Module 3 – Frontend – CSS

Theory Assignment:

CSS Selectors & Styling

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

  CSS selector indicates the HTML element you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to it's id, class, type, attribute etc.

  - **Examples:**

  1. **Element Selector:** The element selector selects the HTML element by their tag name.

```
<head>
   <style>
      /* h1 element selected here */
      h1 {
         color: red;
         text-align: center;
      }
</style>
</head>

<body>
   <h1>Element</h1>
</body>
</html>
```

2. **Class selector:** If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector. The basic class selector applies styles to all elements with a specified class name. Use a period (.) before the class name in the CSS.

```
<style>
.center {
text-align: center;
color: blue;
}
</style>

<body>
<h1 class="h1">This heading is not affected</h1>
<p class="center">This paragraph is blue and center-aligned.
</p>
</body>
```

3. **ID Selectors:** The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element. It is written with the hash character (#), followed by the id of the element.

```
 <style>
    #header {
        color: blue;
        font-size: 24px;
        text-align: center;
     }
   </style>

<body>
   <div id="header">ID selector example</div>
</body>
```

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

  Ans.
  Ever wonder why they are called “cascading” style sheets? CSS allows you to apply several style sheets to the same document, which means there are bound to be conflicts.

  It's basically a **priority system** that helps resolve conflicts between styles. CSS **specificity** is a method which is used by the browsers to determine which CSS rule applies when multiple rules are targeting the same HTML element.

  Once the applicable style sheet has been chosen, there may still be conflicts; therefore, the cascade continues at the rule level. When two rules in a single style sheet conflict, the type of selector is used to determine the winner. The more specific the selector, the more weight it is given to override conflicting declarations.

  Priority for different rules follows in this order:
  1 → Inline styles
  2 → ID selectors
  3 → Classes, attributes, and pseudo-classes
  4 → Element selectors and pseudo-elements

  Finally, there is rule order like if there are conflicts within style rules of identical weight, whichever one comes last in the list “wins.” Or it is the one which overwrites the previous ones.

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

  Ans.

  1. **Inline CSS** is written directly in the HTML element using the **`style`** attribute, applying styles only to that specific element.
  2. **Internal CSS** is placed within a **`<style>`** tag in the HTML **`<head>`,** affecting styles on that specific page.
  3. **External CSS** is stored in a separate (**`.css`**) file and linked to the HTML, which allowing styles to be reused across multiple pages.

Type	**Advantages**	**Disadvantages**
Inline CSS	1. It is quick to apply 2. It has high specificity 3. It is useful for dynamic styling 4. It doesn't depend on external file 5. It has immediate effect	1. It is not reusable 2. It clutters HTML 3. It is hard to maintain 4. It has poor readability 5. It slows down debugging because of so many properties
Internal CSS	1. It is centralized for a single page 2. It doesn't need extra file 3. It is easy testing 4. It overrides external styles 5. It has moderate maintainability	1. It is not reusable across pages 2. It increases page size 3. It mixes style with content 4. It can cause duplication 5. It becomes slower for large sites
External CSS	1. It is reusable across pages 2. HTML looks cleaner 3. It has better maintainability 4. Faster page load after caching 5. Separation of concerns	1. It requires HTTP request 2. It depends on file loading 3. It can cause flash of unstyled content 4. It has more complex structure 5. It requires organized file management

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 easiest way to think of the box model is that browsers see every element on the page (both block and inline) as being contained in a little rectangular box. You can apply properties such as borders, margins, padding, and backgrounds to these boxes, and even reposition them on the page.

  <u>The Element Box:</u>

  1. **Content area**

     At the core of the element box is the content itself.

  2. **Inner edges**

     The edges of the content area are referred to as the inner edges of the element box. This is the box that gets sized when you apply width and height properties.

  3. **padding**

     The padding is the area held between the content area and an optional border. In the diagram, the padding area is indicated by a yellow-orange color. Padding is optional.

  4. **Border**

     The border is a line (or stylized line) that surrounds the element and its padding. Borders are also optional.

