Class 4 - CSS Display, Box Model and Positioning

Agenda:

1. CSS Box Model
2. Display properties (block, inline, inline-block)
3. Positioning (absolute, relative, fixed, sticky , static)

CSS BOX model

1. The box model in web design is a way to explain how space is managed around and within HTML elements on a webpage. It's like imagining each element as a box with several layers
2. In other words if i put it simply, everything that you write in html is inside a box
3. Code
4. Create a new index.html in class4 folder

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Box Model</title>
</head>
<body>
   <div class="box1">
       <h1>Hello </h1>
   </div>
```

```
    <div class="box2">
        <h1>Good Bye</h1>
    </div>

</body>
</html>
```

5. Run with live server and check
6. Give some style inside the head tag

```
    <style>
        .box1{
            background-color: lightblue;
        }
        .box2{
            background-color: lightgreen;
        }
    </style>
</head>
```

7. Let us give some custom height and width to our boxes

```
.box1{
            background-color: lightblue;
            height:100px;
            width:100px;
        }
        .box2{
            background-color: lightgreen;
            height:150px;
            width:200px;
        }
```

a. px" stands for "pixels." In digital graphics and web design, a pixel is the smallest unit of measurement on a digital display. It's essentially a tiny dot of color on your screen

8. Discuss border and margin

```
.box1{
    background-color: lightblue;
    height:100px;
    width:100px;
    border:1px solid black;
    margin:20px;
}
.box2{
    background-color: lightgreen;
    height:150px;
    width:200px;
    border:2px dashed blue
}
```

9. Now give some padding to the first element
10. Make their height and width the same to strike out the difference

```
.box1{
    background-color: lightblue;
    height:200px;
    width:200px;
    border:2px solid black;
    margin:20px;
    padding:100px;
}
```

```
        .box2{
            background-color: lightgreen;
            height:200px;
            width:200px;
            border:2px dashed blue
        }
```

11. See how the box1 grows in size
12. Generally this idea is not correct.
13. When we build houses, our plot area is fixed. You cannot have a big house and also have a big lawn to your liking
14. Here our plot area is determined by height and width that is given
15. So we use Border-box

Universal selectors

1. We will make use of universal selectors to apply some styles to all the elements in the html page

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

2. Change the padding to a smaller value and have margin for both the boxes

```
.box1{
```

```
        background-color: lightblue;
        height:200px;
        width:200px;
        border:2px solid black;
        margin:20px;
        padding:10px;
    }
    .box2{
        background-color: lightgreen;
        height:200px;
        width:200px;
        border:2px solid blue;
        margin:20px;
    }
```

3. See that changing the padding adjusts the height and width of the element in border box (in the dev tools)
4. **Width = border-left + padding-left + width of element + border-right + padding-right**
5. box-sizing: content-box is the default box model.
    a. When you set an element's box-sizing to content-box, the width and height you apply to that element are only applied to the content area of the box.
    b. This means that the padding and border are added to the outside of the box, making the actual size of the box larger than the width and height specified

Some more padding, margin discussions

1. Create another mbp.html
2. Create three divs with class container

```
<body>
    <div class="container">
        <h1>this is heading</h1>
        <p>Lorem, ipsum dolor sit amet consectetur
adipisicing elit. Facere pariatur ex explicabo! Ut
laboriosam beatae voluptates aliquid totam omnis esse animi
veritatis magnam, eveniet a autem officiis, in suscipit ex
doloremque dolore harum exercitationem quos unde corporis,
culpa asperiores facere doloribus. Omnis soluta quam animi
eaque fugiat tenetur rerum sed.</p>
    </div>
    <div class="container">
        <h1>this is heading</h1>
        <p>Lorem, ipsum dolor sit amet consectetur
adipisicing elit. Facere pariatur ex explicabo! Ut
laboriosam beatae voluptates aliquid totam omnis esse animi
veritatis magnam, eveniet a autem officiis, in suscipit ex
doloremque dolore harum exercitationem quos unde corporis,
culpa asperiores facere doloribus. Omnis soluta quam animi
eaque fugiat tenetur rerum sed.</p>
    </div>
    <div class="container">
        <h1>this is heading</h1>
        <p>Lorem, ipsum dolor sit amet consectetur
adipisicing elit. Facere pariatur ex explicabo! Ut
laboriosam beatae voluptates aliquid totam omnis esse animi
veritatis magnam, eveniet a autem officiis, in suscipit ex
doloremque dolore harum exercitationem quos unde corporis,
```

