**MERN Stack Theory Exercise**

**Module 3– Mernstack-CSS & CSS3**

**Css Selectors and Styling**

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

Ans> A CSS selector targets HTML elements to style them.

* **Element selector**: Targets all elements of a type.  
  Example: p { color: blue; } (styles all <p> tags)
* **Class selector**: Targets elements with a specific class.  
  Example: .btn { background: red; } (styles elements with class="btn")
* **ID selector**: Targets a single element with a specific ID.  
  Example: #header { font-size: 20px; } (styles element with id="header")

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

Ans: CSS Specificit determines which CSS rule takes precedence when multiple rules target the same element. It is calculated based on the types of selectors used.

- Inline styles have the highest specificity.

- ID selectors have higher specificity than clas or element selectors.

- Class and pseudo-class selectors have higher specificity than element selectors.

**Question 3: What is the difference between internal, external, and inline CSS?**

**Discuss theadvantages and disadvantages of each approach.**

**Ans =>** Internal CSS : CSS inside a `<style>` tag in the HTML `<head>`.

- Advantages : Easy for single-page sites.

- Disadvantages : Not reusable, increases page size.

External CSS : CSS in a separate `.css` file linked to HTML.

- Advantages : Reusable, cleaner HTML.

- Disadvantages : Requires extra HTTP request.

Inline CSS : CSS applied directly in HTML elements using the `style` attribute.

- Advantages : Quick for specific elements.

- Disadvantages : Not reusable, messy HTML.

**• CSS Box Model**

**Question 1: Explain the CSS box model and its components (content, padding,**

**border,margin). How does each affect the size of an element?**

**Ans =>** The CSS box model defines the structure of an element and how its size is calculated. It consists of four components:

**1. Content** : The actual area where text or other content is displayed. It defines the width and

height of the element.

**2. Padding** : Space around the content, inside the element's border. It increases the overall size of

the element.

**3. Border** : Surrounds the padding (if any). It adds extra space around the content and padding.

**4. Margin** : The outermost space, outside the border. It separates the element from other

elements.

**How they affect size:**

- Width/height only apply to the content.

- Padding and borde add to the total width and height of the element.

- Margin doesn't affect the element's size but controls the space around it.

Total element size = content + padding + border + margin.

**Question 2: What is the difference between border-boxand content-boxbox-sizing**

**inCSS? Which is the default?**

**Ans => content-box (default):** The width and height apply only to the content area. Padding and border are added outside the element's width and height, increasing its total size.

border-box: The width and height include padding and border. The total size remains the same, and padding/border are included within the specified width and height.

**• CSS Flexbox**

**Question 1: What is CSS Flexbox, and how is it useful for layout design? Explain the**

**terms flex-container and flex-item.**

**Ans =>** CSS Flexbox is a layout model that allows items within a container to be dynamically arranged, aligned, and distributed with flexibility. It simplifies the design of complex layouts.

- **Flex-container** : The parent element that holds the flex items. It is set using `display: flex` or

`display: inline-flex`.

- **Flex-item** : The child elements inside the flex-container. They are automatically arranged

according to the flexbox rules.

**Usefulness** : Flexbox makes it easy to create responsive layouts, align items, and distribute space efficiently across rows or columns.

**Question 2: Describe the properties justify-content, align-items, and flex-direction**

**used in Flexbox.**

**Ans =>** 1. justify-content : Aligns flex items along the main axis (horizontal by default). It controls the spacing between items.

- Values: flex-start, flex-end, center, space-between, space-around, space-evenly .

2. align-items : Aligns flex items along the cross axis (vertical by default).

- Values: flex-start, flex-end, center, baseline, stretch.

3. flex-direction : Defines the direction of the main axis and the layout of flex items.

- Values: row (default), row-reverse, column, column-reverse .

**• CSS Grid :--**

**• Question 1: Explain CSS Grid and how it differs from Flexbox. When would you use Grid overFlexbox?**

**Ans => CSS Grid vs. Flexbox:--**

**1. CSS Grid** - A two-dimensional layout system, allowing you to design layouts in rows and columns simultaneously.

- Best for creating complex layouts like grids, dashboards, or page templates.

- Uses `grid-template-rows` and `grid-template-columns` to define the structure.

- Example: `display: grid; grid-template-columns: 1fr 1fr;`.

