Class 5 - Flexbox and Media queries

Agenda:

1. overflow
2. Flexbox
3. media queries

CSS Overflow

When the content goes out of the container is known as Overflow.

Let's take an example where we create a div element and add content of 100 words to it. We create a container and give it a static width and height.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>overflow</title>
 <style>
   .container{
     height:100px;
     width:300px;
     border:2px solid tomato;
   }
 </style>
</head>
<body>
 <div class="container">
   Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequuntur rem doloribus
perspiciatis, accusamus sit voluptatum voluptas vel, mollitia odio, doloremque optio
```

```
asperiores praesentium corporis molestias natus officia numquam fugit. Quibusdam,
ipsam. Cumque, tenetur. Vel, nulla sed delectus, error accusamus ut quibusdam
obcaecati quos voluptatum rerum laudantium, veniam quam iste possimus numquam facere
ducimus? Culpa, nostrum doloribus. Laudantium voluptatem nulla molestias velit a
praesentium, fuga nobis dolorem aliquid placeat ipsa cupiditate quaerat dolorum vel
vero ex libero rerum, deleniti similique! Tempore ea tenetur amet cumque recusandae
aliquid magnam dolores! Dignissimos, sequi. Odio assumenda quam, harum culpa mollitia
obcaecati facilis consectetur eveniet?
 </div>
 </body>
</html>
```

The result will be that the box will not contain the whole content and it will create an overflow condition. Therefore, we will use overflow to overcome it. We can use auto, hidden, scroll and visible overflow properties.

Example

```
<style>
   .container{
     height:100px;
     width:300px;
     border:2px solid tomato;
     overflow: hidden  ;
   }
 </style>
```

Properties of Overflow

Hidden: The content that remains outside the container will be hidden or not visible. It does not occupy space where the content is hidden.

Scroll: A scroll bar is added inside the container so that you can go through the whole content inside the container.

Auto: It works similarly to scroll the only difference is that scroll will always show scroll inside the container while auto will only show when the content is overflowing.

Visible: It will make the content visible. There will be no change.

The container takes only the space designated to it. Therefore, the overflow content will overlap with the other content added after the container.

YOu can also choose to add scroll in any one particular direction like x-axis or y-axis

```
.horizontal-container{
      font-size: 30px;
      width: 50px;
      height: 30px;

      border: 2px solid black;

      overflow-x: auto;
  }
```

```
<div class = "horizontal-container">
  Scaler
</div>
```

If you want to hide the vertical or horizontal scroll bar you can use the following syntax: overflow-x: hidden; or overflow-y: hidden;.

Flexbox

1. Flexbox is a CSS layout model optimized for building complex web layouts.
2. It makes it easier to design flexible responsive layout structures without using complicated css
3. Starter code

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Flexbox</title>
   <style>
       .container {
           background-color: tomato;
           border: 2px solid black;
           height: 500px;
           width: 500px;
       }

       .box {
           height: 100px;
           width: 100px;
```

```
            background-color: dodgerblue;
            margin: 10px;
        }

    </style>
</head>

<body>
    <div class="container">
        <div class="box">1</div>
        <div class="box">2</div>
        <div class="box">3</div>
        <div class="box">4</div>
    </div>
</body>

</html>
```

4. Lets say we want to bring them in the same line
5. Now we apply flexbox on the parent container

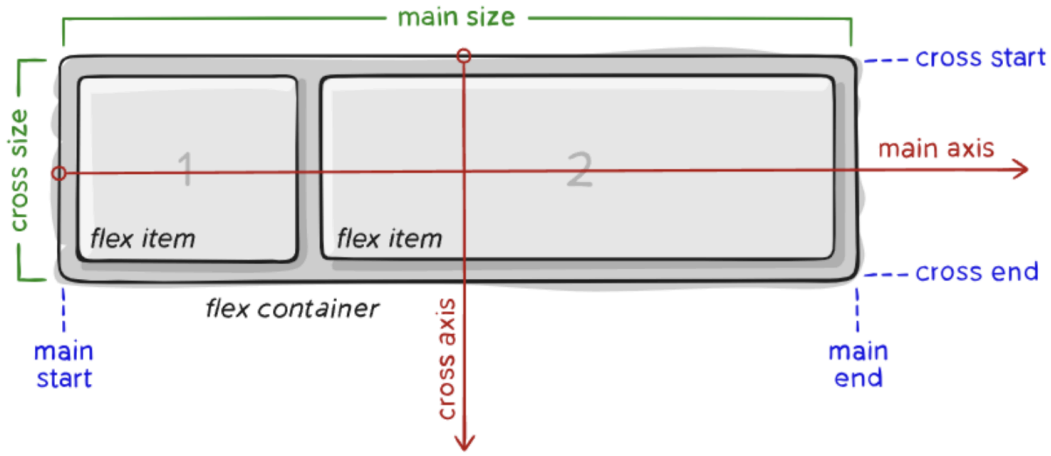
```
.container {
            background-color: tomato;
            display: flex;
            border: 2px solid black;
            height: 500px;
            width: auto;
        }
```

