4 – Layout in CSS, Icons and Social Medial Icons, Navigation bars

4.1 Layout in CSS

In this chapter we are going to look at how to control where each element sits on a page and how to create attractive page layouts.

Screen size

Different visitors to your site will have different sized screens that show different amounts of information, so your design needs to be able to work on a range of different sized screens.

Dell XPS 13 Laptop
Display: **13.3 inches across screen**
Display resolution: **1920 × 1080 pixels**

Apple MacBook
Display: **12 inches across screen**
Display resolution: **2304 × 1440 pixels**

Building blocks

CSS treats each HTML element as if it is in its own box. This box will either be a block-level box or an inline box.

If one block-level element sits inside another block-level element then the outer box is known as the containing or parent element.

It is common to group a number of elements together inside a (or other block-level) element.

Static Layouts: Fixed Width

Fixed width layout designs do not change size as the user increases or decreases the size of their app window. Measurements tend to be given in pixels.

Advantages

- Pixel values are accurate at controlling size and positioning of elements.

- The designer has far greater control over the appearance and position of items on the page than with liquid layouts.
- You can control the lengths of lines of text regardless of the size of the user's window.
- The size of an image will always remain the same relative to the rest of the page.

Disadvantages

You can end up with big gaps around the edge of a page.

- If the user's screen is a much higher resolution than the designer's screen, the page can look smaller and text can be harder to read.
- If a user increases font sizes, text might not fit into the allotted spaces.
- The design works best on devices that have a site or resolution similar to that of desktop or laptop computers.
- The page will often take up more vertical space than a liquid layout with the same content.

Dynamic Layouts: Liquid Layout

The liquid layout uses percentages to specify the width of each box so that the design will stretch to fit the size of the screen.

Advantages

- Pages expand to fill the entire app window so there are no spaces around the page on a large screen.
- If the user has a small window, the page can contract to fit it without the user having to scroll to the side.
- The design is tolerant of users setting font sizes larger than the designer intended (because the page can stretch).

Disadvantages

- If you do not control the width of sections of the page then the design can look very different than you intended, with unexpected gaps around certain elements or items squashed together.
- If the user has a wide app window, lines of text can become very long, which makes them harder to read.
- If the user has a very narrow app window, words may be squashed and you can end up with few words on each line.
- If a fixed width item (such as an image) is in a box that is too small to hold it (because the user has made the window smaller) the image can overflow over the text.

Floating Elements. `float` property in CSS

The float property allows you to take an element in normal flow and place it as far to the left or right of the containing element as possible.

Anything else that sits inside the containing element will flow around the element that is floated.

When you use the float property, you should also use the width property to indicate how wide the floated element should be. If you do not, results can be inconsistent but the box is likely to take up the full width of the containing element (just like it would in normal flow).

The CSS `float` property specifies how an element should float. The `float` property can have one of the following values:

- **`left`** - The element floats to the left of its container

`right`- The element floats to the right of its container**`none`** - The element does not float (will be displayed just where it occurs in the text). This is default**`inherit`** - The element inherits the float value of its parent**tResponsive effect**

Responsive app web is about using html and css to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

The **`box-sizing`** property can make building CSS layouts easier and a lot more intuitive. **`box-sizing: border-box;`** , we can change the **box** model to what was once the "quirky" way, where an element's specified width and height aren't affected by padding or borders **.**

`<div>` division tag

The **`<div>`** tag defines a division or a section in an HTML document and is very often used together with CSS, to layout a web page. The **`<div>`** element is often used as a container for other HTML elements to style them with CSS or to perform certain tasks with JavaScript.