```
culpa asperiores facere doloribus. Omnis soluta quam animi
eaque fugiat tenetur rerum sed.</p>
   </div>

</body>
```

3. Add style tag

```
<style>
        *{
            margin:0;
            padding:0;
            box-sizing:border-box;
        }
        .container{
            height:200px;
            width:500px;
            margin:30px;
        }
```

4. Create three class to give background colors

```
.first{
            background-color: lightblue;
        }
        .second{
            background-color: lightgreen;
        }
        .third{
            background-color: lightpink;
        }
```

5. Play with margin left , top, bottom and see

```
.container{
            height:200px;
            width:500px;
            /* margin:30px; */
            margin-left:10px;
            margin-top: 30px;
        }
```

6. Givr padding left and padding top
7. Border top bottom etc can be given separately as well

CSS shorthand for margin, padding and border

1. In CSS, you can add margin and padding to these sides, but typing out each side individually can be time-consuming. This is where shorthand comes in.
2. Four Sides (Top, Right, Bottom, Left): clockwise
    a. If you want to add margin or padding to all four sides equally, use a single value: margin: 10px; or padding: 10px;. This applies 10px to top, right, bottom, and left, uniformly decorating your box.
3. Vertical & Horizontal (Top & Bottom, Right & Left):
    a. To specify the vertical (top and bottom) and horizontal (right and left) sides differently, use two values: margin: 10px 5px; or padding: 10px 5px;. Here, 10px is for top and bottom, and 5px is for right and left.

4. Three Sides:
    a. If you need to set different values for three sides, use three values: margin: 10px 5px 15px; or padding: 10px 5px 15px;. The first value is for the top, the second for right and left, and the third for the bottom.
5. Each Side Differently:
    a. To decorate each side with a unique value, use four values: margin: 10px 5px 15px 20px; or padding: 10px 5px 15px 20px;. This applies 10px to the top, 5px to the right, 15px to the bottom, and 20px to the left.

Display property in css

1. Create a new file display.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Display</title>
</head>
<body>
   <h1>Hello Everyone</h1>
   <h2>Welcome to Scaler</h2>
   <a href="https://scaler.com">scaler website</a>

</body>
```

```
</html>
```

2. What we see is that each of this in a new line
3. Add border to each and see

```
<style>
        h1{
            border:1px solid black;
        }
        h2{
            border:1px solid black;
        }
        a{
            border:1px solid black;
        }
    </style>
```

4. What if we add a span tag before anchor

```
<body>
    <h1>Hello Everyone</h1>
    <span>Welcome to Scaler</span>
    <a href="https://scaler.com">scaler website</a>
    <span>another span content</span>

</body>
```

5. Now notice that except h1, all are on same line and with the help of border, we can see that it only occupies the space that it needs

6. BLOCK ELEMENTS: Your heading, divs, paragraph etc are what are called as block elements
    a. These elements are characterized by the way they behave in the document layout
    b. **Full Width by Default**: Block-level elements typically take up the full width available in their parent container. This means they stretch out to the full width of the page if they are not inside another container, or to the full width of their parent element if they are nested within another element.
    c. **Blocking Adjacent Elements**: Block elements create a “block” in the layout. This doesn't mean they "block" the area next to them in a literal sense, but rather that they don't allow other elements to sit beside them horizontally in the normal document flow.
    d. **Starting on a New Line**: Each block-level element starts on a new line, and any content or elements after it will also start on a new line
    e. In addition to your div, p and headings (h1 to h6), lot of semantic tags are block elements like your navs, section, etc
7. Inline elements
    a. Inline Elements: Inline elements are HTML elements that do not start on a new line and only take up as much width as necessary. This is in contrast to block-level elements which occupy the full width available.

