Module - 3 - MERN-Stack - CSS and CSS3

Q - 1 : What is a CSS selector? Provide examples of element, class, and ID selectors.

A - 1 : CSS selectors are used to "find" (or select) the HTML elements you want to style.

Example of Element Selector :	Example of Class Selector :	Example of ID Selector :

p {
  text-align: center;
  color: red;
}
	.center {
	  text-align: center;
	  color: red;
	}
		#para1 {
		  text-align: center;
		  color: red;
		}

Q - 2 : Explain the concept of CSS specificity. How do conflicts between multiple styles get resolved?

A - 2 : Specificity is a calculated weight or rank assigned to different CSS selectors. The selector with the highest weight always "wins" the conflict.

- Each selector is assigned a **specificity value**, usually expressed as four parts:
- inline styles, IDs, classes/attributes/pseudo-classes, elements/pseudo-elements.

Specificity Levels (from lowest to highest):

Selector type	Examples	Specificity
Element / pseudo-element	Div , p , :: before	0 , 0 , 0 , 1
Class / attribute / pseudo-class	.box , [type=”text”] , :hover	0 , 0 , 1 , 0

ID	#header	0 , 1 , 0 , 0
Inline Style	style=”color:red”	1 , 0 , 0 , 0

- When multiple rules apply to the same element and property, the browser resolves conflicts in this order:
    - !important (**Overrides almost everything**)
    - Specificity (The selector with the **highest Specificity** wins.)
    - Source Order (If specificity is equal, **the last rule defined wins.**)
    - Inheritance (Some properties (like color or font-family) are inherited from parent elements , **only if no rule directly applies.**

Q - 3 : What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

A - 3 : **Difference between internal, external, and inline CSS.**

Feature	Inline CSS	Internal CSS	External CSS
Location	Within a specific HTML tag's `style` attribute	Within the `<head>` section of an HTML document	In a separate `.css` file
Syntax	`style="property: value;"`	`<style> selector { ... } </style>`	`selector { ... }`
Scope	Only the single element it is applied to	Only the specific page it is embedded in	The entire website (multiple pages)
reusability	None	Limited (per page)	High (site-wide)
Pros	Quick fixes, useful for testing	No extra file needed	Clean HTML, efficient caching, site-wide control
Cons	Clutters HTML, difficult to manage	Clutters `<head>`, not reusable	Requires an extra file/HTTP request

		across pages	

Q - 4 : Explain the CSS box model and its components (content, padding, border,margin). How does each affect the size of an element?
A - 4 : The **CSS box model** is the layout model that describes how every HTML element is represented as a rectangular box on a webpage.

Components of the CSS Box Model :

1. Content

- The actual content of the element (text, image, video, etc.)
- Effects : Defines the inner dimensions (width/height) of the element.
- Its size is controlled by **properties** like:
    - Hight,
    - Width.

2. Padding

- Space **between the content and the border.**
- Increases the visible size of the element
- Padding is transparent.
- Effects : Adds space between the content and the border. Increases the total visible size (background color/image extends into this area).
- **Properties**:
    - padding.
    - padding-top , padding-right , etc .

3. Border

- Wraps around the padding and content
- Has width, style, and color.
- Adds a visible boundary. Increases the total size of the box visually and spatially within the document flow.
- **Properties**:
    - Border - width
    - Border - style
    - Border - color

4. Margin

- Space **outside the border**, separating the element from others.
- Margins are transparent.
- Vertical margins can collapse (margin collapsing).
- Adds space *outside* the element's boundary. Does not affect the element's background area, but increases the total *footprint* the element takes up on the page.
- Properties:
    - Margin
    - Margin-top , margin-bottom , etc .