Exercise) Create a layout using `<div>` to save four images

To create the layout, first we create a container to hold the header and the four images. We can create this container using `<div>` and name it `container`

```
<div class="container">

  </div>
```

Once the container is set, we are going to create five more division within the container; one division for the header and four for the images:

```
<div class="container">
      <div class="header">NEW YORK CITY</div>
      <div class="images">IMAGE 1</div>
      <div class="images">IMAGE 2</div>
      <div class="images">IMAGE 3</div>
      <div class="images">IMAGE 4</div>
  </div>
```

A complete HTML file looks as the following:

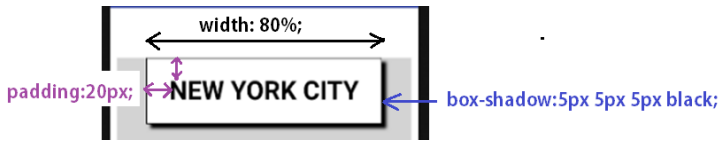
html file

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="index.css" rel="stylesheet" type="text/css"/>
    <title>Layout Design</title>
  </head>
  <body>
    <div class="container">
       <div class="header">NEW YORK CITY</div>
       <div class="images">IMAGE 1</div>
       <div class="images">IMAGE 2</div>
       <div class="images">IMAGE 3</div>
       <div class="images">IMAGE 4</div>
  </div>
  </body>
</html>
```

Now in the CSS file, we can set the background-color, width, and height to the container first:

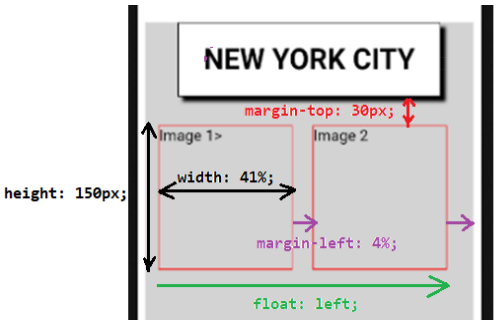
```
.container{
  background-color: lightgray;
  width: 100%;
  height: 500px;
}
```

After it, we can set the css to the header as the following:




```
.header{
  width: 80%;
  margin: auto;
  border: solid black;
  box-shadow: 5px 5px 5px black;
  padding: 20px;
  font-size: 30px;
  font-weight: 600;
  text-align: center;
  background-color: white;
}
```

Once the header is set, we can set the css to the **`images`** container. For this, we can add a border to the images container to use it as reference. This border can be erased after the layout is set.




```
.images{
       border: solid red;
       height: 150px;
       width: 41%;
       float: left;
       margin-left: 2%;
       margin-top: 30px;
       margin-left: 4%;
        }
```

The complete CSS file looks as the following:

index.css

```
*{ box-sizing: border-box; }
.container{
  background-color: lightgray;
  width: 100%;
  height: 500px;
}
.header{
  width: 80%;
  margin: auto;
  border: solid black;
  box-shadow: 5px 5px 5px black;
  padding: 20px;
  font-size: 30px;
  font-weight: 600;
  text-align: center;
  background-color: white;
}
.images{
  border: solid red;
  height: 150px;
  width: 41%;
  float: left;
  margin-left: 2%;
  margin-top: 30px;
  margin-left: 4%;
}
```

4.2 CSS styling images

The images can fit in a container. For the image to fully fit within the `<div>`, the image **`width`** and **`height`** must be set to 100%

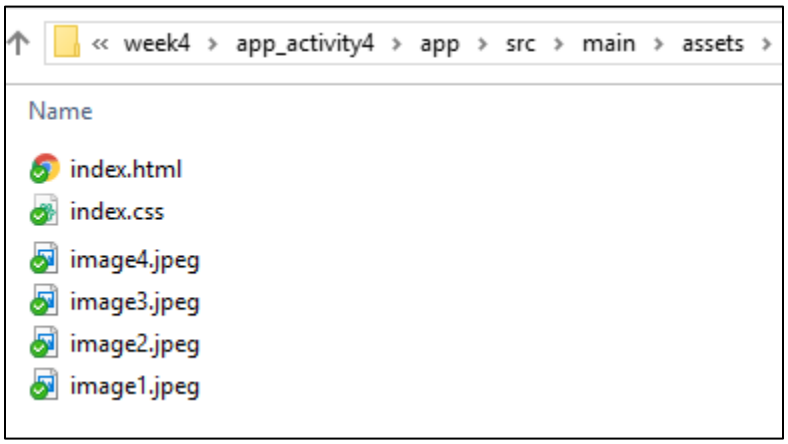
Exercise) Insert four different images to the previous code.

We can download four images and save them in assets folder. If we want to organize our assets folder and put all the images in one folder name *images*:

If the images are inside of the *images* folder, the code in HTML should indicate the location of the folder as the following:

