**Module 3 – Frontend – CSS and CSS3**

**CSS Selectors & Styling**

**Theory Assignment**

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

* A **CSS selector** is a pattern used to target (select) HTML elements so that styles can be applied to them.

1. **Element Selector :-**

p {

color: blue;

font-size: 16px;

}

1. **Class Selector :-**

.myClass {

background-color: yellow;

padding: 10px;

}

1. **ID Selector :-**

#myId {

color: red;

font-weight: bold;

}

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

* **Specificity** is a set of rules that determine **which CSS style is applied** when multiple selectors target the same HTML element.

**Think of it as a priority system**:

1. **Universal selector (\*)**, element selectors (div, p, h1) → **lowest**
2. **Class selectors (.class),** attribute selectors ([type="text"]), pseudo-classes (:hover, :first-child)
3. **ID selectors (#id)**
4. **Inline styles (**style**="..." inside HTML)** → highest
5. What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

## 1. Inline CSS

## Written directly in the style attribute of an HTML element.

**Advantages:**

Quick and easy for small changes.

**Disadvantages:**

Makes HTML messy and harder to maintain.

**2. Internal CSS**

Written inside <style> tags within the <head> section of the HTML file.

**Advantages:**

Styles are applied to the whole page, not just one element.

**Disadvantages:**

Styles cannot be reused across multiple pages.

**3. External CSS**

Written in a separate .css file and linked using <link> inside <head>.

**Advantages:**

Reusable across multiple pages → consistency in design.

**Disadvantages:**

Requires an extra file (not self-contained).

**CSS Box Model**

**Theory Assignment**

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

* Every HTML element is treated as a **rectangular box**.

The box model defines **how space is calculated** around each element.

It consists of **4 main components** (inside → out):

* 1. **Content :-**

The **actual text, image, or other content** inside the element.

Its size is controlled by properties like width, height, font-size.

* 1. **Padding :-**

The **space between the content and the border**.

Increases the background area but does **not overlap the content**.

* 1. **Border :-**

The **line that wraps around the padding and content**.

Controlled by properties like border-width, border-style, border-color.

* 1. **Margin :-**

The **space outside the border**, creating distance between this element and others.

Transparent (no background color).

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

* In CSS, the box-sizing property defines how the total width and height of an element are calculated.
* **content-box :-**

**Width/height apply only to the content area**.

Padding and border are **added outside** the specified width and height.

This means the element’s total size = content + padding + border + margin.

* **border-box :-**

**Width/height include content, padding, and border**.

The declared width is the final size of the element (except margin).

This makes layouts easier to control.

* + The default box-sizing in CSS is **content-box**.

**CSS Flex box**

**Theory Assignment**

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

* **Flexbox (Flexible Box Layout Module)** is a CSS layout model designed to make it easier to arrange elements in a **responsive and flexible way**, even when their sizes are unknown or dynamic.

### ****Flex Container :-****

* + It defines a flex formatting context.
  + Controls how its children (flex-items) are placed.

**Flex Item :-**

* They follow the rules defined by the container.
* Can be stretched, shrunk, or ordered using flexbox properties.

**Flexbox is Useful in Layout Design :**

* **Responsive Design**: Items adjust to different screen sizes easily.
* **Easy Alignment**: Centering content both horizontally & vertically is simple.
* **Space Distribution**: Even spacing between items without extra margins.
* **Ordering**: Change item order without changing HTML structure.
* **Flexibility**: Items can grow or shrink based on available space.

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

* **1. flex-direction :-**

Defines the **main axis** along which flex items are placed inside the flex container.

* row → items in a row (left to right, default)
* row-reverse → items in a row (right to left)
* column → items in a column (top to bottom)
* column-reverse → items in a column (bottom to top)

**2. justify-content :-**

Controls how flex items are **distributed along the main axis** (horizontal if row, vertical if column).