Q - 5 : What is the difference between border-box and content-box box-sizing in CSS? Which is the default?
A - 5 :

Border-Box	**Content-box**
The width and height apply only to the content	The width and height include content, padding, and border
Padding and border are added outside the specified width and height	Padding and border are inside the specified width and height
Actual width = 200px (content) + 40px (padding) + 10px (border) = 250px	**Actual width =** 200px total (content shrinks to fit padding and border)

Q - 6 : What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.
A - 6 : CSS Flexbox is a one-dimensional layout system used to design responsive and flexible layouts.

- It allows elements to be aligned, spaced, and distributed efficiently within a container, even when their sizes are dynamic or unknown.

Flexbox is useful for layout design :

- Makes responsive design easier
- Aligns items horizontally and vertically with less code
- Automatically adjusts spacing between elements
- Handles different screen sizes smoothly
- Reduces the need for floats and complex positioning

Flex Container	**Flex Item**
The parent element that enables Flexbox.	The direct children of a flex container.
Controls alignment and spacing of items	Can grow, shrink, or stay fixed
Uses properties like: - `Flex-direction` - `Justify-content` - `Align-items` - `flex-wrap`	**Use properties like:** - `Flex-grow` - `Flex-shrink` - `Flex-basis` - `align-self`

Example :
IN CSS :
.container {
    display: flex;
}
	Example :
	IN HTML :
	<div class="container">
	    <div class="item">1</div>
	    <div class="item">2</div>
	</div>

Items are arranged in a row (left to right)	**IN CSS :**
	
	.container {
	    display: flex;
	    justify-content: center;
	    align-items: center;
	}
	This centers all flex items horizontally and vertically.

Q - 7: Describe the properties justify-content, align-items, and flex-direction used in Flexbox.
A - 7 :

Properties of Flexbox :	**Purpose :**	**Common Values :**
justify-content	Aligns and distributes flex items along the main axis (defined by `flex-direction`).	**flex-start *(default)*** → items at the start **center** → items centered **flex-end** → items at the end **space-between** → equal space between items **space-around** → equal space around items **space-evenly** → equal space between all items
align-items	Aligns flex items along the cross axis (perpendicular to the main axis).	**stretch *(default)*** → items stretch to fill container **flex-start** → items aligned at start **center** → items centered **flex-end** → items aligned at end **baseline** → items aligned by text baseline
flex-direction	Defines the direction of the main axis, i.e., how flex items are placed.	**row *(default)*** → items arranged left to right **row-reverse** → items arranged right to left **column** → items arranged top to bottom **column-reverse** → items arranged bottom to top

Q - 8 : Explain CSS Grid and how it differs from Flexbox. When would you use Grid overFlexbox?
A - 8 : CSS Grid Layout is a powerful layout system in CSS designed for creating **two-dimensional layouts**—meaning it controls both rows and columns at the same time.

- With Grid, you define a grid container and specify rows and columns using properties like **grid-template-rows** and **grid-template-columns.**

Difference Between CSS Grid and Flexbox :

Feature	CSS Grid	Flexbox
Layout type	2D (rows + columns)	1D (row or column)
Main purpose	Page-level layouts	Component-level layouts
Control	Precise placement of items	Content-based alignment
Item placement	Explicit (grid lines/areas)	Automatic flow
Complexity	Best for complex layouts	Best for simple layouts

Q - 9 : Describe the grid-template-columns, grid-template-rows, and grid-gapproperties. Provide examples of how to use them.

	grid-template-columns	**grid-template-rows**	**grid-gap**
Definition :	Defines the number and **width** of columns in a grid layout	Defines the number and **height** of rows in a grid layout.	Specifies the **space** between rows and columns in a grid layout

Example :	.container {
	  display: grid;
	  grid-template-columns: 200px 1fr 100px;
		.container {
		  display: grid;
		  grid-template-rows: 100px auto 50px;
			.container {
			  display: grid;
			  grid-gap: 20px;
			}

	}	}	
Explanation Of example :	`200px` → fixed width column `1fr` → flexible column that takes remaining space `100px` → fixed width column	`100px` → fixed height row `auto` → adjusts height based on content `50px` → fixed height row	Adds `20px` space between both rows and columns