```
<div class="images"><img src="images/image1.jpeg" /></div>
```

If the images are in the same folder that the HTML file:



the code in HTML should just indicate the image name:

```
<div class="images"><img src="image1.jpeg" /></div>
```

For this activity, we are going to organize all images inside the *images* folder. The complete HTML file

html file

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="index.css" rel="stylesheet" type="text/css"/>
    <title>Layout Design</title>
  </head>
  <body>
    <div class="container">
      <div class="header">NEW YORK CITY</div>
      <div class="images"><img src="images/image1.jpeg" /></div>
      <div class="images"><img src="images/image2.jpeg" /></div>
      <div class="images"><img src="images/image3.jpeg" /></div>
      <div class="images"><img src="images/image4.jpeg" /></div>
    </div>
  </body>
</html>
```

Now in css, for the images to sit within the box that contain them, we have to write a css code that specifies that images will sit 100% to the height and width of its container:

```
img{
  width: 100%;
  height: 100%;
}
```

The complete css file looks as the following:

index.css

```
*{ box-sizing: border-box; }
img{   width: 100%;   height: 100%;
}
.container{
  background-color: lightgray;
  width: 100%;
  height: 800px;
}
.header{
  width: 80%;
  margin: auto;
  border: solid black;
  box-shadow: 5px 5px 5px black;
  padding: 20px;
  font-size: 30px;
  font-weight: 600;
  text-align: center;
  background-color: white;
}
.images{
  height: 300px;
  width: 41%;
  float: left;
  margin-left: 2%;
  margin-top: 30px;
  margin-left: 4%;
}
```

Image format

To create rounded and circled images, we can use the `border-radius` property.

Rounded Images

```
img1 {
  border-radius: 50px;
}
```

Circled Image

```
img2 {
  border-radius: 50%;
}
```

To create a border or shadow to an image, we can use the `border` property to create thumbnail images.

Border image

```
img3 {
  border: 1px solid green;
  border-radius: 10px;
  padding: 20px;
}
```

Shadow image

```
img4  {
  box-shadow: 0 0 5px 20px lightblue;
}
```

Exercise) Using the previous code, add the css code to round the first image, make the second image to be a cicle, add border to the third image, and box-shadow to the fourth image.

To complete this activity, we can add the css code within the HTML code using `style` as the following:

```
<div class="images"><img src="images/image1.jpeg" style="border-radius: 8px;"/></div>
```

If we want to complete this activity using external css code, we can add a class name to each image and then add the css code in the css file. For this case, the HTML code will be:

```
<div class="images "><img src="images/image1.jpeg" class="img1" /></div>
```

