Module 3 – Frontend – CSS and CSS3

CSS Selectors & Styling

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

=> CSS selectors are patterns used to select and style HTML elements. They allow developers to target specific elements or groups of elements based on various criteria, such as element type, class, ID, attributes, and hierarchy within the document structure.

Element Selector

-> Selects elements based on their tag name (e.g., p, div, h1).

-> Example:

```
div {
  text-align: center;
  color: blue;
}
```

Universal Selector

-> Selects all elements on the page using the asterisk `*`.

-> Example:

```
*{
background-color:blue;
color:white;
}
```

CSS Id Selector

-> Selects a single element with a specific ID. The id of an element is unique within a page, so the id selector is used to select one unique element , To select an element with a specific id, write a hash (#) character, followed by the id of the element.

-> Example:

```
#my-id {
  text-align: center;
  color: red;
}
```

CSS Class Selector

-> Selects elements with a specific class. To select elements with a specific class, write a period (.) character, followed by the class name.

-> Example:

```
.center {
  text-align: center;
  color: red;
007D
```

CSS Grouping Selector

-> The grouping selector selects all the HTML elements with the same style definitions. Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

-> Example:

```
h1, div, p {
  text-align: center;
  color: red;
}
```

Attribute Selector

-> Selecting Elements with a Specific Attribute

-> Example:

```
input[type="checkbox"] {
margin-right: 5px;
}
```

Pseudo-classes and Pseudo-elements

-> Pseudo-classes: Select elements based on their state or position in the document (e.g., :hover, :first-child, :nth-child()).

-> Pseudo-elements: Select and style a part of an element (e.g., ::before, ::after).

Combinator Selectors

-> Descendant Selector (Whitespace): Selects an element that is a descendant of another element.

-> Child Selector (>): Selects direct children of an element.

-> Adjacent Sibling Selector (+): Selects an element that is immediately preceded by a sibling element.

-> General Sibling Selector (~): Selects siblings of an element that share the same parent.

Q2. Explain the concept of CSS specificity. How do conflicts between multiple stylesget resolved?

=> When you write CSS to style a webpage, sometimes more than one rule tries to change the same thing — like the color of a heading. The browser needs a way to decide which rule to follow. That's where specificity comes in.

1. Inline styles (written directly in the HTML) are the strongest.

-> Example:

```
<p style="color: red;">This is red</p>
```

2. ID selectors (#myId) are strong.

-> Example:

```
#myId {
  color: blue;
}
```

3. Class selectors (.myClass), attributes ([type="text"]), and things like :hover are in the middle.

-> Example:

```
.myClass {
  color: green;
}
```

4. Element tags (p, h1, div) are the weakest.

-> Example:

```
p {
  color: black;
}
```

5. !important overrides everything — it's like saying "I don't care, do this!"

Example:

Imagine you have this Html.

Hello!

And This CSS

```
p {
  color: black;
}
.text {
  color: green;
}
#para {
  color: blue;
}
```

What color will the <p> be?

-> The inline style (color: red) wins because it has the highest specificity.

!important (strongest)

Inline styles

ID selectors

Class selectors / attributes / pseudo-classes

Tag selectors (like p, h1, etc.)

Universal selector (*) (weakest)

Q3. What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

=> * Inline CSS

CSS is written directly inside the HTML tag using the style attribute.

Example:

This is blue text

Advantages:

- Quick and easy for small changes.

- Styles only one element — very specific.

Disadvantages:

- Hard to maintain if you have many elements.

- Makes your HTML messy and long.

- Not reusable — you must write the same style again for other elements.

* Internal CSS

CSS is written inside a <style> tag in the <head> section of the HTML file.

Example:

```
<head>
    <style>
        p {
          color: green;
        }
    </style>
</head>
```

Advantages:

- Easier to manage than inline CSS.
- Good for small websites or single pages.
- Styles are all in one place (at the top).

Disadvantages:

- If you have many HTML pages, you must copy the style into each one.
- Not great for large websites.

*External CSS

CSS is written in a separate file (like style.css), and linked to the HTML using a <link> tag.

Example:

```
<head>
  <link rel="stylesheet" href="style.css">
</head>

p {
        color: red;
}
```

Advantages:

- Best for large websites.
- You can reuse the same CSS file on many pages.
- Keeps HTML clean and easy to read.
- Easy to update — change one file and it updates everywhere.

Disadvantages:

- Needs an extra file, so one more thing to manage.

- If the CSS file is missing or doesn’t load, styles won’t appear.

CSS Box Model

Q1. Explain the CSS box model and its components (content, padding, border,margin). How does each affect the size of an element?

=> Box Model:

Every HTML element on a web page is like a box, and it has 4 parts:

1. Content

- This is where text or images appear.

- It’s the main part of the box.

