Responsive Webpages

Making a webpage responsive to the screen size is an important features in web technology. Nowadays, web pages open on devices with different screen sizes, for example a **desktop computer** screen has a width of 1200px, a Samsung Galaxy Tab 10.1 tablet has the width of 900px, and the width of a Samsung Galaxy J7 smartphone is 720px . Therefore, it is important that when you design a webpage, the elements on the webpage can adjust to the screen of the device. You can visit: http://screensiz.es/ to view more devices width.

For app view development, it is an important to add the following line:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

The **`viewport`** is the user's visible area of a web page. It varies with the device, and will be smaller on a mobile phone than on a computer screen. **`viewport`** element gives the browser instructions on how to control the page's dimensions and scaling.

The **`width=device-width`** part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The **`initial-scale=1.0`** part sets the initial zoom level when the page is first loaded by the browser.

Without the viewport meta tag

With the viewport meta tag

Flexbox container

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning. Flexbox is a one-dimension layout method for layout g out items in rows or columns. It allows us to distribute space dynamically across elements of an unknown size, hence the term *flex*.

Example) Create a flex container for three square division

HTML

```
<section class="flex_container">
  <div class="sqr"></div>
  <div class="sqr"></div>
  <div class="sqr"></div>
</section>
```

CSS

```
.flex_container{
   border: solid gray;
   width: 80%;
   margin: 10%;
}
.sqr{width: 100px; height: 100px;}
.sqr:nth-child(1){background-color: magenta;}
.sqr:nth-child(2){background-color: olive;}
.sqr:nth-child(3){background-color: orange;}
```

without applying a *display: flex* to the flex container, the output will look as:

Now, if we add the *display:flex;* property to the flex container, the three division will wrap, from left to right, around the flex container

```
.flex_container{
   border: solid gray;
   width: 80%;
   margin: 10%;
   display: flex;
}
```

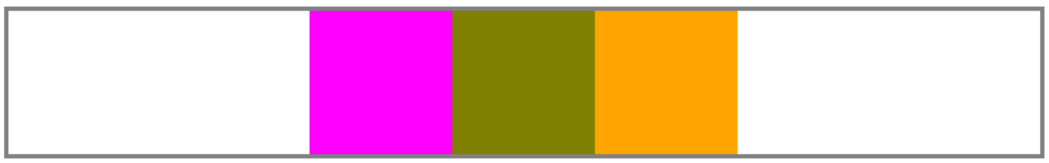
We can also change the direction from right to left by adding the property *flex-direction: row-reverse;*

```
.flex_container{
   border: solid gray;
   width: 80%;
   margin: 10%;
   display: flex;
   flex-direction: row-reverse;
}
```

justify-content property

The CSS justify-content property defines how the browser distributes space between and around content items along the main-axis of a flex container, and the inline axis of a grid container. **justify-content** uses different values such as: **start, center, space-between, space-around,** and **space-evenly**

```
.flex_container{
   border: solid gray;
   width: 80%;
   margin: 10%;
   display: flex;
   flex-direction: row;
   justify-content: center;
}
```



```
.flex_container{
   border: solid gray;
   width: 80%;
   margin: 10%;
   display: flex;
   flex-direction: row;
   justify-content: space-between;
}
```

```
.flex_container{
   border: solid gray;
   width: 80%;
   margin: 10%;
   display: flex;
   flex-direction: row;
   justify-content: space-around;
}
```

```
.flex_container{
   border: solid gray;
   width: 80%;
   margin: 10%;
   display: flex;
   flex-direction: row;
   justify-content: space-evenly;
}
```

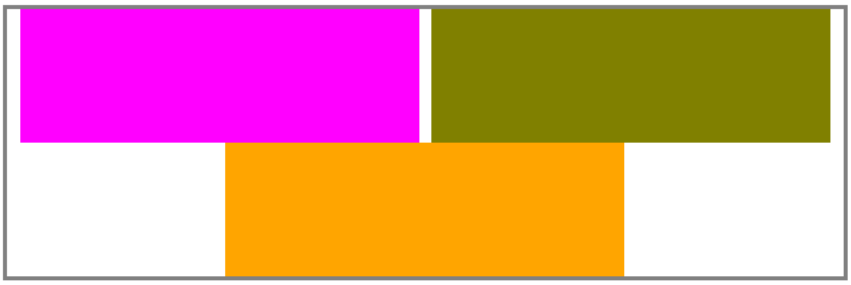
Flex-wrap property

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

Example) from the previous example, let change the **`width`** of the divisions to **`500px`**.