2. **Flexbox :**

- A one-dimensional layout system, focusing on arranging items in a row or a column (not

both).

- Best for aligning and distributing items within a container (like a navbar or single-row card

layout).

- Uses `flex-direction` (row or column) and alignment properties like `justify-content`.

**When to Use Grid over Flexbox :--**

- Use CSS Grid for complex, two-dimensional layouts (e.g., image galleries, full-page layouts).

- Use Flexbox for simple, one-dimensional layouts (e.g., navbars, button groups, or aligning items in

a single direction).

**• Question 2: Describe the grid-template-columns, grid-template-rows, and grid-gapproperties. Provide examples of how to use them.**

**Ans => Grid Properties:--**

**1. grid-template-columns :--**

- Specifies the number and sizes of columns in a grid layout.

- Can use values like `px`, `%`, `fr` (fractional units), or `auto`.

- Example: `grid-template-columns: 1fr 2fr;` creates two columns, one taking 1 fraction of space, and the other taking 2.

**2. grid-template-rows :--**

- Defines the number and sizes of rows in a grid.

- Similar to `grid-template-columns`, supports flexible units.

- Example: `grid-template-rows: 100px auto;` creates two rows, the first with a fixed height of 100px, the second dynamically filling available space.

**3. grid-gap (or `gap`) :--**

- Adds spacing between grid rows and columns.

- Example: `grid-gap: 20px;` adds 20px spacing between both rows and columns.

**• Responsive Web Design with Media Queries :--**

**• Question 1: What are media queries in CSS, and why are they important for**

**responsivedesign?**

**Ans =>**

**Media Queries in CSS :--**

- **Definition :** Media queries are a CSS feature that applies styles based on device characteristics like screen size, resolution, or orientation.

- **Syntax :** `@media` is used to define conditional styles.

**Example:**

**css**

@media (max-width: 768px) {

body {

font-size: 14px;

}

}

**Importance for Responsive Design :---**

- Adaptability : Ensures a website looks good on all devices (desktop, tablet, mobile).

- User Experience : Adjusts layout, font size, and spacing for better readability and usability.

- Efficiency : Enables a single codebase to handle multiple screen sizes without duplicating styles.

Media queries are a cornerstone of responsive web design!

**• Question 2: Write a basic media query that adjusts the font size of a webpage for**

**screenssmaller than 600px.**

**Ans => Basic Media Query Example :--**

To adjust the font size for screens smaller than 600px:

**css:--**

@media (max-width: 600px) {

body {

font-size: 14px; /\* Smaller font size for small screens \*/

}

}

**Explanation :--**

- `@media (max-width: 600px)`: Targets devices with a screen width of 600px or less.

- `body { font-size: 14px; }`: Reduces the font size to improve readability on smaller screens.

**• Typography and Web Fonts :--**

**• Question 1: Explain the difference between web-safe fonts and custom web fonts.**

**Whymight you use a web-safe font over a custom font?**

**Ans => Web-Safe Fonts vs. Custom Web Fonts :--**

1. **Web-Safe Fonts:--**

- Pre-installed on most devices (e.g., Arial, Times New Roman, Verdana).

- Reliable and load quickly since they don't require external files.

- Example: `font-family: Arial, sans-serif;`.

2. **Custom Web Fonts:---**

- Fonts not pre-installed, loaded via external services (e.g., Google Fonts, Adobe Fonts).

- Offer unique typography but can impact performance due to file loading.

- Example: `font-family: 'Roboto', sans-serif;`.

**Why Use Web-Safe Fonts?**

- Faster loading times (no external requests).

- Greater compatibility across devices.

- Ideal for performance-critical or older-browser-compatible websites.

**• Question 2: What is the font-familyproperty in CSS? How do you apply a custom**

**Google Font to a webpage?**

**Ans => Font-Family Property in CSS :--**

- **Definition** : The `font-family` property specifies the font type for text in a webpage.

- **Syntax:--**

**css**

font-family: "Arial", sans-serif;

- Includes a font stack: primary font and fallback fonts.

**Applying a Custom Google Font :--**

1. **Import the Font:--**

- Add the font's `<link>` from Google Fonts to the `<head>` of your HTML:

**html:--**

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"

rel="stylesheet">

2. **Use in CSS :--**

- Apply the font with `font-family`:

```css

body {

font-family: 'Roboto', sans-serif;

}