6. When you apply the CSS property display: flex; to an element, you're essentially creating a flex container. This container

establishes a flex formatting context for its child elements, or "flex items." This means that the child elements within this container will follow the rules and behaviors defined by the flexbox layout model.

7. Default property of flex box is to arrange all the elements in the row as you can also see here.
8. You can also use many flex-direction property to arrange the elements in different ways.
9. Use different-different values with flex-direction property like row, column, row-reverse, column-reverse etc, to explain it better.

Main axis and cross axis



1. Main Axis: This is the primary direction in which the flex items are laid out. Think of it as the main line that the items follow. If you choose flex-direction: row;, the main axis runs horizontally (left to right). If you choose flex-direction: column;, it runs vertically (top to bottom).

2. Cross Axis: This is perpendicular to the main axis. It's the direction across the main line of items. So, if your main axis is horizontal (left to right), the cross axis runs vertically (top to bottom), and vice versa.
3. Justify content
    a. justify-content: Defines how flex items are distributed along the main axis (horizontal for row layout, vertical for column layout).
    b. So if we want to center all the row items, we can use justify-content: center

```
.container {
    background-color: tomato;
    display: flex;
    border: 2px solid black;
    height: 500px;
    width: auto;
    justify-content: center;
}
```

4. To align items in the cross axis, we can use align items property

```
.container {
    background-color: tomato;
    display: flex;
    border: 2px solid black;
    height: 500px;
    width: auto;
    justify-content: center;
    align-items: center;
}
```

5. If I say, I want my flex direction is row, what will be my main axis?
    a. X axis
6. If I say, I want my flex direction is column, what will be my main axis?
    a. Y axis
7. Now lets take more examples of justify-content property.
    a. justify-content: flex-start;
        i. Flex items are aligned at the beginning of the container (left for a row layout, top for a column layout).
    b. justify-content: flex-end;
        i. Flex items are aligned at the end of the container (right for a row layout, bottom for a column layout).
    c. justify-content: center;
        i. Flex items are centered along the container's main axis.
        ii. Equal space is added before the first item and after the last item, creating a balanced appearance.
    d. justify-content: space-between;
        i. Flex items are evenly spaced along the main axis.
        ii. The first item aligns with the container's start, the last item aligns with the container's end, and equal space is added between the items.
    e. justify-content: space-around;

    i. Flex items are evenly spaced along the main axis, with space distributed around them.
    ii. Space is added before the first item, after the last item, and between each pair of items.
  f. justify-content: space-evenly;
    i. Flex items are evenly spaced along the main axis, with equal space added between them.
    ii. Equal space is added before the first item, between all items, and after the last item.
    iii. There are many options for this property, you can choose to play around with all of them.
8. Now lets understand align-items property.
  a. The align-items property lets you control how flex items are positioned on the cross axis within the container.
  b. align-items: flex-start;
    i. Flex items align at the start of the cross axis (top for row layout, left for column layout).
    ii. No additional space is added between items and the container's cross axis edge.
  c. align-items: flex-end;
    i. Flex items align at the end of the cross axis (bottom for row layout, right for column layout).
    ii. No additional space is added between items and the container's cross axis edge.
  d. align-items: center;
    i. Flex items are vertically centered along the cross axis.

ii. Equal space is added above and below the items, creating a balanced appearance.

e. align-items: baseline;

i. Flex items are aligned along their text baselines.