For this case, if we change the browser window to a smaller view, all three divisions it will squeeze to fit the width of the *flex_container*

If we do not want the divisions to squeeze, but instead, we want to keep the division's width and have the divisions to wrap around the *flex_container*, then we can add the **flex-wrap:wrap;** property to the *flex_container*



Example) from the previous example, if we set the **height** of the *flex_container* to **200px** and the **flex-direction** to **column**,

```
.flex_container{
      border: solid gray;
      width: 80%;
      margin: 10%;
      height: 200px;
      display: flex;
      flex-direction: column;
      justify-content: space-evenly;
    }
```

all three division at the container will squeeze to fit the height of the container. The output will look as:

Now, if we want to keep the height of each divisions, we can use **`flex-wrap:wrap`** to allow the flexible divisions to wrap within the *`flex_container`*

```
.flex_container{
  border: solid gray;
  width: 80%;
  margin: 10%;
  height: 200px;
  display: flex;
  flex-direction: column;
  justify-content: space-evenly;
  flex-wrap: wrap;
}
```

Align-item property

Align-items property sets the align-self value on all direct children as a group. In flexbox, it controls the alignment of items inside the flex container on the cross axis.

Example) using the three divisions, set the divisions to align to the end of the flex container.

Without using the *`align-item`* property,

```
.flex_container{
  border: solid gray;
  width: 80%;
  height: 200px;
  margin: 10%;
  display: flex;
  flex-direction: row;
  justify-content: center;
}
```

By adding the ***align-items:flex-end;*** to *flex_container*

```
.flex_container{
  border: solid gray;
  width: 80%;
  height: 200px;
  margin: 10%;
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items:flex-end;
}
```

We can also set the divisions to the center of the cross axis of the flex container by using **align-items:center;**

```
.flex_container{
  border: solid gray;
  width: 80%;
  height: 200px;
  margin: 10%;
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items:center;
}
```

align-content property

align-content property sets the distribution of space between and around content items along a flexbox's cross-axis or a grid's block axis. The `align-content` property modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines. It sets the distribution of space between and around content items along a flexbox's cross-axis or a grid's block axis.

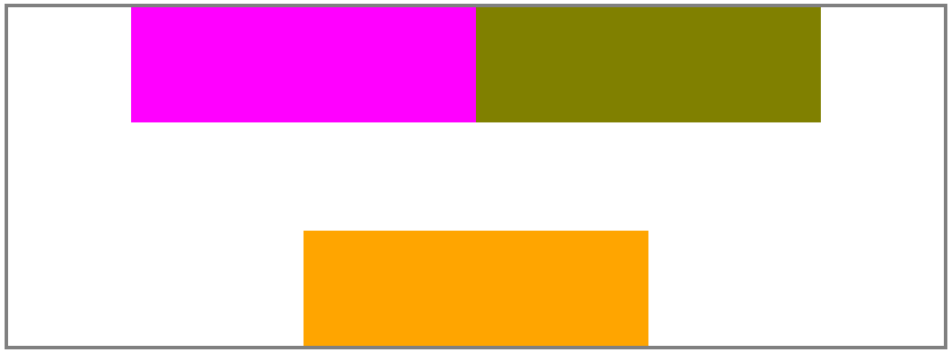
Note: There must be multiple lines of items for this property to have any effect!