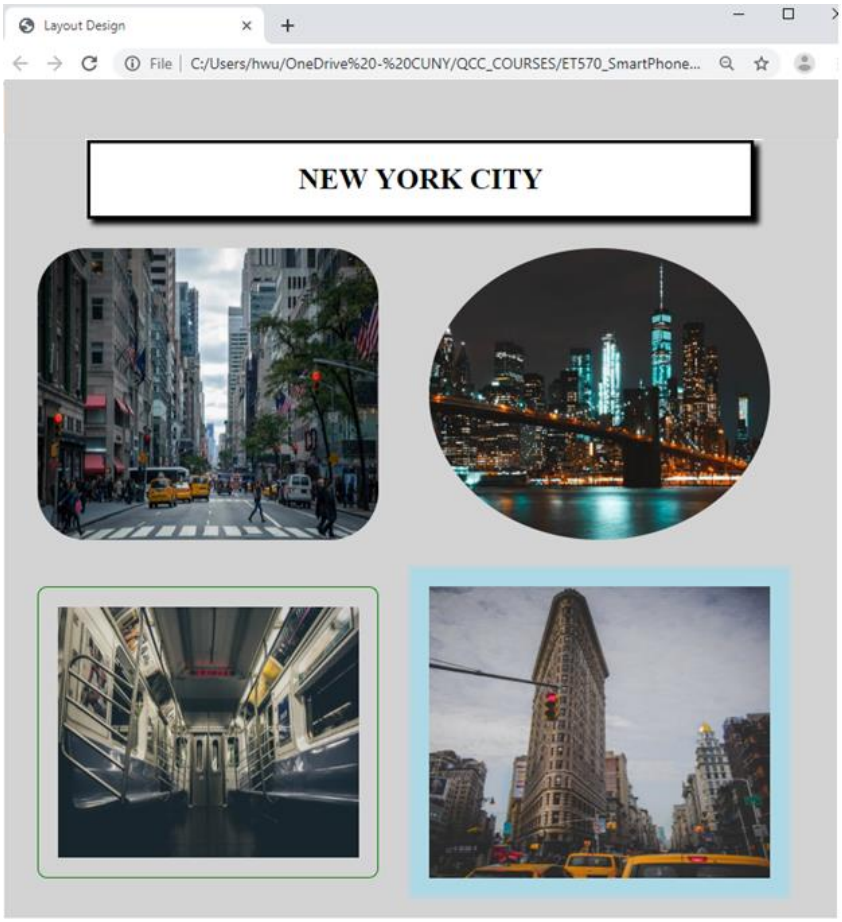
CSS code:

```
.img1{border-radius: 8px;}
```

You can also use an internal CSS code in the HTML file. The complete HTML code using internal css code is as the following:

html file

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="index.css" rel="stylesheet" type="text/css"/>
    <title>Layout Design</title>
  </head>
  <body>
    <div class="container">
      <div class="images"><img src="images/image1.jpeg" style="border-radius: 8px;"/></div>
      <div class="images"><img src="images/image2.jpeg" style="border-radius: 50%;"/></div>
      <div class="images"><img src="images/image3.jpeg" style="border: 1px solid green;
            border-radius: 6px; padding: 10px;" /> </div>
      <div class="images"><img src="images/image4.jpeg"
            style="box-shadow:0 0 5px 10px lightblue"/></div>
   </div>
  </body>
```



4.3 Navigation Bars

A navigation bar needs standard HTML as a base, which can be built using the `<ul>` and `<li>` elements or `<div>` element. To create a navigation bar using `<ul>` and `<li>`

Vertical Navigation Bars

To create a basic vertical navigation bar using **`<ul>,`** we can list each hyperlink using **`<a>`** tag as the following:

```
<ul class="navbar">
  <li><a href="#home">Home</a></li>
  <li><a href="#nav">Tabs</a></li>
  <li><a href="#flex">Flex-Wrap</a></li>
  <li><a href="#social">Social Media</a></li>
</ul>
```

To take off the bullet of each list, we can set the **`list-style-type`** to **`none`** of the `<ul>`:

```
.navbar{
  list-style-type: none;
}
```

To make the vertical navigation bar with a purple background-color, we can write CSS codes to make changes to the hyperlinks in the list item. Also, **`display: block;`** property sets the list in a block, column, format.

```
.navbar li a {
  display: block;
  padding: 8px 16px;
  color: white;
  background-color: purple;
  width: 50%;
  text-decoration: none;
}
```

After it, we can change the background color of the links when the user clicks on them:

```
.navbar li a:hover {
  background-color: gray;
  color: white;
}
```

The HTML and CSS code looks as the following:

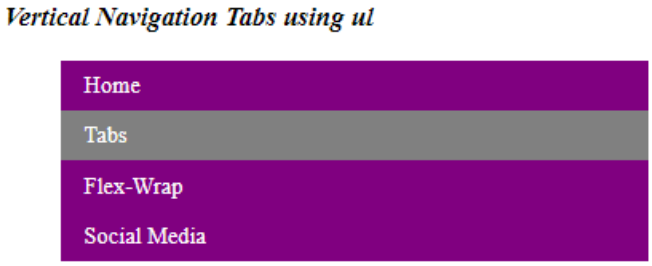
html file

