Section 4
Layout in CSS, Navigation Tabs, and Social Medial Icons

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.

Apple iPhone XS
Display: 2.6 inches × 5.8 inches
Display resolution: 2436 ×1125 pixels

Samsung Galaxy S8
Display: 2.8 inches × 5.8 inches
Display resolution: 2960 x 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

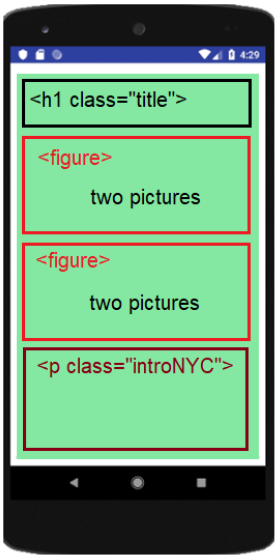
- Pages expand to fill the entire app window so there are no spaces around the page on a large screen.
- If the user has a small window, the page can contract to fit it without the user having to scroll to the side.
- The design is tolerant of users setting font sizes larger than the designer intended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

Activity) Prepare a cordova project folder (chapter 1). Open a text editor and save a HTML file as *index*.html in the www folder within the cordova project folder.

Now, create a layout using `<div>, <section>,` or `<main>` element, and two `<figure>` elements to display four images, two images per row, and description of a city using a <p> element. The layout organization should look as the following:



To create the layout, first we create a container to hold the elements of our layout. We can create this container using `<section>` or `<main>` element and assign a class name as `container`

```
<main class="container">

</main>
```

Once the container is set, we are going to use a `<h1>` element for the title, with a class name as "title", two <figure> elements to store two images in one figure element, and a <p> element to display text information. The HTML structure should look as:

```
<main class="container">
  <h1 class="title">New York City</h1>
  <figure>

  </figure>
  <figure>

  </figure>
  <p class="intro">> </p>
</main>
```

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 rel="stylesheet" href="index.css" type="text/css">
  <title>Layout by Prof. Wu</title>
 </head>
 <body>
  <main class="container">
       <h1 class="title">New York City</h1>
       <figure>

       </figure>
       <figure>

       </figure>
       <p class="intro"> </p>
  </main>
 </body>
</html>
```

Now in the CSS file, we can set the **box-sizing** for all elements by using the asterisk symbol as the following

```
*{box-sizing: border-box;}
```

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

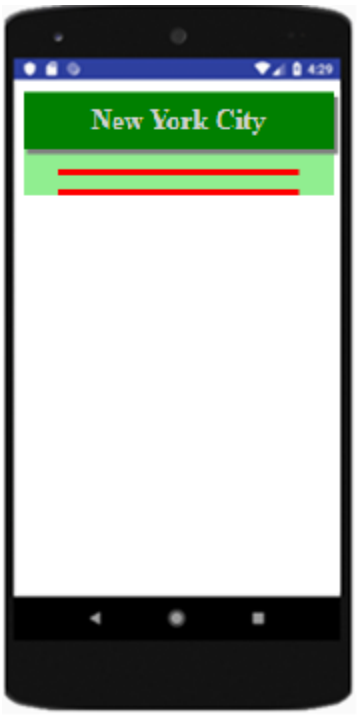
```
.title{
background-color: green;
text-align: center; color:lightgray;
padding: 12px;
box-shadow: 5px 5px 2px gray;
}
```

The title should look as the following

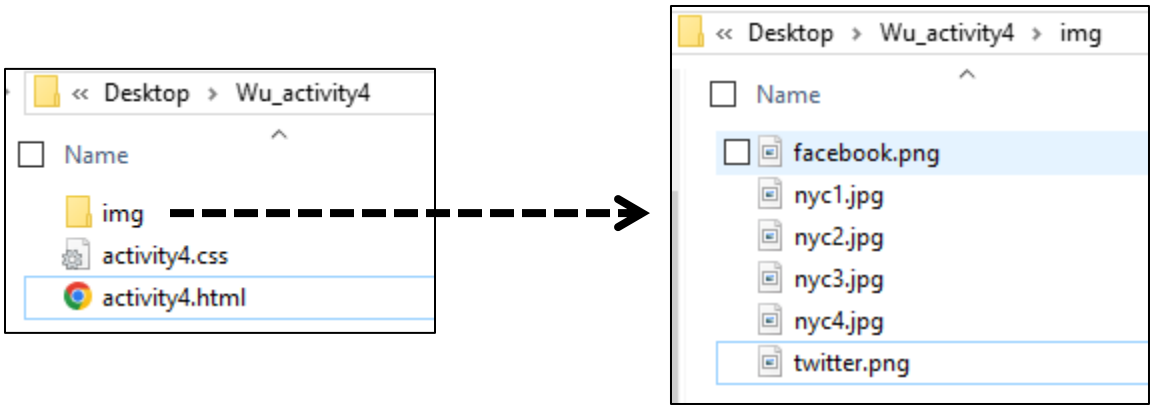
Once the title is set, we can add background-color to the main **`container`** and border the **`<figure>`** elements. We can add a border to the **`<figure>`** elements to use it as reference. This border is removed after the layout is set.

