**Assignment 2 – PPT Web Dev**

Q.1 What’s Box Model in CSS ?

The Box Model is a fundamental concept in CSS (Cascading Style Sheets) that describes how elements are rendered and how their dimensions and spacing are calculated.

In the Box Model, every element in a web page is considered as a rectangular box. This box consists of four main components:

1. Content

2. Padding

3. Border

4. Margin

The dimensions of an element, including its width and height, are calculated based on the combination of its content, padding, border, and margin.

Total width = content width + left padding + right padding + left border + right border + left margin + right margin

= 200px + 20px + 20px + 2px + 2px + 10px + 10px

= 264px

The same concept applies to the height calculation.

Q.2 What are the Different Types of Selectors in CSS & what are the advantages of them?

CSS (Cascading Style Sheets) provides various types of selectors that allow you to target and style specific elements or groups of elements on a web page. Here are some common types of selectors:

* Element Selectors
* Class Selectors
* ID Selectors
* Attribute Selectors
* Pseudo-Selectors

The advantages of using different types of selectors in CSS include:

- Specificity: Different selectors have different levels of specificity, allowing you to target elements with varying degrees of precision.

- Reusability: Class selectors allow you to apply styles to multiple elements with the same class, promoting code reusability.

- Modularity: Class and ID selectors enable you to create modular styles and easily update or modify specific elements without affecting others.

- State-based styling: Pseudo-selectors allow you to style elements based on user interactions or specific states, enhancing user experience.

- Targeted styling: Attribute selectors let you target elements based on specific attribute values, allowing for more precise styling.

Q.3 What is VW/VH ?

VW and VH are units of measurement in CSS that are used to define lengths relative to the viewport, which is the visible area of a web page.

Here's what VW and VH stand for:

- VW: Viewport Width

- VH: Viewport Height

The VW and VH units are useful when you want to create responsive designs that adapt to different screen sizes. By using VW and VH units, you can ensure that elements on the page scale proportionally based on the dimensions of the viewport, providing a consistent experience across devices.

Q.4 Whats difference between Inline, Inline Block and block ?

The main differences between inline, inline-block, and block elements lie in their rendering behavior, positioning, and how they interact with surrounding elements. Inline elements flow within the text content, inline-block elements are a hybrid of inline and block behavior, and block elements start on a new line and occupy the full width available.

Q.5 How is Border-box different from Content Box?

Border-box includes padding and border within the specified width and height, while content-box calculates width and height based on the content area only.

Q.6 What’s z-index and How does it Function ?

z-index is a CSS property that controls the stacking order of elements in the z-axis, which determines their visibility and overlapping behavior on the webpage.

* z-index specifies the stacking order of positioned elements (those with a position property other than static).
* Elements with a higher z-index value are positioned in front of elements with lower values.
* The z-index property accepts integer values, where higher values indicate elements appear closer to the viewer.
* By default, elements have a z-index value of auto, which means they are stacked based on their order in the HTML document.
* Elements with a negative z-index value are positioned behind elements with positive values or default stacking order.

Q.7 What’s Grid & Flex and difference between them?

Grid: Grid is a two-dimensional layout system in CSS that allows for the creation of complex and structured layouts by dividing the webpage into rows and columns. It provides precise control over the placement, alignment, and sizing of elements in both horizontal and vertical directions.

Flex: Flex, short for Flexbox, is a one-dimensional layout system in CSS that enables the creation of flexible and responsive layouts. It operates along a single axis, either horizontally or vertically, allowing elements to dynamically adjust their size and spacing to fill available space. Flexbox provides powerful alignment and distribution capabilities, making it ideal for creating dynamic and responsive layouts of individual components.

The main difference between them is that a Grid is a two-dimensional layout system for complex layouts with precise control over rows, columns, and alignment whereas a Flex is one-dimensional layout system for simpler layouts with flexible and dynamic component positioning along a single axis.

Q.8 Difference between absolute and relative and sticky and fixed position explain with example.

1. Absolute Positioning:

- With absolute positioning, an element is positioned relative to its closest positioned ancestor or to the initial containing block if there is no positioned ancestor.

- The element is taken out of the normal document flow, and its position is set using the `top`, `bottom`, `left`, and `right` properties.

- Other elements in the document flow may overlap or occupy the space originally occupied by the absolutely positioned element.

- Example:

<div style="position: relative;">

<div style="position: absolute; top: 50px; left: 50px;">Absolute Position</div>

</div>

In this example, the inner `<div>` is positioned absolutely 50 pixels from the top and 50 pixels from the left of its closest positioned ancestor, which is the outer `<div>` with relative positioning.

2. Relative Positioning:

- With relative positioning, an element is positioned relative to its normal position in the document flow.

- The element still takes up space in the document flow, and its position is adjusted using the `top`, `bottom`, `left`, and `right` properties.

- Other elements are not affected by the relative positioning of this element.

- Example:

<div style="position: relative;">

<div style="position: relative; top: 20px; left: 20px;">Relative Position</div>

</div>

In this example, the inner `<div>` is positioned relatively 20 pixels from the top and 20 pixels from the left of its normal position in the document flow.

3. Sticky Positioning:

- With sticky positioning, an element is positioned based on the user's scroll position.

- Initially, the element behaves like a relatively positioned element until it reaches a specific threshold (defined using `top`, `bottom`, `left`, or `right` values).

- Once the threshold is reached during scrolling, the element becomes "sticky" and remains fixed at that position until it reaches another boundary.

- Example:

<div style="position: sticky; top: 50px;">Sticky Position</div>

In this example, the `<div>` is initially positioned according to the document flow. When the user scrolls and the `<div>` reaches 50 pixels from the top of its containing element, it becomes sticky and remains fixed at that position.

4. Fixed Positioning:

- With fixed positioning, an element is positioned relative to the browser window or viewport.

- The element is removed from the normal document flow and its position is set using `top`, `bottom`, `left`, and `right` properties.

- The element remains fixed at its specified position, even when the user scrolls the page.

- Example:

<div style="position: fixed; top: 0; right: 0;">Fixed Position</div>

In this example, the `<div>` is positioned fixed at the top right corner of the viewport. It will remain in that position regardless of scrolling.