Module 3 – MERN Stack – CSS and CSS3

CSS Selectors & Styling :

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

A **CSS selector** is a pattern used to select and style HTML elements. It tells the browser which HTML element(s) the styles should be applied to.

Element Selector: Targets HTML elements by name.

```
p {
  color: blue;
}
```

Class Selector: Targets elements with a specific class attribute.

```
.highlight {
  background-color: yellow;
}
```

ID Selector: Targets an element with a specific ID attribute.

```
#header {
  font-size: 24px;
}
```

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

CSS specificity is a set of rules that determines which style rule is applied when multiple rules match the same element.

Specificity hierarchy:

1. Inline styles (e.g., `style="..."`) → Highest
2. ID selectors
3. Class selectors, attributes, pseudo-classes
4. Element selectors and pseudo-elements

When conflicts occur, the rule with **higher specificity** is applied. If specificity is equal, the **last rule** in the CSS is used.

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

Type	Description	Example	Advantages	Disadvantages
Inline CSS	Styles within an element's `style` attribute	`<p style="color:red; ">`	Quick and specific	Difficult to maintain, not reusable
Internal CSS	Styles within a `<style>` tag in the HTML `<head>`	`<style> p { color:red; } </style>`	Easy for small projects	Increases HTML size
External CSS	Styles in an external `.css` file linked to HTML	`<link rel="stylesheet" href="style.css">`	Reusable, clean HTML	Requires additional HTTP request

CSS Box Model :

Question 1: Explain the CSS box model and its components.

The **CSS Box Model** consists of four parts:

1. **Content**: The actual content (text, image, etc.)
2. **Padding**: Space between content and border
3. **Border**: Surrounds the padding (or content if no padding)
4. **Margin**: Space outside the border, separating elements

These affect the **total size** of an element.

Question 2: What is the difference between `border-box` and `content-box`?

- **content-box** (default): Width includes only the content. Padding and border are added outside.
- **border-box**: Width includes padding and border. Helps in creating consistent layouts.

```
box-sizing: border-box;
```

CSS Flexbox :

Question 1: What is CSS Flexbox?

Flexbox is a layout model that allows responsive alignment and distribution of space among items in a container.

- **flex-container**: The parent element that defines flex context
- **flex-item**: The children inside the container

- **Example:**

```
.container {
  display: flex;              /* enables flexbox */
  justify-content: center;    /* horizontal alignment */
  align-items: center;        /* vertical alignment */
}
```

Question 2: Properties of Flexbox

- **justify-content**: Aligns items horizontally
    - `center`, `flex-start`, `space-between`, etc.
- **align-items**: Aligns items vertically
    - `center`, `stretch`, `flex-end`, etc.
- **flex-direction**: Direction of flex items
    - `row`, `column`, `row-reverse`, `column-reverse`

CSS Grid :

Question 1: What is CSS Grid and how is it different from Flexbox?

CSS Grid

CSS Grid is a **two-dimensional** layout system in CSS. It allows you to create layouts using **rows and columns** at the same time. You define a grid container and specify how items should be placed inside it.

Example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr; /* 3 equal columns */
  grid-template-rows: auto auto;      /* 2 rows */
  gap: 10px;                          /* spacing */
}

.item {
  background: lightblue;
  padding: 20px;
}
```

This creates a **3-column grid layout** where items align neatly in rows and columns.

CSS Grid is a 2D layout system (rows and columns). It's best for **entire page layouts**.

Flexbox

Flexbox is a **one-dimensional** layout system. It works either in a **row (horizontal)** or **column (vertical)** direction at a time. It's great for aligning items and distributing space dynamically.

Example:

```
.container {
  display: flex;
  justify-content: space-between; /* distribute items */
  align-items: center;            /* vertical alignment */
}

.item {
  background: lightgreen;
  padding: 20px;
}
```

Flexbox is 1D (row or column) and better for aligning **items within a container**.

Question 2: Describe Grid properties

- **grid-template-columns**: Defines column sizes
- **grid-template-rows**: Defines row sizes
- **grid-gap**: Adds space between items

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
}
```

Responsive Web Design with Media Queries :

Question 1: What are media queries?

Media queries in **CSS** are rules that let you apply styles only under certain conditions—like screen size, device orientation, or resolution. They make websites **responsive**, meaning the layout and design can adapt to different devices (desktop, tablet, mobile, etc.).

Media queries apply styles based on device size. Essential for **responsive design**.

Common Media Query Features

- **max-width / min-width** → based on screen width
- **orientation** → `portrait` or `landscape`
- **resolution** → e.g., for high-DPI (Retina) screens
- **color** → detects if device supports color

Question 2: Example media query

```
body {
  font-size: 18px;
  background-color: white;
}

@media (max-width: 600px) {
  body {
    font-size: 14px;
    background-color: lightgray;
  }
}
```

Typography and Web Fonts :

Question 1: Web-safe vs Custom Fonts

- **Web-safe fonts**: Default system fonts (e.g., Arial, Times)
    - Faster, no download needed
- **Custom fonts**: Loaded from web (e.g., Google Fonts)
    - Better branding, more stylish

Question 2: font-family and Google Fonts

```
<link
href="https://fonts.googleapis.com/css2?family=Roboto&display=swap"
rel="stylesheet">


body {
  font-family: 'Roboto', sans-serif;
}
```