The topics we are going to cover in today's session are:

- Box Model
- Display Property in CSS
- Overflow in CSS
- CSS Units
- Anything that we create in HTML and CSS takes the form of a box.
- You can check this by inspecting.
- Go to any website that we have created till now, select an html element paragraph and right click, then click on **inspect**. Here you see the boxing of your content in the right panel.

Always remember to add height and width to your box otherwise, it will not be visible. Make the box as shown below.

> **Image Not Showing**
> Possible Reasons
> - The image was uploaded to a note which you don't have access to
> - The note which the image was originally uploaded to has been deleted
>
> Learn More →

You can customize the size of the box by changing the height and width values.

> Take a real-life example of creating a township including steps like making land cut, house placing, etc. to discuss the box concept. And introduce `padding`.

Padding

- It is the empty area inside the container.
- You can apply the padding outside your content but it should be inside your container box.
- Syntax to add padding:

```
h1{
    padding : 20px;
}
```

- **Increasing the padding** area **decreases the container** space. Because padding is only inside that specific container, it can not go outside of the box.

Margin

- The CSS margin properties are used to create space around two different elements, outside of any defined borders.
- It prevents your elements from getting overlapped.
- Syntax:
- Create another box as in the previous example then,

```
.box_2{
    width: 100px;
    height: 120px;
    background-color: blue;
}
```

- There are already some margins by CCS default. But you can customize it as :

```
.box_2{
    width: 100px;
    height: 120px;
    background-color: blue;
    margin : 0px;
}
```

> You can use the universal selector (*) to apply the desired CSS property to all the elements.

Syntax:

```
*{
   CSS_Property
}
```

Applying Margins, Padding and Border

- First, let us create three containers:

Image Not Showing

Possible Reasons

- The image was uploaded to a note which you don't have access to
- The note which the image was originally uploaded to has been deleted

Learn More →

- The CSS that we apply in the containers will be applied to all the containers because they all have the same name.
- When you apply margin 20px, it gets reflected in the element from all sides.
- If you want to apply margin from a specific side say top, then :

```
.container{
    background-color : lightgreen;
    martin-top : 20px;
}
```

- Likewise, you can apply margin for **bottom**, **left** and **right** side too.

Now, let us apply padding.

- If you apply padding to the container, it gets reflected in the **div tag**. You can check by applying padding to the container.
- So, it should be avoided. Rather it is best practice to **apply padding to the elements** by targeting the element separately in the style tag as:

```
h1{
    padding : 20px;
}
```

Coming to margins,

- If can apply four values to the margin property as `margin : 10px 20px 30px 40px;` it will be reflected as `margin : top right bottom left`
- You can try using different margin values to see the different results.
- If you pass three values to the margin property and forget the last value, as `margin: 10px 20px 30px;` , then it will be reflected as "10 px for top and 20 px as left and right both and 40px for bottom".
- If you pass two values to the margin property as `margin : 10px 20px;` , then it will be reflected as "10px for top and bottom, and 20px as left and right.

> All these concepts are similar for **padding**. You just need to use padding in place of margins. And the rest of the rules will be the same.

Border

- Border is defined as that extra space that you can create around your HTML elements.
- There are many types of borders such as **solid**, **dashed**, **dotted**, etc. that you need to define in the syntax to see it.
- Syntax:

```
.container{
    background-color : lightgreen;
    martin-top : 20px;
    border : 2px solid red;
}
```

Border Alignment Properties

- border-top : 5px solid blue ;
- border-left: 2px solid red;
- border-right: 6px dashed green;
- border bottom: 5px solid red;

Border Radius Properties

- It is used to apply curved edges to the borders
- Syntax:

```
border-radius : 20px;
```

You can also apply this radius to a specific side of the border for example : `border-top-right_radius: 20px;`

> These are the all properties and concepts (Margins, Padding, and Borders) that together are termed as "**Box Model** in CSS".

> Summarizing

- Everything that we create in an HTML is taken as a Box known as **Box Model**.
- It will have two dimensions that are Height and Width.
- And these boxes have three properties that are **Margin**. **Padding**, and **Border**.

First, let us create a file as shown below to understand the working of the Display property:

> **Image Not Showing**
> Possible Reasons
> - The image was uploaded to a note which you don't have access to
> - The note which the image was originally uploaded to has been deleted

Learn More →

> Note: Rem is also a unit to define pixels. It converts the pixel into 10 times. For example **5rem** is **50px**.

When you see the output you can see that the border is taking the whole area of paragraphs. But it should be on the paragraph **p1** only.

Now, let us add a span tag to the file as shown below:

Image Not Showing

Possible Reasons

- The image was uploaded to a note which you don't have access to
- The note which the image was originally uploaded to has been deleted

Learn More →

When you run this file, You can see that these **span tags** are not acting as a paragraph. This means that there are no borders applied to it.

Now, give this span tag border, height, and width property as :

```
span{
    border: 2px solid blue;
    height: 50px;
    width: 50px;
}
```

Now give the p tag height and width as :

```
p{
    border: 2px solid red;
    height: 50px;
    width: 50px;
}
```

You can see in the output that there are no changes seen in the span tag.