ii. This value can be useful when items have varying font sizes or text content.

iii. To see, comment the background color of the box class and add align -items:baseline

```
.box {
            height: 100px;
            width: 100px;
            /* background-color: dodgerblue; */
            margin: 10px;
           text-decoration:underline
        }
```

```
.container {
            background-color: tomato;
            border: 2px solid black;
            height: 500px;
            /* width: auto; */

            display: flex;
            justify-content: center;
            /* align-items: baseline; */
        }
<div class="container">
    <div class="box box1" style="font-size: 16px;">1</div>
    <div class="box box2"  style="font-size: 32px;">2</div>
    <div class="box box3"  style="font-size: 48px;">3</div>
    <div class="box box4"  style="font-size: 22px;">4</div>
```

```
</div>
```

iv. Now we see that the all the text are on the same base line (same bottom base line)

f. align-items: stretch;

i. Flex items are stretched to fill the container's cross axis.

ii. If no height is explicitly set on the items, they will take up the full height of the container.

1. Use align-items:stretch and comment the height of the boxes

```
.box {
            /* height: 100px; */
            width: 100px;
            background-color: dodgerblue;
            margin: 10px;
        }
```

Responsive website

1. A responsive website is a website that can work of different screen size and can adapt to those different screen sizes. It responds and adjusts its layout, images, and content to fit the screen it's being viewed on, whether that's a desktop computer, tablet, or smartphone.
2.

3. In inspect tab of our HTML website, we can check how our website will look in different dimensions and screen sizes.
4. flex-wrap
    a. flex-wrap: The flex-wrap property in flexbox controls whether flex items should wrap onto multiple lines within the flex container when there's not enough space to fit them all on a single line.
    b. So if we use the above code with flex-wrap property, we can see that the boxes are now responsive and not shrinking.

Order in flexbox

1. here is 1 property called order that is used for ordering the items.
2.
3. The order property in flexbox allows you to control the visual order in which flex items appear within a flex container, regardless of their order in the HTML markup. It's particularly useful for reordering items for different screen sizes or creating unique visual layouts. H
4.
5. lets give the order property to box1, and see what happens:

```
.box1{
            order:4
        }
```

6. Now you can see that the box1 is ordered at the last.
7. If we give order to other boxes

```
.box1{
            order:4
        }
        .box2{
            order:1
        }
        .box3{
            order:2
        }
        .box4{
            order:3
        }
```

Flex shrink

1. The flex-shrink property in flexbox determines how flex items shrink when the container's available space is limited. It defines the ability of an item to shrink in relation to other items in the container when the container's size is reduced.
2. let's understand this with an example:
    a. We are providing flex-shrink property as 2 to box3.
3.

```
.container {
            background-color: tomato;
            border: 2px solid black;
            height: 500px;
            /* width: auto; */
            display: flex;
            justify-content: center;
            align-items: flex-start;
            /* flex-wrap: wrap; */
```

```
            align-content: center;
        }
```

```
    .box3{
            order:2;
            flex-shrink:4;
        }
```

4. Flex-grow
    a. This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up

```
5.  .box2{
6.          order:1;
7.          flex-grow: 3;
8.      }
```

Media queries

1. Responsiveness in web design refers to the ability of a website or web application to adapt and provide an optimal user experience across various devices and screen sizes.
2. A responsive design ensures that the content and layout of a website adjust dynamically based on the device's characteristics, such as screen width, height, and orientation.

3. This approach enhances usability and accessibility, offering a seamless experience for users on desktops, laptops, tablets, and smartphones.
4. apply styles based on certain conditions, typically related to the device's features. Here's an example:
5. Add in the style tag

```
@media only screen and (max-width: 600px) {
     body {
       font-size: 14px;
       background-color: lightcoral;
     }
   }

   /* Media query for devices with a screen width between
601 and 900 pixels */
   @media only screen and (min-width: 601px) and
(max-width: 900px) {
     body {
       font-size: 18px;
       background-color: lightblue;
     }
   }

   /* Media query for devices with a screen width of 901
pixels or more */
   @media only screen and (min-width: 901px) {
     body {
       font-size: 20px;
       background-color: lightgreen;
```

```
        }

    }
```