**Module 4**

**Frontend – CSS and CSS3 CSS Selectors & Styling**

**Theory Assignment:**

**CSS Selectors & Styling**

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 specific HTML elements on a web page. It tells the browser which element(s) the CSS rules should apply to.

**Types of CSS Selectors:**

* **Element Selector**

Targets all HTML elements of a given type.

* **Class Selector**

Targets all elements with a specific class attribute. Begins with a dot (.).

* **ID Selector**

Targets a single element with a specific ID attribute. Begins with a hash (#).

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 CSS rule applies** when there are conflicting rules targeting the same HTML element. The more specific the selector, the higher priority it has.

**How Specificity Works**

Specificity is calculated based on the types of selectors used. It's often thought of as a **4-part value**:

a: Inline styles (e.g., style="...") → Highest

b: Number of ID selectors

c: Number of class selectors, attributes, and pseudo-classes

d: Number of element (type) selectors and pseudo-elements

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

CSS can be applied to HTML documents in **three main ways**. Each method has its **use cases**, **advantages**, and **disadvantages**.

**1. Inline CSS**

CSS is written **directly inside the HTML element** using the style attribute.

**Advantages:**

* Quick and easy for small changes.
* Useful for testing or single-use styles.
* Overrides internal and external styles.

**Disadvantages:**

* Not reusable.
* Clutters HTML code.
* Harder to maintain or scale.
* Poor separation of content and style.

**2. Internal CSS**

CSS is placed **inside a <style> tag** within the <head> section of an HTML document.

**Advantages:**

* Easier to manage styles for a single page.
* No external file needed.
* Can override external styles if placed later in the source.

**Disadvantages:**

* Styles can't be reused across multiple pages.
* Increases HTML file size.
* Not ideal for large projects.

**3. External CSS**

CSS is written in a **separate .css file** and linked using the <link> tag.

**Advantages:**

* **for large websites.**
* Promotes reusability (can style multiple pages).
* Keeps HTML clean and maintainable.
* Loads faster after the first request (cached by browser).

**Disadvantages:**

* One extra HTTP request (can slightly delay loading if not cached).
* Doesn’t apply styles if the external file fails to load.

**CSS Box Model**

1: Explain the CSS box model and its components (content, padding, border, margin). How does each affect the size of an element?

**CSS Box Model:**

**Content**: The text or image inside the box.

**Padding**: Space **inside** the box, around the content.

**Border**: The line around the box.

**Margin**: Space **outside** the box, between this and other boxes.

**Total size = Content + Padding + Border**  
(Margin doesn’t add to size; it’s just outer space).

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

**content-box (default)**: width & height include **only content**. Padding and border are **added outside.**

**border-box**: width & height include **content + padding + border**.

**content-box** is the default box-sizing in CSS.

1: What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.

**CSS Flexbox:**

* Flexbox (Flexible Box Layout) is a CSS layout model used to arrange elements in a **row or column**, making it easier to design **responsive and flexible layouts**.
* It automatically adjusts the size and position of items to fit different screen sizes.

**Why is it useful?**

* Aligns items **horizontally & vertically** easily.
* Handles **spacing, resizing, and wrapping** without complex CSS.
* Perfect for **navigation bars, grids, and responsive designs**.

**Key Terms:**

* **Flex Container**: The parent element where Flexbox is applied using display: flex. It defines how child items behave.  
  Example:
* **Flex Items**: The **direct children** of the flex container that are arranged according to flex rules.

**In simple words**:

* *Flex container* = box holding the items.
* *Flex items* = the boxes inside the container.

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

**1. justify-content**

* Aligns items **horizontally** (along the main axis).
* **Values:**
  + flex-start → items start from the left.
  + flex-end → items at the right end.
  + center→ items in the center.
  + space-between → equal space *between* items.
  + space-around → equal space *around* items.

**2. align-items**

* Aligns items **vertically** (along the cross axis).
* **Values:**
  + flex-start → top.
  + flex-end → bottom.
  + center → center vertically.
  + stretch → stretch items to fill height.

**3. flex-direction**

* Sets the **direction of items** in the container.
* **Values:**
  + row (default) → left to right.
  + row-reverse → right to left.
  + column → top to bottom.
  + column-reverse → bottom to top.