```
<h2 >NAVIGATION TABS</h2>
<hr/>
<i><h3 >Vertical Navigation Tabs using ul</h3></i>
<ul class="navbar">
  <li><a href="#home">Home</a></li>
  <li><a href="#nav">Tabs</a></li>
  <li><a href="#flex">Flex-Wrap</a></li>
  <li><a href="#social">Social Media</a></li>
</ul></html>
```

css file

```
/* -- navigation bars --*/
h2{
  width: 90%;
  text-align: center;
  margin-top: 70px;
}
.navbar{
  list-style-type: none;
}
.navbar li a {
  display: block;
  padding: 8px 16px;
  color: white;
  background-color: purple;
  width: 50%;
  text-decoration: none;
}
/* Change the link color when is active*/
.navbar li a:hover {
  background-color: gray;
  color: white;
}
```

The output will look as the following:



Horizontal Navigation Bars

There are two ways to create a horizontal navigation bar using **`inline`** or **`floating`** list items in css once the list is created in HTML

Exercise) Create a horizontal navigation bar using **`<div>`** tag

To create the navigation bar using **`<div>,`** we create the `<div>` as the container of each hyperlink:

```
<div class="nav_div">
  <a href="#home">Home</a>
  <a href="#nav">Tabs</a>
  <a href="#flex">Flex-Wrap</a>
  <a href="#social">Social Media</a>
</div>
```

Once we have the HTML code set, in CSS, we can set the hyperlink container with background-color, the width, the height, and the font-size:

```
.nav_div{
    background-color: black;
    width: 100%;
    height: 50px;
    font-size: 16px;
}
```

After it, we can set hyperlink with left border, height, width, font color, center the text, and add the padding. We also set the hyperlink to float from the left to right without underline:

```
.nav_div a {
  border-left: ridge 2px white;
  height: 100%;
  width: 25%;
  color: white;
  text-align: center;
  padding: 10px;
  float: left;
  text-decoration: none;
}
```

After it, we can change the background color of the links when the user clicks on them:

```
.navbar li a:hover {
  background-color: orange;
}
```

The HTML and CSS code looks as the following:

html file

```
<i><h3 >Horizontal navigation Tabs using div tag</h3></i>
<div class="nav_div">
  <a href="#home">Home</a>
  <a href="#nav">Tabs</a>
  <a href="#flex">Flex-Wrap</a>
  <a href="#social">Social Media</a>
</div>
```

css file

```
/* navigation bar using div tag */
.nav_div{
    background-color: black;
    width: 100%;
    height: 50px;
    font-size: 16px;
}
.nav_div a {
  float: left;
  border-left: ridge 2px white;
  height: 100%;
  width: 25%;
  color: white;
  text-align: center;
  padding: 10px;
  text-decoration: none;
}
.nav_div a:hover{
  background-color: orange;
}
```

The output will look as the following:

Horizontal navigation Tabs using div tag

Fixed navigation bar

Fixed navigation bar happens when the navigation bar remain at the top or bottom of the app, even when the user moves the page. For this, you can use the CSS property `position: sticky;` to the element that contains the hyperlinks:

```
position: sticky;
position: fixed;
top: 0;
```

`position:fixed;` and **`top: 0;`** will make the element to have a position fixed to the top of the app.

Exercise) Create a navigation bar that will stick on the top of the app.

We can create the hyperlink using **`<div>`**

```
<div class="nav_top">
  <a href="#home">Home</a>
  <a href="#nav">Tabs</a>
  <a href="#flex">Flex-Wrap</a>
  <a href="#social">Social Media</a>
</div>
```

Once we have the HTML code set, in CSS, we can set the hyperlink container with background-color, the width, the height, the font-size, and the position:

