here

Your response must be between 1 and 100000 words.

Submit **Save**


CiTi
CLICK NEXT TO CONTINUE