  5. **Margin**

     The margin is an optional amount of space added on the outside of the border. In the diagram, the margin is indicated with light-blue shading, but in reality, margins are always transparent, allowing the background of the parent element to show through.

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

  In CSS, box-sizing is a property that defines how the width and height of an element are calculated. There are two main values for box-sizing: content-box and border-box.

  Content-Box:

  The default value is content-box. When you set box-sizing: content-box;, the width and height of an element only include the content area. This means that any padding, borders, or margins are added to the width and height, making the overall size of the element larger than the specified width and height.

  For example:

  If you set width: 100px; and padding: 10px; on an element with box-sizing: content-box;, the total width of the element would be 100px + 20px (10px padding on each side) = 120px.

  Border-Box

  When you set box-sizing: border-box;, the width and height of an element include the content area, padding, and borders. This means that any padding or borders are subtracted from the width and height, making the overall size of the element the exact size you specify.

  Using the same example:

  If you set width: 100px; and padding: 10px; on an element with box-sizing: border-box;, the total width of the element would still be 100px, with the content area being 80px (100px - 20px padding).

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.:

  What is CSS Flexbox?

  CSS Flexbox, also known as Flexible Box Layout, is a layout mode in CSS that makes it easier to design complex layouts. It's a one-dimensional layout system that allows you to create flexible, responsive, and dynamic layouts that adapt to different screen sizes and devices.

  How is Flexbox useful for layout design?

  Flexbox is useful for layout design in several ways:

  1. **Easy horizontal and vertical alignment**: Flexbox makes it simple to align elements horizontally and vertically, without using hacks or workarounds.
  2. **Flexible and responsive layouts**: Flexbox allows you to create layouts that adapt to different screen sizes and devices, making it perfect for responsive web design.
  3. **Simplified layout code**: Flexbox reduces the need for complex layout code, making it easier to maintain and update your CSS.
  4. **Efficient use of space**: Flexbox helps you make the most of the available space, reducing the need for unnecessary margins, padding, or positioning.
  5. **Improved accessibility**: Flexbox allows you to create layouts that are more accessible to users with disabilities, by providing a clear and consistent layout structure.

  Key benefits of using Flexbox

  (1) **Flexibility**: Flexbox allows you to create flexible layouts that adapt to different screen sizes and devices.
  (2) **Ease of use**: Flexbox is relatively easy to learn and use, especially for developers who are already familiar with CSS.
  (3) **Improved performance**: Flexbox can improve the performance of your website or application, by reducing the need for complex layout code.

  Overall, Flexbox is a powerful tool for creating flexible and responsive layouts. Its intuitive syntax and flexible layout options make it a popular choice among web developers.

Flex-Container:

A **Flex-Container** is the parent element that contains the flex items. It's the element that has display: flex; or display: inline-flex; applied to it. When an element becomes a flex-container, it becomes a flexible container that can hold multiple flex-items.

The flex-container is responsible for:

- Defining the direction of the flex-items (e.g., horizontal or vertical)
- Controlling the alignment of the flex-items
- Managing the space between and around the flex-items

Flex-Item:

A **Flex-Item** is a child element of the flex-container. It's an element that is directly contained within a flex-container. Flex-items are the elements that will be laid out using Flexbox.

Flex-items can be any type of element, such as a div, span, img, or even a button. When an element becomes a flex-item, it becomes a flexible element that can grow or shrink to fit the available space in the flex-container.

The flex-item is affected by the properties set on the flex-container, such as:

- flex-direction
- justify-content
- align-items and more….

Relationship between Flex-Container and Flex-Item:

The flex-container and flex-item have a parent-child relationship. The flex-container is the parent, and the flex-items are the children. The flex-container controls the layout of the flex-items, and the flex-items respond to the properties set on the flex-container.

- **Question 2: Describe the properties justify-content, align-items, and flex direction used in Flexbox.**
  Ans.
  Justify-Content

  The justify-content property in Flexbox is used to align flex-items horizontally along the main axis of the flex-container. The main axis is determined by the flex-direction property.