b. , <a>, , and are common inline elements. They can be used to apply styling or semantic meaning to a part of a text without disrupting the flow of the text.
c. Inline elements do not cause a line break, meaning they appear on the same line alongside other elements, as long as there is space.
d. Inline elements can be nested within block elements, but nesting block elements within inline elements can create unexpected behaviors and is generally not recommended.

8. Ljst of inline and block elements - https://www.w3schools.com/html/html_blocks.asp

Changing Display properties

1. Changing their default display behavior

```
h1{
            border:1px solid black;
            display: inline;
        }
```

2. Lets also change anchor tag's default display behavior

```
a{
            border:1px solid black;
            display: block;
        }
```

Height and Width in inline elements

1. Lets give some height and width to h1 which we have made as inline

```
h1{
            border:1px solid black;
            display: inline;
            height: 500px;
            width:500px;
        }
```

2. Observer that no effect. Give it a larger value and still no effect
3. Now give height and width to span tag which is inline by default
4. No changes
5. The concept is that When you set an element to display: inline;, it will not respect height and width properties.
6. This is a fundamental characteristic of inline elements
7. The content will still determine the size of the element, and the height and width properties won't force it to resize

1.

Inline block

1. But often you would want to give some height and width in addition to making elements as inline
2. In such cases, we have a third display property known as inline-block

Margin and Padding for inline and block elements (extra)

2. Block Elements: For block elements, all margins (top, bottom, left, right) and padding are respected. They can influence the layout by affecting the element's total size and its spacing with other elements
3. Horizontal margins for inline elements-
    a. Horizontal margins (left and right) and padding are respected on inline elements. This means they will affect the horizontal spacing of the element.
4. Vertical margin and padding:
    a. This might change things visually but not in expected manner
5. Make anchor tag inline again and give it margin and padding

```
a{
            margin:200px;
            border:1px solid black;
            /* display: block; */
        }
```

6. Comment the inline block of h1
7. Change html to have a div below anchor tag

```
<h1>Hello Everyone</h1>
   <span>Welcome to Scaler</span>
   <a href="https://scaler.com">scaler website</a>
   <div>another div content</div>
```

8. **Now with margin , we observe that horizontal margins are respected but not vertical margins**
9. Now try giving some padding to a tag

```
a{
            margin:200px;
            padding:20px;
            border:1px solid black;
            /* display: block; */
        }
```

10. Change the html below to have some more text

```
<div>lorem100</div>
```

11. What we observe is that although padding area was created but there is an overlap with the content below
12. Crux of all this discussion is that these vertical margins and paddings **do not change the line height of the line the element** is in for inline elements. This means they don't push away adjacent lines of text in the same way as they would in a block element

Postitions

1. Let us understand this with examples
2. Create a file position.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Position</title>
</head>
<body>
   <img src="./poster1.jpeg">
   <img src="./poster2.jpeg">
   <img src="./poster3.jpeg">
   <img src="./poster4.jpeg">

</body>
</html>
```

We will see that we can give the height and width of these images so it behaves as inline-block

3. Give all images height and width some value

```
<style>
       img{
           width:250px;
           height: 250px;
       }
   </style>
```

4. Add some margin to the img tag and a border on the container class

```
<style>
        .container{
            border: 2px solid black;
        }
        img{
            width:250px;
            height: 250px;
            margin:10px;
        }
    </style>
```

5. Default position:
    a. here we see that Elements are positioned according to the normal flow of the document.this position property is called as static.
    b. You can think of it as the natural flow of elements on a page, just like books arranged on a shelf in the order they're added.
    c. Use Case: It's used when you want elements to follow the normal page flow. No need to specify this unless you're overriding another positioning type.

6. Relative positioning
    a. Relative positioning moves an element relative to its normal position. The space it originally took up remains reserved.
    b. Let us add a relative position property to one of the img
    c. Give different values for top
    d. It moves relative to its original position