> Note: Block elements can have customizable height and width but you can not provide height and width to the inline elements (**span tag** in this case).

Here comes the need for **Display property**

- Add a random image to the file using **img src tag**.
- Also add an anchor tag entitled "click me".

- You can see that you can provide customizable height and width to the image. Here, it is working as both inline and block elements.
- CSS allows the user to add all of these tags as a Display property.
- Syntax:

```
p{
    display : inline;
}
```

This makes your element an **inline-block**. That's the work of the Display property in CSS. After that, you can provide height and width to the inline elements too.

CSS Units in Detail

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Units Examples</title>
  <style>
    /* Pixels (px) */
    .px-example {
      width: 200px;
      height: 150px;
      font-size: 16px;
      background-color: lightblue;
    }

    /* Percentage (%) */
    .percentage-example {
      width: 50%;
      padding: 5%;
      font-size: 120%;
      background-color: lightgreen;
    }

    /* Viewport Width (vw) and Viewport Height (vh) */
    .viewport-example {
      width: 50vw;
      height: 30vh;
      background-color: lightcoral;
    }

    /* EM */
    body {
      font-size: 16px;
    }

    .em-example {
      margin: 1em; /* Equivalent to 16px in this case */
      background-color: lightsalmon;
    }
```

```
    /* REM */
    html {
      font-size: 16px;
    }

    .rem-example {
      margin: 1rem; /* Always equivalent to 16px */
      background-color: lightgoldenrodyellow;
    }

    /* Ch */
    .ch-example {
      width: 20ch; /* Approximately 20 times the width of '0' character */
      background-color: lightblue;
    }

    /* Ex */
    .ex-example {
      line-height: 2ex; /* Twice the height of 'x' character */
      background-color: lightpink;
    }

    /* Viewport Percentage Width/Height (vmin, vmax) */
    .vmin-vmax-example {
      width: 50vmin;
      height: 70vmax;
      background-color: lightseagreen;
    }
  </style>
</head>
<body>

  <div class="px-example">Pixel Example</div>

  <div class="percentage-example">Percentage Example</div>

  <div class="viewport-example">Viewport Example</div>

  <div class="em-example">EM Example</div>

  <div class="rem-example">REM Example</div>

  <div class="ch-example">Ch Example</div>

  <div class="ex-example">Ex Example</div>

  <div class="vmin-vmax-example">Vmin/Vmax Example</div>

</body>
</html>
```

Open this HTML file in a web browser to see the different examples of CSS units in action. Adjust the values and observe how the layout changes based on the specified units.

With each unit Example you use these Definitions-

1. **Pixels (px):**
   - The `px` unit is fixed and does not change with the size of the viewport or user settings.
   - Useful for creating fixed layouts.
   - Example:

```
div {
  width: 200px;
  height: 150px;
  font-size: 16px;
}
```

2. **Percentage (%):**
   - The `%` unit is relative to the size of the containing element.
   - Useful for creating fluid layouts that adapt to different screen sizes.
   - Example:

```
div {
  width: 50%;
  padding: 5%;
  font-size: 120%;
}
```

3. **Viewport Width (vw) and Viewport Height (vh):**
   - `vw` is a percentage of the viewport width, and `vh` is a percentage of the viewport height.
   - Useful for creating responsive designs based on the viewport size.
   - Example:

```
div {
  width: 50vw;
  height: 30vh;
}
```

4. **EM:**
   - The `em` unit is relative to the font-size of the closest parent or the element itself.
   - Useful for creating scalable and accessible layouts.
   - Example:

```
body {
  font-size: 16px;
}

div {
  margin: 1em; /* This is equivalent to 16px in this case */
}
```

5. **REM:**
   - Similar to `em` but relative to the font-size of the root element (usually the `<html>` tag).
   - Useful for maintaining consistent spacing throughout the document.
   - Example:

```
html {
  font-size: 16px;
}

div {
  margin: 1rem; /* This is always equivalent to 16px */
}
```

6. **Ch:**
   - Represents the width of the "0" (zero) character in the font used.
   - Useful for creating layouts based on character width.
   - Example:

```
div {
  width: 20ch; /* Approximately 20 times the width of the '0' character */
}
```

7. **Ex:**
   - Represents the height of the "x" character in the font used.
   - Useful for setting sizes based on the height of characters.
   - Example:

```
div {
  line-height: 2ex; /* This is equivalent to twice the height of the 'x' chara
}
```

8. **Viewport Percentage Width/Height (vmin, vmax):**

- `vmin` is the smaller of `vw` and `vh`, while `vmax` is the larger of the two.
- Useful for responsive designs based on the smaller or larger viewport dimension.
- Example:

```
div {
  width: 50vmin;
  height: 70vmax;
}
```

Remember to choose units based on your specific design requirements and the behavior you want for your web page. Combining different units wisely can lead to more flexible and responsive layouts.

- **Resources:**
- MDN Web Docs - Units and Values
- CSS Units: A Free Quick Reference by Example

Recommended Resources:

1. MDN Web Docs: Box Model