CSS PART-2

INTRODUCTION TO WEB DEVELOPMENT

R.A: **Assad Ullah Khan**

assadullahkhan

AssadUllahKhan

DISPLAY PROPERTY

In web development, the **display** property is used to control how an element is rendered and displayed on a webpage. It determines the type of layout the element will have and how it interacts with other elements in the document flow.

- ✓ **Block**
- ✓ **Inline**
- ✓ **Inline-block**
- ✓ **None**
- ✓ **Flex**
- ✓ **Grid**

DISPLAY BLOCK

- The **block** value for the **display** property is a commonly used value in web development. When an element's **display** property is set to **block**, it generates a block-level box.

    1) **Block-level** elements start on a new line and occupy the **full width**.
    2) They typically **stack vertically**, one after another.
    3) **Block-level** elements ignore the **height** and **width** properties unless specified.
    4) They can have **margins**, **padding**, and **borders** applied to them.
    5) By default, **block-level** elements expand to fill the available **width** of their **parent** container, pushing other elements **horizontally**.

- By using the **display: block** property, you can control the layout and positioning of elements in a block-like manner, allowing you to create structured and organized web page layouts.

DISPLAY INLINE

- The **inline** value for the **display** property is another commonly used value in web development. When an element's **display** property is set to **inline**, it generates an inline-level box.

1. **Inline-level** elements do not start on a new line. They flow along with the surrounding content **horizontally**.
2. They only occupy the space necessary for their content, so they won't create line breaks or disrupt the **layout** flow.
3. **Inline-level** elements cannot have a **width** and **height** set explicitly. They expand or shrink based on the content within them.
4. **Margins** and **padding** on **inline-level** elements only affect the left and right sides, not the top and bottom.

DISPLAY INLINE-BLOCK

- The **inline-block** value for the **display** property is a combination of both **inline** and **block** behaviors. When an element's **display** property is set to **inline-block**, it generates an inline-level box that behaves like a block-level element.

    1. Inline-block elements flow along with the surrounding content horizontally, just like inline elements.
    2. They can have a width, height, margins, padding, and borders set explicitly, similar to block-level elements.
    3. Inline-block elements respect top and bottom margins, allowing for vertical spacing between elements.

Example

Block Example

```
.header{
  border: 1px solid blue;
  display: block;
}
```

```
div{
  border: 2px solid blue;
  padding: 10px;
  margin-top: 5px;
  display: inline-block;
  width: 47%;
}
```

Inline Example

```
.header{
  border: 1px solid blue;
  display: inline;
}
```

- You can convert **block-level** elements into **inline-level** element and **inline-level** element to **block-level** element. Hide element using **none** property.
- You can set the layout using display inline-block level element.

DISPLAY NONE

- The **none** value for the **display** property is used to hide an element completely from the web page. When an element's **display** property is set to **none**, it removes the element from the document flow, making it effectively invisible.

    1. Elements with **display: none** do not take up space in the layout. They are not rendered and do not affect the positioning or layout of other elements.

    2. Any child elements within an element with **display: none** will also be hidden.

    3. Applying **display: none** to an element is commonly used for conditional hiding or showing of elements based on user interactions or other events.

MAX-WIDTH & MIN-WIDHT

- **MAX-WITDH:** This property specifies the maximum width that an element can have. It restricts the element from expanding beyond the specified value. If the content within the element requires less space, the element will shrink accordingly. However, if the content exceeds the specified maximum width, the element will maintain the maximum width and may overflow or wrap the content.
- **MIN-WIDTH:** This property specifies the minimum width that an element must have. It ensures that the element does not become narrower than the specified value. If the content within the element requires more space, the element will expand to accommodate it.
- By using **max-width** and **min-width** in combination, you can create responsive designs that adapt to different screen sizes. For instance, you can set a maximum width to prevent an element from becoming too wide on larger screens and set a minimum width to ensure it remains readable on smaller screens.

CSS BACKGROUNDS

- SS backgrounds allow you to apply styles and visuals to the background of an element on a webpage. The **background** property is used to control various aspects of the background, and there are multiple properties available to customize its appearance.

“background-color” : Sets the background color of an element. You can specify colors using color **names**, **hexadecimal** values, **RGB** values, or **HSL** values.

```
.container{
    background-color: aqua;
}
```

BACKGROUND IMAGE

- **Background-image:** Sets an image as the background of an element. You can use an image URL or specify gradients and other special effects.

Syntax:

```
.element{
    background-image: url('path/to/image.jpg');
}
```

Example:

```
.container{
    background-image: url('images/banner/header.jpg');
}
```

BACKGROUND REPEAT

- **Background-repeat:** Determines how a background image repeats or doesn't repeat in the background.

Syntax:

```
.element{
    background-repeat: repeat-x;
    background-repeat: repeat-y;
    background-repeat: no-repeat;
}
```

Example:

```
.container {
    background-image: url('images/sub1.jpg');
    height: 400px;
    background-repeat: repeat-x;
}
```

BACKGROUND POSITION

- **Background-position:** Sets the position of a background image within the element's background area.

Syntax:

```
.element{
    background-position: top center;
    background-position: top left;
    background-position: top right;
}
```

Example:

```
.container {
    background-image: url('images/sub2.jpg');
    height: 400px;
    background-position: top center;
}
```

BACKGROUND SIZE

- **background-size**: Specifies the size of a background image. It can be set to values like **cover**, **contain**, or specific dimensions.

Syntax:

```
.element{
    background-size: cover;
}
```

Example:

```
.container {
    background-image: url('images/sub2.jpg');
    height: 1200px;
    background-size: cover;
}
```

End

I hope this slide will help you to clear your idea