**Tip to remember:**

* justify-content → main axis (horizontal by default).
* align-items → cross axis (vertical by default).
* flex-direction → changes the main axis.

**CSS Grid**

1: Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?

**CSS Grid vs Flexbox**

**CSS Grid**:

* A layout system for creating **2D layouts** (rows **and** columns).
* Works well for **full page/grid-based designs**.

**Flexbox**:

* A layout system for **1D layouts** (row **or** column).
* Great for aligning items **in a single line**.

**Key Difference**:

* **Grid** → controls **rows & columns together** (2D).
* **Flexbox** → controls **items in a single direction** (1D).

**When to use Grid?**

* Use **Grid** for complex, structured layouts (e.g., dashboards, page layouts).
* Use **Flexbox** for simple, linear alignments (e.g., navbars, toolbars).

2: Describe the grid-template-columns, grid-template-rows, and grid- gap properties. Provide examples of how to use them.

**grid-template-columns**

* Defines the **number and size of columns** in a grid.
* Example:  
  grid-template-columns: 100px 200px auto;  
  (Creates 3 columns: 100px, 200px, and auto width).

**grid-template-rows**

* Defines the **number and size of rows** in a grid.
* Example:  
  grid-template-rows: 50px auto 100px;  
  ( Creates 3 rows: 50px, auto height, and 100px).

**grid-gap (or gap)**

* Sets the **space between rows and columns** in a grid.
* Example:  
  grid-gap: 20px;  
  (Adds 20px gap between all grid items).

**In short**:

* grid-template-columns → sets **columns**.
* grid-template-rows → sets **rows**.
* grid-gap → sets **spacing** between grid cells.

**Responsive Web Design with Media Queries**

1: What are media queries in CSS, and why are they important for responsive design?

* Media queries in CSS allow you to **apply styles based on device characteristics** like screen size, resolution, or orientation.
* They help design websites that **adapt to different devices** (mobiles, tablets, desktops).

**Why are they important:**

* Enable **responsive design** (content looks good on all screen sizes).
* Avoids horizontal scrolling and ensures a better **user experience**.

**In short:**  
**Media Queries = Make websites responsive & mobile-friendly**.

2: Write a basic media query that adjusts the font size of a webpage for screens smaller than 600px

Media queries help create **responsive designs** by changing styles for different screen sizes.

Here’s the **basic media query:**

**/\* Default font size \*/**

body {

font-size: 16px;

}

**/\* For screens smaller than 600px \*/**

@media (max-width: 600px) {

body {

font-size: 14px;

}

}

**What it does:**

* On screens **larger than 600px**, the font size is **16px**.
* On screens **600px or smaller**, the font size becomes **14px** (better for mobiles).

**Typography and Web Fonts**

1: Explain the difference between web-safe fonts and custom web fonts. Why might you use a web-safe font over a custom font?

**Web-safe fonts:**

* Fonts that are **pre-installed on most devices** (Windows, Mac, Android, etc.).
* They don’t require downloading because they are already available on users’ systems.
* **Examples:** Arial, Times New Roman, Verdana, Courier New.

**Custom web fonts:**

* Fonts that are **not pre-installed** and are loaded from the web using @font-face or services like **Google Fonts**.
* **Examples:** Roboto, Open Sans, Lato.

**Why use web-safe fonts?**

* **Faster loading** (no extra font files to download).
* **Better compatibility** across older browsers and devices.
* Useful when **performance is critical** or **internet speed is slow**.

**Why use custom fonts?**

* Offers **unique typography** and branding.
* More **style and design flexibility.**

2: What is the font-family property in CSS? How do you apply a custom Google Font to a webpage?

font-family in CSS is used to **change the style of text** by setting a font. You can also give a list of fonts as backup.

***Example:***

body {

font-family: Arial, sans-serif;

}

* **How to use a Google Font:**

Go to **Google Fonts**, choose a font (e.g., *Roboto*).  
Copy the link they give and paste it in the <head> of your HTML:

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

Set it in CSS:

body {

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

}

**In short:**

* font-family = changes the text font.
* Google Fonts = add link + use in CSS.