Example) use the previous example and change the **width** of the division to **300px**

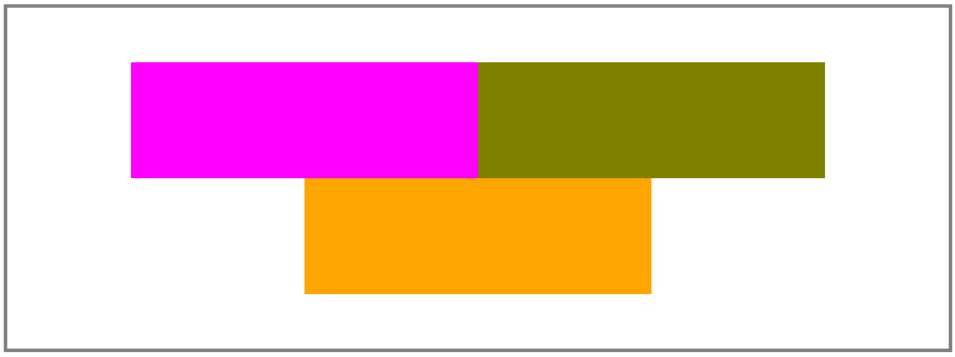
```
.flex_container{
      border: solid gray;
      width: 80%;
      height: 300px;
      margin: 0 auto;
      display: flex;
      flex-direction: row;
      justify-content: center;
      align-items:flex-end;
      flex-wrap: wrap;
    }

.sqr{width: 300px; height: 100px;}
```

Now, by adding the property ***align-content: space-between;***



Now, if we change ***align-content: center;***

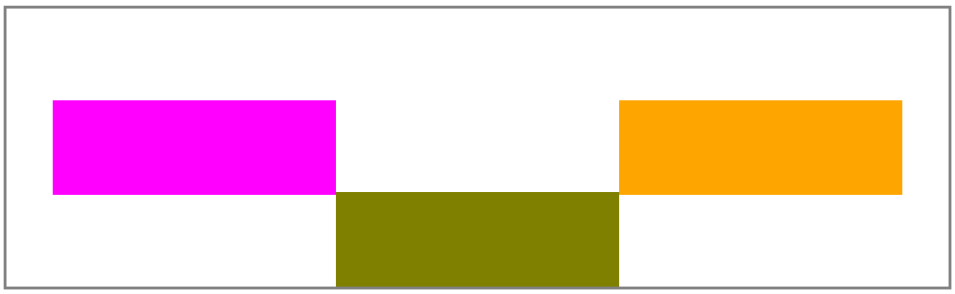


align-self property

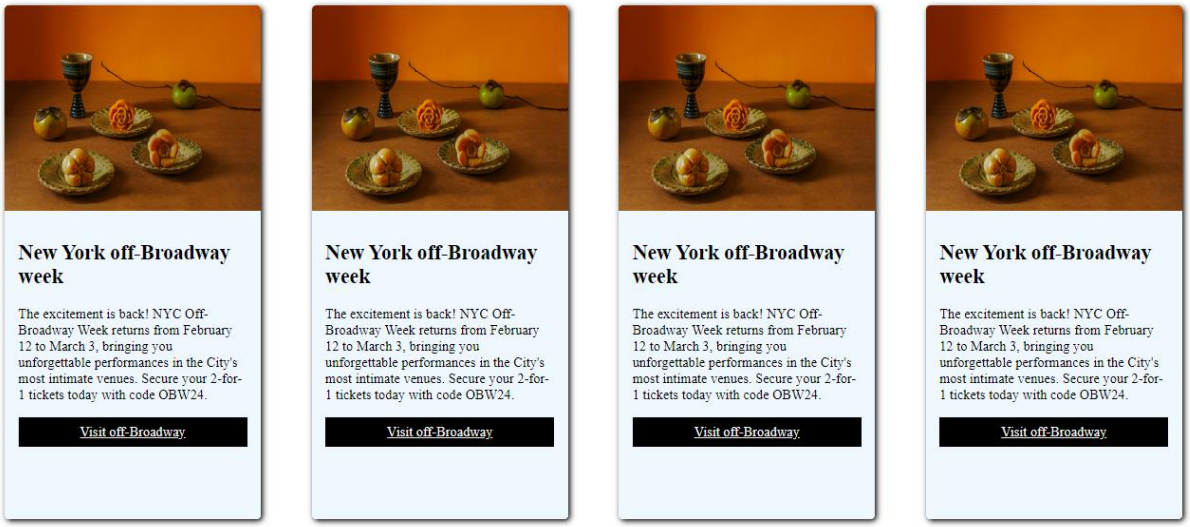
The *align-self* CSS property overrides a grid or flex item's align-items value. In Grid, it aligns the item inside the grid area. In Flexbox, it aligns the item on the cross axis. The *align-self* property specifies the alignment for the selected item inside the flexible container.