```
.nav_top{
    background-color: orange;
    width: 100%;
    height: 50px;
    font-size: 16px;
    position: sticky;
    position: fixed;
    top: 0;
}
```

After it, we can set hyperlink with left border, height, font color, center the text, and add the padding. We also set the hyperlink to float from the left to right without underline:

```
.nav_top a {
  float: left;
  color: white;
  text-align: center;
  text-decoration: none;
  padding: 12px;
  height: 100%;
}
```

After it, we can change the background color of the links when the user clicks on them:

```
.navbar li a:hover {
  background-color: green;
}
```

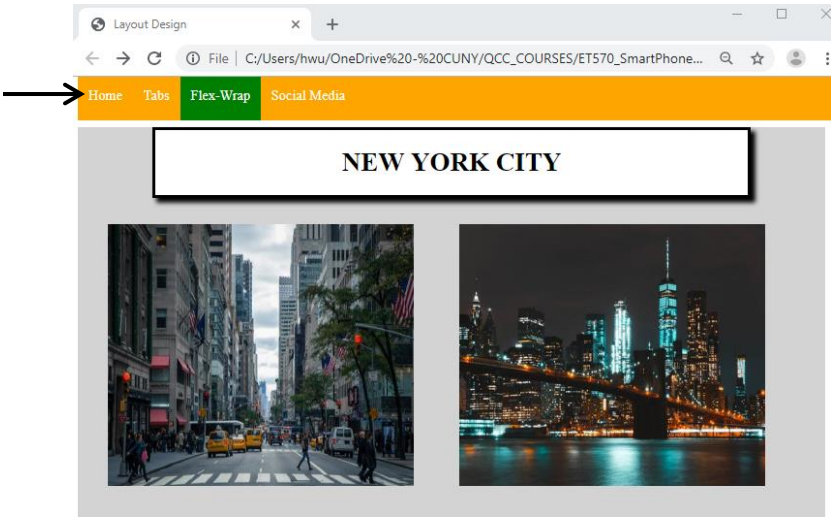
The HTML and CSS code looks as the following:

html file

```
<i><h3>Navigation Tabs sticky on the top</h3></i>
<div class="nav_top">
  <a href="#home">Home</a>
  <a href="#nav">Tabs</a>
  <a href="#flex">Flex-Wrap</a>
  <a href="#social">Social Media</a>
</div>
```

css file

```
/* navigation stick on the top */
.nav_top{
    background-color: orange;
    width: 100%;
    height: 50px;
    font-size: 16px;
    position: sticky;
    position: fixed;
    top: 0;
}
.nav_top a {
  float: left;
  color: white;
  text-align: center;
  text-decoration: none;
  padding: 12px;
  height: 100%;
}
.nav_top a:hover{
  background-color: green;
}
```



If you need the navigation bar stick in the bottom, you can change **`top: 0;`** to **`bottom: 0;`**

4.4 Extra Markup - Adding Media Logos – contact information (flex-container)

CSS flexbox layout

Flexible container layout makes it easier to design flexible responsive layout structure without using float or positioning.

The flex-wrap Property

The `flex-wrap` property in css specifies whether the flex items should wrap or not.

Exercise) The example below have 6 flex items, to better demonstrate the `flex-wrap` property. For this first try we can set the `flex-wrap` container with **`width: 40%;`**

html file

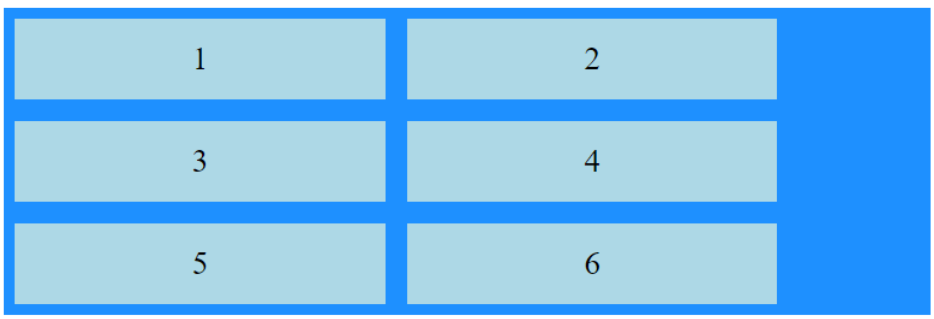
```
<h2>FLEX CONTAINER </h2>
<i><h3 id="about">The flex-wrap property</h3></i>
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```