```
.image2{
            position:relative;
            top:250px;
        }
```

    e. Add one more class on image3 and move them left and top

```
 .image3{
            position:relative;
            left:50px;
            bottom: 40px;
        }
```

7. Absolute position
    a. It's positioned relative to its closest positioned ancestor (other than static).
    b. Use Case: Great for creating overlays, modals, or when you need an element to stay in a specific place within a relative container.
    c. Out of document flow

```
.image3{
            position:absolute
```

```
        }
```

i. Observe the ui and notice that the fourth image is not visible
ii. What happens here is when we make an element as absolute, it is removed from the document flow like it does not exist and the neighbouring element if any swoops in and takes its position
iii. So in our case, image 4, swoops in and takes the position that image3 was occupying
iv. It behaves like a spirit floating above , detached

d. Positioning wrt nearest positioned ancestor
    i. The use case for this property lies in the fact that you can position an element wrt nearest positioned ancestor (ancestor has position relative or absolute or other property except static)
    ii. If there are no ancestor positions, it is placed wrt viewport
e. Absolute positioning is often used for elements that need to be placed very precisely within a container, like a modal, a custom tooltip, or a dropdown menu
f. Give left and buttom values to image3

```
.image3{
        position:absolute;
        left:50px;
        bottom: 40px;
    }
```

g. Notice that currently it is places relative to the viewport
h. If we give position as relative to our container, then observe the difference

8. Position: fixed
    a. Lets make height of our container to be 1000px so that we have some scroll

```
.container{
    border: 2px solid black;
    margin-top: 100px;
    /* position: relative; */
    height: 1000px;
}
```

    b. Observe that when we scroll images move out of the view eventually
    c. Now let us make image4 as fixed
    d. It stays fixed to its position
    e. We can give top, left, right , bottom to specify its position
    f. use case for this
        i. Perfect for navigation bars or buttons that need to stay visible, regardless of scrolling.

9. Position:sticky
    a. Let make image 3 as sticky

```
.image3{
        position:sticky;
        top:0;
}
```

b. Starts Normal: An element with position: sticky; starts out behaving just like any regular element on the page. It stays in the spot where you put it in your HTML code.
c. Sticks When Needed: As you scroll down the page, the sticky element stays where it is until the top of the element reaches a certain distance from the top of the screen — this distance is what you set using the top property.
d. Glue Point: Once the top of the sticky element is the same distance from the top of the screen as you specified with the top property it's like the element becomes glued to that spot. Even if you keep scrolling down the page, the element won't move any further up; it will stay stuck at that exact position from the top of the screen.
e. Unsticks When Scrolled Past: If you keep scrolling down and reach the end of the container that the sticky element is in, the element will stop being sticky and will scroll off the screen like a normal element again.
f. In summary, a sticky element toggles between being normal and being fixed based on your scroll position. It "sticks" to a certain point when scrolled into position and then "unsticks" when you scroll past its container.

10. Lets see one more example to make it clear
    a. In the second container, we will create a sticky header

```
<div class="container2">
    <h2 class="sticky-header">I'm a Sticky Header!</h2>
    <p>Scroll down to see the header stick to the top.</p>
    <div class="filler-content">Filler content</div>
  </div>
```

b. styles

```
.container2 {
      background-color: lightgoldenrodyellow;
      height: 1500px;
    }
    .sticky-header {
      background-color: #333;
      color: white;
      padding: 10px;
      font-size: 20px;
      position: sticky;
      top: 10px;
    }
    .filler-content {
      height: 1800px;
      background-color: #f1f1f1;
      padding-top: 20px;
    }
```

c. Now the header sticks when the offset of 10px is reached from the top