Example) using the previous example, align the second division to the bottom of the flex container

```
      .flex_container{
        border: solid gray;
        width: 80%;
        height: 300px;
        margin: 0 auto;
        display: flex;
        flex-direction: row;
        justify-content: center;
        align-items:center;
        flex-wrap: wrap;
      }
.sqr:nth-child(2){background-color: olive; align-self: flex-end;}
```



Example) create four cards webpage and apply responsive properties to the cards



HTML file

```
<main class="maincontainer">
    <!-- card 1 -->
    <section class="card">
        <img src="images/fruits.jpg" alt="" class="imgcard">
        <div class="cardinfo">
            <h2>New York off-Broadway week</h2>
             <p>The excitement is back! NYC Off-Broadway Week returns from February
             12 to March 3, bringing you unforgettable performances in the City's
             most intimate venues. Secure your 2-for-1 tickets today with code
             OBW24.</p>
            <a href="https://ww.nyctourism.com/off-broadway-week/" target="_lank"
            rel="noopener noreferrer" class="cardlink">Visit off-Broadway</a>
        </div>
     </section>

     <!-- card 2 -->
     <section class="card">
        <img src="images/fruits.jpg" alt="" class="imgcard">
        <div class="cardinfo">
            <h2>New York off-Broadway week</h2>
             <p>The excitement is back! NYC Off-Broadway Week returns from February
             12 to March 3, bringing you unforgettable performances in the City's
             most intimate venues. Secure your 2-for-1 tickets today with code
             OBW24.</p>
            <a href="https://ww.nyctourism.com/off-broadway-week/" target="_lank"
            rel="noopener noreferrer" class="cardlink">Visit off-Broadway</a>
        </div>
     </section>

    <!-- card 3 -->
    <section class="card">
        <img src="images/fruits.jpg" alt="" class="imgcard">
        <div class="cardinfo">
```

```
            <h2>New York off-Broadway week</h2>
             <p>The excitement is back! NYC Off-Broadway Week returns from February
             12 to March 3, bringing you unforgettable performances in the City's
             most intimate venues. Secure your 2-for-1 tickets today with code
             OBW24.</p>
            <a href="https://ww.nyctourism.com/off-broadway-week/" target="_lank"
            rel="noopener noreferrer" class="cardlink">Visit off-Broadway</a>
        </div>
     </section>

</main>
```

CSS code

```
main.maincontainer{
    width: 80%;
    margin: 10%;
    display: flex;
    justify-content: space-evenly;
    flex-wrap: wrap;
}
section.card{
    width: 300px;
    height: 600px;
    box-shadow: 2px 2px 6px black;
    border-radius: 5px;
    overflow: hidden;
    margin: 1em;
}
img.imgcard{
    width: 100%;
    height: 40%;
    display: block;
}
div.cardinfo{
    background-color: aliceblue;
    padding: 1em;
    height: 60%;
}
a.cardlink{
    display: inline-block;
    background-color: black;
    color: whitesmoke;
    padding: 0.5em;
    width: 100%;
    text-align: center;
    transition: 300ms;
}
a.cardlink:hover{
    background-color: crimson;
    box-shadow: 0px 0px 6px 2px black;
    transform: scale(1.01);
    font-size: 1.05em;
    text-decoration: none;
}
```

@media query

Media query is a CSS technique introduced in CSS3 and it is used to make responsive pages.

It uses the `@media` rule to include a block of CSS properties only if a certain condition is true.