css file

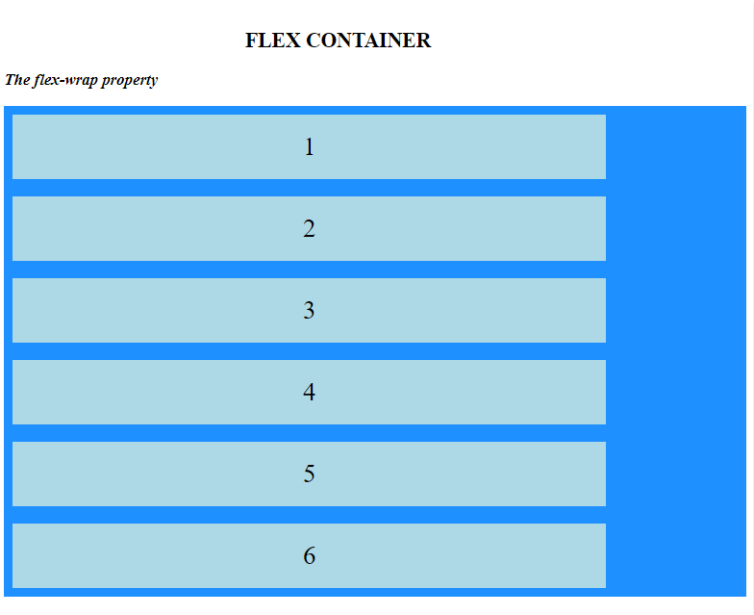
```
.flex-container {
  display: flex;
  flex-wrap: wrap;
  background-color: DodgerBlue;
}
.flex-container > div {
  background-color: lightblue;
  width: 40%;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

FLEX CONTAINER

The flex-wrap property



Now we can set the **`width: 80%;`** and see the changes:



4.5 Logo and social media logo

There are many logos on the internet that we can use in the app you are creating. One of the website that you can use to download logos and use in your app is https://www.iconfinder.com/

To add the social media information in your app as part of the contact you can use `<a>`:

HTML

```
<a target="_blank" title="follow me on Twitter"
href="https://www.twitter.com/PLACEHOLDER"><img alt="follow me on Twitter"
src="twitter.png" border=0></a>
```

The PLACEHOLDER space should go the Twitter username, for example, if your Twitter username was '`@thecakeshop`', you would replace '`www.twitter.com/PLACEHOLDER` with '`www.twitter.com/thecakeshop`', ensuring you don't include the @ symbol so that the button directs to the right page.

Also, `flex-wrap` can also use create the social medial container

Exercise) Download three icon images and use flex-wrap to create a social media contacts

html file

```
<h2>SOCIAL MEDIA </h2>
<i><h3 "social">Social Media Icons Using flex-wrap</h3></i>
<div class="flex-media">
      <div><a target="_blank" title="follow me on Twitter"
      href="https://www.twitter.com/PLACEHOLDER"><img alt="follow me on Twitter"
      src="images/twitter.png" border=0></a> </div>
      <div><a target="_blank" title="Android Icon" href="#"><img alt="Android Icon"
      src="images/android.png" border=0></a></div>
      <div><a target="_blank" title="Mail to" href="mailto:hwu@qcc.cuny.edu"><img
      alt="Mail to Icon" src="images/mail.png" border=0></a></div>
</div>
```

css file

```
/* -- flex media --*/
.flex-media {
  display: flex;
  flex-wrap: wrap;
  background-color: lightgray;
  height: 400px;
}
.flex-media > div {
  width: 30px;
  height: 30px;
  margin: 10px;
  text-align: center;
  font-size: 30px;
}
```

SOCIAL MEDIA

Social Media Icons Using flex-wrap