* flex-start → items packed at start (default)
* flex-end → items packed at end
* center → items centered
* space-between → equal space between items
* space-around → equal space around items
* space-evenly → equal space between and around items

**3. align-items :-**

Controls how flex items are **aligned along the cross-axis** (perpendicular to the main axis).

* stretch → items stretch to fill container (default)
* flex-start → items align to top (if row) or left (if column)
* flex-end → items align to bottom (if row) or right (if column)
* center → items centered along cross-axis
* baseline → items align by text baseline

**CSS Grid**

**Theory Assignment**

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

* CSS Grid is a **2D layout system** in CSS that lets you create layouts using rows **and** columns simultaneously.

Flexbox is a **1D layout system** (one direction at a time: row **or** column).  
It distributes space along a single axis and aligns items relative to each other.

**Use Flexbox when :-**

* You’re working **in one dimension** (row OR column).
* Items should align relative to each other.
* Example: navbars, form controls, buttons, centered layouts.

**Use Grid when :-**

* You need a **two-dimensional layout** (rows AND columns).
* You want precise control over placement.
* Example: web page layouts, image galleries, dashboards, magazine-style designs.

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

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

Defines the **number of columns** and their **sizes** in the grid.

**Example :**

.container {

display: grid;

grid-template-columns: 100px 200px auto;

}

* First column = 100px
* Second column = 200px
* Third column = takes up the remaining space (auto)

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

Defines the **number of rows** and their **sizes**.

**Example :**

.container {

display: grid;

grid-template-rows: 100px 200px auto;

}

* First row = 100px
* Second row = 200px
* Third row = takes remaining space

**3. grid-gap :-**

Defines the **spacing between rows and columns.**

**Example :**

.container {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

grid-template-rows: 100px 100px;

grid-gap: 20px; /\* gap between rows and columns \*/

}

* Adds 20px space between all grid cells.

**Responsive Web Design with Media Queries**

**Theory Assignment**

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

* Media queries are a CSS technique that lets you apply styles based on the device’s characteristics, such as:
* Screen **width** / **height**
* **Orientation** (portrait or landscape)
* **Resolution** (DPI, retina displays)
* Device type (screen, print, speech, etc.)

**Why are Media Queries Important for Responsive Design?**

* **Adapt layouts to different devices** → desktop, tablet, mobile.
* **Improve readability** → adjust font sizes, spacing for small screens.
* **Optimize performance** → hide or rearrange elements for mobile.
* **Better user experience** → ensures a consistent, usable design everywhere.
* **Mobile-first design** → build for small screens first, then scale up.

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

* body {

font-size: 18px;

}

/\* Media query: for screens smaller than 600px \*/

@media (max-width: 600px) {

body {

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

}

}

**How it works:**

* On desktops or tablets (>600px) → text is 18px.
* On mobile devices (≤600px wide) → text shrinks to 14px.

**Typography and Web Fonts**

**Theory Assignment**

**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, macOS, Linux, Android, iOS).

Since they’re already on the system, the browser doesn’t need to **download** them.

**Examples:**

* Arial
* Verdana
* Times New Roman
* Georgia
* Courier New
* Trebuchet MS

**Custom Web Fonts :-**

Fonts that are **not guaranteed** to exist on user devices, but are loaded from the web (usually via @font-face or services like Google Fonts).

**Why Use Web-Safe Fonts Over Custom Fonts?**

* **Performance matters** → critical for fast-loading sites (news sites, blogs, government portals).
* **Email templates** → many email clients block custom fonts.
* **Fallback reliability** → when custom fonts fail to load.
* **Simple/minimalist design** → when you don’t need fancy typography.

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

* The font-family property in CSS specifies the **font(s)** used to display text.

You can provide a **list of fonts** → if the first isn’t available, the browser uses the next one.

**Applying a Custom Google Font:**

**Step 1**: Import the font

Using <link> in HTML <head>

<head>

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

</head>

**Step 2**: Apply with font-family

body {

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

}