When using @media, instead of changing styles when the width gets *smaller* than 800px, we should change the design when the width gets *larger* than 800px. This will make our design Mobile First. The syntax code will look as:

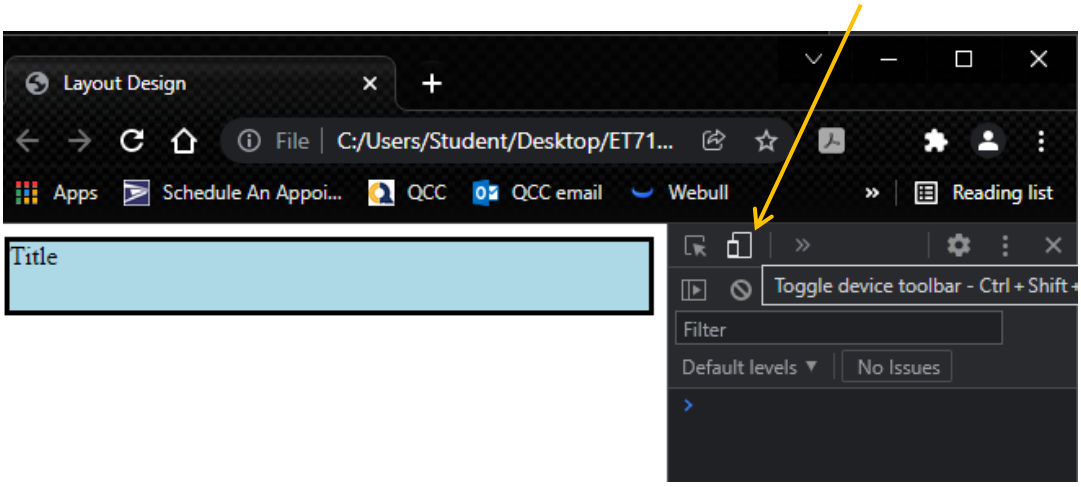
```
@media only screen and (min-width: 800px){

}
```

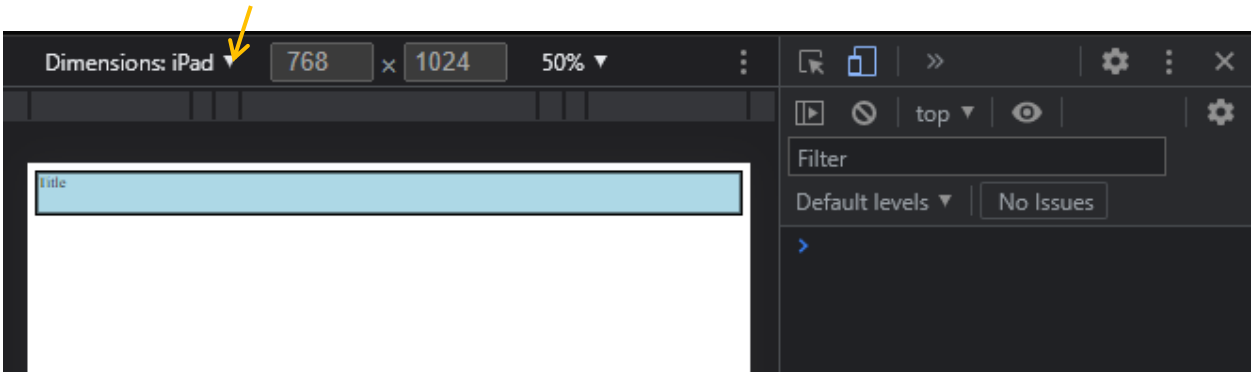
Between the curly brackets should go the CSS attributes of the elements that will changed when the screen has the width of 800px or greater.

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices). Some web developer prepares to design a mobile view first as it moves toward the tablet's, laptop's, and desktop's screen size. Therefore, when we apply @media query, the screen size has property **`min-width: 800px`**. On the other hand, since the material in this lab manual was designed from a desktop computer screen view, then we can design from the desktop computer screen toward the tablet's and smartphone's screen size. For this, instead of using **`min-width:800px`** we use **`max-width: 800px`**