```
.container{ background-color: lightgreen; }
figure{border: solid red;}
```

The layout should look as:



Once the **<figure>** elements are set, we can insert the images. We can download four images and save them in the **img** folder in our activity folder:



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

```
<img src="img/nyc1.jpg" class="img1" /></div>
```

As shown in the code above, we also added a class name as **img1** in order to apply CSS to it. After it, we can insert two images within the first **<figure>** element and then apply CSS to each image.

```
<figure>
  <img src="img/nyc1.jpg" class="img1">
  <img src="img/nyc2.jpg" class="img2">
</figure>
```

Now in the CSS file, for the images to sit within the **<figure>** element, we can set the **width** to around 50% to **img1** and **img2** , or adjust the width until both images sit inline within the **<figure>** element:

```
.img1{width: 45%;}

.img2{width: 50%;}
```

The app view should look as the following:

Next, we can insert image 3 and 4 within the second **<figure>** element

```
<figure>
  <img src="img/nyc3.jpg" class="img3">
  <img src="img/nyc4.jpg" class="img4">
</figure>
```

In the CSS file, we can set the **width** for **img3** to 50% and for **img4** to 45%. Also, for the first image, **img1**, and the third image, **img3**, we can set **margin-right: 3%** so the images will not be very close to each other:

```
.img1{width: 45%; margin-right: 3%;}
.img2{width: 50%;}
.img3{width: 50%; margin-right: 3%;}
.img4{width: 45%;}
```

Once we finish the images set, we can remove the border from **`<figure>`** elements. The App view should look as the following:

CSS styling images

Now that we have the four images set, we can different styling to the images. The basic CSS properties that we can use for images are **border-radius, border,** and **box-shadow.**

To create rounded and circled images, we can use the `border-radius` property in the CSS file. To create rounded edges images, we can add pixels, ems, or percentage. The higher the value for

border-radius, the rounded would be the edges. To create a circled image, we can set the border-radius to 50% but the image should be a square image, the same high and width. Otherwise, if we apply 50% of border-radius a rectangle image, the resulting image would be an oval image

Rounded Images

```
border-radius: 8px;
```

Circled Image

```
border-radius: 50%;
```

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

Border image

```
border: 1px solid green;
border-radius: 6px;
padding: 10px;
```

Shadow image

```
box-shadow: 0 0 5px 10px lightblue;
```

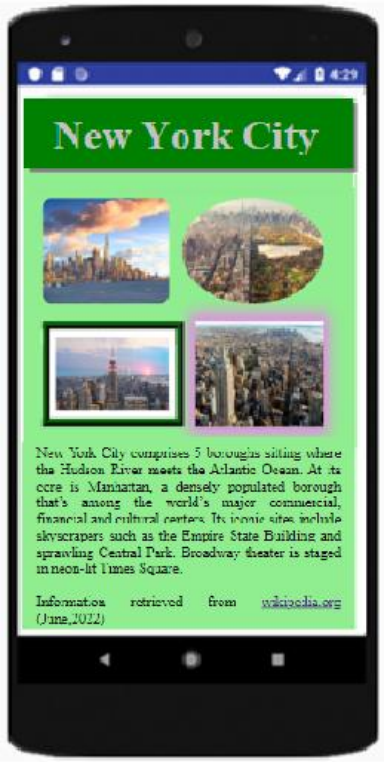
`0 0` means to not set only distance to the botton and right

`5px` means to set the blurriness of the box shadow

`10px` means to set the distance of the shadow around the box

`lightblue` is the color of the shadow of the box

Activity) Using the previous code, add the css code to round the corners of the first image, set the second image to look as a cicle or oval, add border and padding to the third image so it will look as it is framed, and add box shadow to all the corners to the fourth image. After the images, we can add a paragraph with a brief description of New York City. The complete app iew should look as:



The complete HTML code should look 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 rel="stylesheet" href="css/index.css" type="text/css">
      <title>Layout and Nav Tabs by Prof. Wu</title>
  </head>
  <body>
   <main class="container">
    <h1 class="title">New York City</h1>
    <figure>
     <img src="img/nyc1.jpg" class="img1">
     <img src="img/nyc2.jpg" class="img2">
    </figure>
    <figure>
     <img src="img/nyc3.jpg" class="img3">
     <img src="img/nyc4.jpg" class="img4">
    </figure>
    <p class="introNYC">New York City comprises 5 boroughs sitting where the Hudson
    River meets the Atlantic Ocean. At its core is Manhattan, a densely populated
    borough that's among the world's major commercial, financial and cultural
    centers. <br><hr><i>- Information retrieved from <a href="https://www.wikipedia.org/"
    target="_blank">Wikipedia.org</a></i>
    </p>
   <main>
  </body>
</html>
```