2. Padding

- Space inside the box, around the content.

- Makes space between the content and the border.

3. Border

- A line around the padding and content.

- You can change its color, size, and style.

4. Margin

- Space outside the box.

- Creates distance between this box and other elements.

How does it affect the size ?

- Total size of an element = Content + Padding + Border + Margin

- So, if you add padding, border, or margin, the element takes up more space on the page.

Example:

width: 100px;

padding: 10px;

border: 5px;

margin: 20px;

Total space used =

100 (content) + 20 (padding) + 10 (border) + 40 (margin) = 170px wide.

Q2. : What is the difference between border-box and content-box box-sizing in CSS? Which is the default?

=> Box-Sizing

- It controls how the size of an element is calculated.

1. Content-Box (Default).

- width = only the content

- Padding and border are added outside the width

- So the element becomes bigger than what you set.

2. Border-Box

- width = content + padding + border

- Everything fits inside the size you set

- So the element stays the size you set.

-> content-box is the default in CSS.

CSS Flexbox

Q1. What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.

=> Flex Box:

-Flexbox (Flexible Box) is a layout system in CSS that helps you arrange items in a row or column, and control spacing, alignment, and size easily — even when the screen size changes.

- Flex Container

The main box (parent) that holds the items. You set it like this: display: flex;

-Flex Items

The boxes inside the container (children). These are arranged by Flexbox.

-Why use Flexbox?

- Easy to align items left, right, center, top, or bottom.
- Works great on all screen sizes.
- No need for complicated layout tricks.

Q2. Describe the properties justify-content, align-items, and flexdirection used in Flexbox.

=> flex-direction

- Tells Flexbox how to arrange items.
- row (default) → items go left to right
- column → items go top to bottom

justify-content

- Aligns items left to right (main axis).
- Controls space between items.
- Examples:
    - flex-start → items go to the left
    - center → items go to the middle
    - space-between → space between items

align-items

- Aligns items top to bottom (cross axis).

- Examples:

    - flex-start → items align to the top

    - center → items align in the middle

    - stretch → items fill the height

CSS Grid

Q1. : Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?

=> CSS Grid

- CSS Grid is a tool to create layouts using rows and columns — like a grid or table.

It helps place items both horizontally and vertically in exact spots.

- Gird And Flex-Box Different

    - Flex-Box :- arranges items in a single row or column (one direction).

    - Grid : arranges items in rows AND columns (two directions).

- When to use Grid

    - When you want complex layouts with rows and columns.

    - For example: page layouts, photo galleries, dashboards.

Q2. : Describe the grid-template-columns, grid-template-rows, and grid-gap properties. Provide examples of how to use them.

=> 1. grid-template-columns

- Defines how many columns and their widths in a grid.

-Example:

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

-Creates 3 columns: first 100px wide, second 200px, third 100px.

2. grid-template-rows

- Defines how many rows and their heights in a grid.

- Example:

- grid-template-rows: 50px 100px;

- Creates 2 rows: first 50px tall, second 100px tall.

3. grid-gap (or gap)

- Adds space between rows and columns.

-Example:

- grid-gap: 10px;

- Adds 10px space between all rows and columns.

Responsive Web Design with Media Queries

Q1. What are media queries in CSS, and why are they important for responsive design?

=> Media queries are CSS rules that apply styles only when the screen size or device type matches certain conditions.

- Why are they important?

- They help make websites look good on all devices — phones, tablets, and computers — by changing the layout and styles based on screen size.

- Example:
    - @media (max-width: 600px) {
        body {
            background-color: lightblue;
        }
    }
    - This changes the background to light blue only on small screens (600px wide or less).

Q2. Write a basic media query that adjusts the font size of a webpage for screens smaller than 600px

=> @media (max-width: 600px) {
    body {
        font-size: 14px;
    }
}

Typography and Web Fonts

Q1. 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

- These are common fonts already installed on most devices.

- Example: Arial, Times New Roman, Verdana

- They load fast and work on almost all browsers.

Custom Web Fonts

- These are downloaded from the web, like Google Fonts.

- Example: Roboto, Lobster, Open Sans

- They look unique, but may load slower.

Why use Web-Safe Fonts?

- They are fast, reliable, and don’t depend on internet speed.

- Good for performance and simple websites.

Q2. : What is the font-family property in CSS? How do you apply a custom Google Font to a webpage?

=> Font-Family in CSS

- The font-family property sets the style of text on a webpage.

- You can list one or more fonts.

- Example:

    - body {

        font-family: Arial, sans-serif;

    }

- How to use a custom Google Font?

    - Go to fonts.google.com

    - Choose a font (e.g., Roboto)

    - Copy the link they give you

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

    - Paste it in your HTML <head>

- Use it in your CSS:

- body {

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