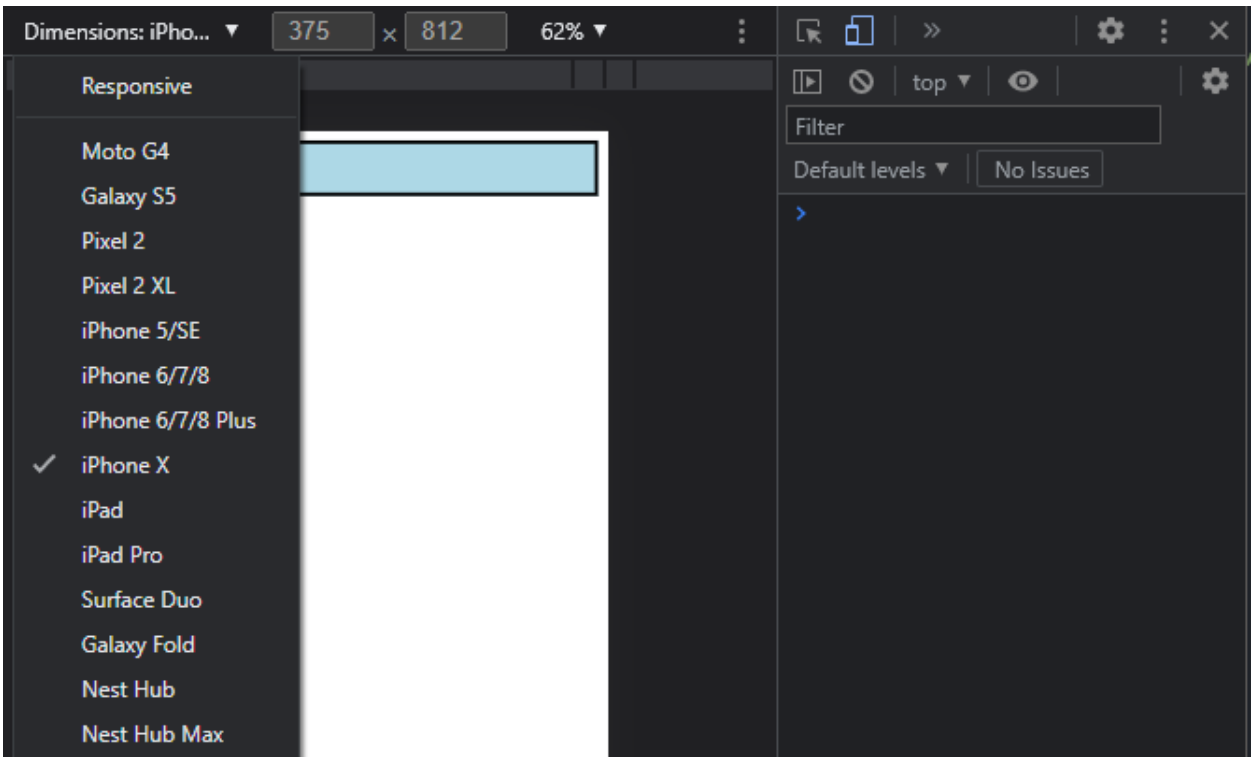
Also, if we are designing from mobile view first, we can change the browser view to mobile view by using the function key **F12** and then click on the **Toggle device toolbar**



Once clicked on the Toggle device toolbar, we can select the size of the mobile view:



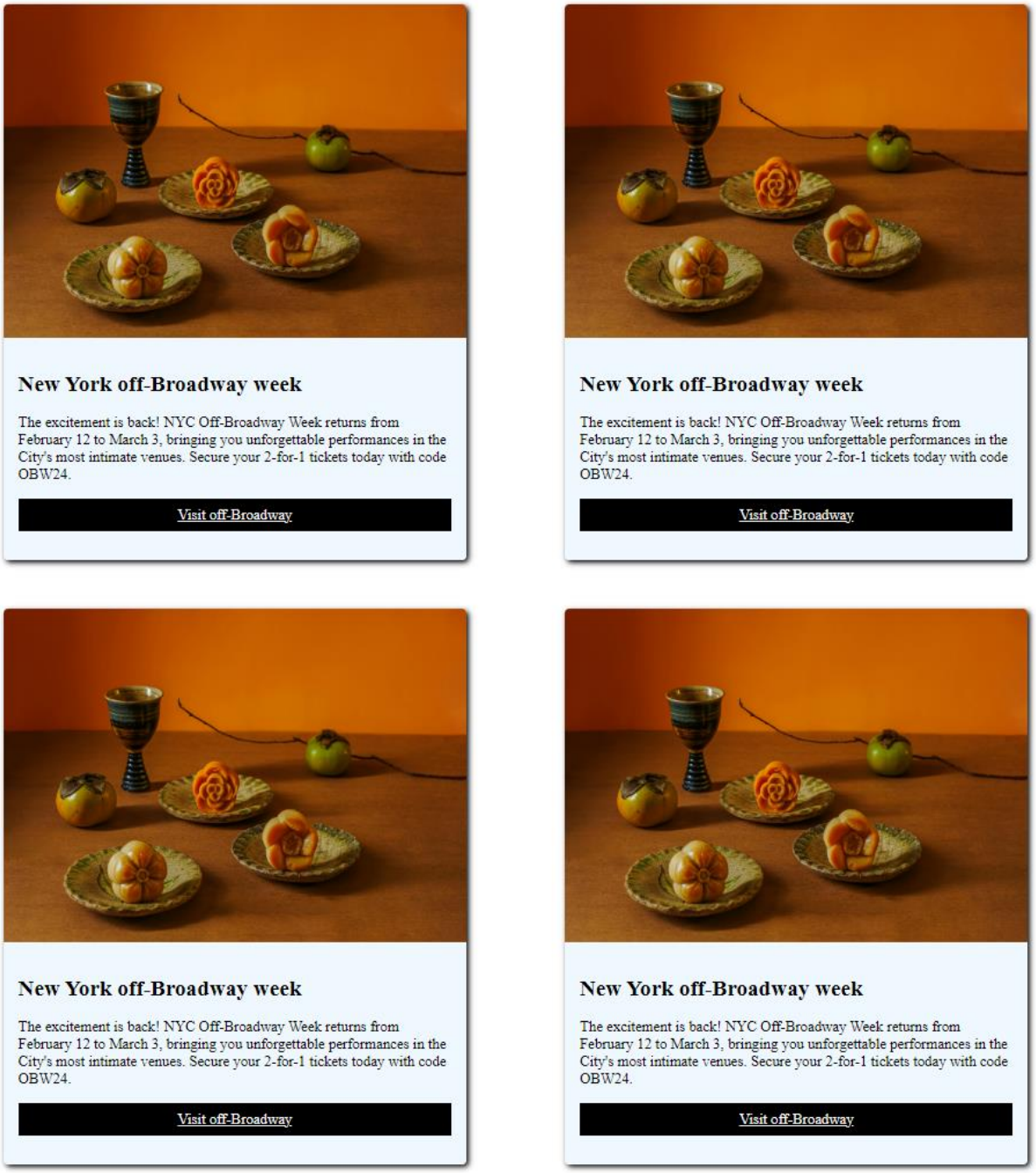
Let us to pick the iPhone X screen size. In this case, since iPhone X has a width of 375px, then we can design a view up to 375px or 450px so the design can be used to other smartphone screen.



Example 2) using the four cards design from example 1, use media query to adjust the CSS styling as:

- At 1700px, make the browser view to change two cards. To do so, adjust the width of the cards.
- At 1450px, make the browser to view one card.

The browser view smaller than 1700px would look as:



The browser view smaller than 1400px would look as:

New York off-Broadway week

The excitement is back! NYC Off-Broadway Week returns from February 12 to March 3, bringing you unforgettable performances in the City's most intimate venues. Secure your 2-for-1 tickets today with code OBW24.

Visit off-Broadway

New York off-Broadway week

The excitement is back! NYC Off-Broadway Week returns from February 12 to March 3, bringing you unforgettable performances in the City's most intimate venues. Secure your 2-for-1 tickets today with code OBW24.

Visit off-Broadway

New York off-Broadway week

The excitement is back! NYC Off-Broadway Week returns from February 12 to March 3, bringing you unforgettable performances in the City's most intimate venues. Secure your 2-for-1 tickets today with code OBW24.

Visit off-Broadway

New York off-Broadway week

The excitement is back! NYC Off-Broadway Week returns from February 12 to March 3, bringing you

CSS code

```
/*------ BREAKPOINT = 1700PX ---------*/
@media only screen and (max-width:1700px){
    section.card{width: 500px ; margin: 2em;}
    img.imgcard{height: 60%;}
}
/*------ BREAKPOINT = 1450PX ---------*/
@media only screen and (max-width: 1450px){
    section.card{width:100%; height: auto;}
    img.imgcard{height: auto;}
    img.cardinfo{height:auto;}
}
```