The complete CSS file code is as the following:

css file

```
*{box-sizing: border-box;}
.container{
   background-color: lightgreen;}
.title{
   text-align: center;
   background-color: green;
   padding: 12px;
   box-shadow: 5px 5px 2px gray;}
.img1{
   width: 45%;
   margin-right: 3%;
   border-radius: 10px; }
.img2{
   width: 50%;
   border-radius: 50%;}
.img3{
   width: 50%;
   margin-right: 3%;
   border: ridge 8px darkgreen;
   padding: 8px; }
.img4{
   width: 45%;
   height:100%;
   box-shadow: 0px 0px 15px 7px violet}
.introNYC{
  padding: 20px;
  text-align: justify;
  font-size: 0.8em;}
```

Navigation Tabs

A navigation tabs needs standard HTML as a base, which can be built using the unordered list `<ul>` and `<li>` elements or `<nav>` element. We can create vertical navigation tabs using `<ul>` and `<li>` and `<nav>` to create horizontal navigation tabs.

Vertical Navigation Bars

To create a basic vertical navigation tabs, in HTML file we can use the unordered list **`<ul>`** and list each hyperlink using **`<a>`** tag as the following:

```
<ul class="nav_vertical">
  <li><a href="https://www.qcc.cuny.edu/" target="_blank">QCC website</a></li>
  <li><a href="https://www1.nyc.gov/" target="_blank">NYC website</a></li>
</ul>
```

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

```
.nav_vertical{
  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.

```
.nav_vertical li a {
  display: block;
  padding: 10px 20px;
  color: lightgray;
  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:

```
.nav_vertical li a:hover {
  background-color: lightgray;
  color: purple;
}
```

When we run the app view, the navigation should look as:

Horizontal Navigation Bars

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

In the HTML file, we can start creating the horizontal navigation tabs using the **<nav>** element:

```
<nav class="nav_horizontal">
  <a href="#home">Home</a>
  <a href="#nyc">New York City</a>
</nav>
```

Once we have the HTML code set, in the CSS file, we can set the **`<nav>`** element with background-color and width:

```
.nav_horizontal{
  background-color: darkgreen;
  width: 100%;
}
```

After it, we can add CSS attributes to an **`a`** element with left border, height, width, font color, center the text, and add the padding:

```
.nav_horizontal a{
  color: lightyellow;
  padding: 10px;
  text-decoration: none;
  display:inline-block;
  width: 40%;
  text-align: center;
}
```

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

```
.nav_horizontal a:hover{
  background-color: lightyellow;
  color: darkgreen;
}
```

When you run the app view, the horizontal navigation tabs should look as:

Fixed navigation tabs

Fixed navigation tabs happens when the navigation tab 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: fixed;` to the element that contains the hyperlinks. For example, if we use the horizontal tabs HTML and CSS code from the previous topic, we just need to add to positions to the container that has the navigation tabs, which is **`nav_horizontal`**. The code will look as the following:

```
.nav_horizontal{
  background-color: darkgreen;
  height: auto;
  position: fixed;
  top: 0;
}
```

`position:fixed;` and **`top: 0;`** will make the element to have a position fixed to the top of the app. If we want the navigation to stick to the bottom of the app view, instead of using **`top: 0`** we can set it to **`bottom: 0;`**

Smooth scrolling effect

Smooth scrolling effect is a web app behavior that smoothly goes to the section of the same web page when is clicked. To do so, we can add `scroll-behavior: smooth` to the **`<html>`** element to enable smooth scrolling for the whole page.

```
html {
  scroll-behavior: smooth;
}
```

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 icons in a mobile app, we can use the <section> or <nav> elements as a container for all social media icons.

Example) create a social media section with two different social media:

Social Media

HTML

```
<section class="socialMedia">
  <a href="https://twitter.com/PLACEHOLDER"><img src="img/twitter.png"></a>
  <a href="https://www.facebook.com/ PLACEHOLDER"><img src="img/facebook.png"></a>
</section>
```

css file

```
.socialMedia{
  margin-bottom: 10em;
  background-color: lightgray;
  padding: 2em;
}
```

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.