  1. **Possible values for justify-content:**

- flex-start: Aligns flex-items to the start of the main axis.
- flex-end: Aligns flex-items to the end of the main axis.
- center: Aligns flex-items to the center of the main axis.
- space-between: Distributes flex-items evenly along the main axis, with equal space between each item.
- space-around: Distributes flex-items evenly along the main axis, with equal space around each item.
- space-evenly: Distributes flex-items evenly along the main axis, with equal space between, around, and including the first and last items.

Example:

```
CSS
.container {
  display: flex;
  justify-content: space-between;
}
```

This will distribute the flex-items evenly along the main axis, with equal space between each item.

2. **Align-Items**

   The align-items property in Flexbox is used to align flex-items vertically along the cross-axis of the flex-container. The cross-axis is perpendicular to the main axis.

   Possible values for align-items:

- flex-start: Aligns flex-items to the start of the cross-axis.
- flex-end: Aligns flex-items to the end of the cross-axis.

- center: Aligns flex-items to the center of the cross-axis.
- baseline: Aligns flex-items to the baseline of the cross-axis.
- stretch: Stretches flex-items to fill the cross-axis.

Example:

```css
.container {
 display: flex;
 align-items: center;
}
```

This will align the flex-items to the center of the cross-axis.

3. Flex-Direction

The flex-direction property in Flexbox is used to define the direction of the main axis of the flex-container.

Possible values for flex-direction:

- row: Sets the main axis to horizontal (left to right).
- row-reverse: Sets the main axis to horizontal (right to left).
- column: Sets the main axis to vertical (top to bottom).
- column-reverse: Sets the main axis to vertical (bottom to top).

Example:

```css
.container {
 display: flex;
 flex-direction: column;
}
```

This will set the main axis to vertical, and the flex-items will be stacked on top of each other.

By combining these properties, you can create complex and flexible layouts using Flexbox.

CSS Grid

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

  Ans.

  CSS Grid:

  CSS Grid is a two-dimensional layout system that allows you to create complex, grid-based layouts using HTML and CSS. It provides a powerful way to create responsive, flexible, and dynamic layouts that can be easily maintained and updated.

  ➢ **Key Features of CSS Grid:**

1. **Grid Container**: The parent element that contains the grid items.
2. **Grid Items**: The child elements that are laid out within the grid container.
3. **Grid Template**: A set of rows and columns that define the structure of the grid.
4. **Grid Cells**: The individual cells within the grid that contain the grid items.
5. **Grid Tracks**: The rows and columns that make up the grid.

  ➢ **How CSS Grid Differs from Flexbox:**

A. **Two-Dimensional vs One-Dimensional**: CSS Grid is a two-dimensional layout system, whereas Flexbox is a one-dimensional layout system. This means that CSS Grid can handle both rows and columns, while Flexbox can only handle one axis at a time.

B. **Grid-Based vs Flex-Based**: CSS Grid is based on a grid system, where elements are placed within a predefined grid structure. Flexbox, on the other hand, is based on a flexible box model, where elements are laid out within a flexible container.

C. **More Control over Layout**: CSS Grid provides more control over the layout of elements, allowing you to define specific grid areas, rows, and columns. Flexbox, while flexible, can be more limited in terms of layout control.

D. **Better Support for Complex Layouts**: CSS Grid is better suited for complex, grid-based layouts, such as magazine-style layouts or complex web applications. Flexbox is better suited for simpler, one-dimensional layouts.

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

1. **Complex, Grid-Based Layouts**: Use Grid when you need to create complex, grid-based layouts that require precise control over rows and columns.
2. **Two-Dimensional Layouts**: Use Grid when you need to create two-dimensional layouts that involve both rows and columns.
3. **Grid-Based Components**: Use Grid when building grid-based components, such as image galleries, data tables, or magazine-style layouts.
4. **Responsive Design**: Use Grid when you need to create responsive designs that adapt to different screen sizes and devices.
5. **Precise Control over Layout**: Use Grid when you need precise control over the layout of elements, including the ability to define specific grid areas, rows, and columns.

- **Scenarios where Grid is a better choice than Flexbox:**

I. **Creating a Magazine-Style Layout**: Grid is better suited for creating complex, grid-based layouts that involve multiple rows and columns.

II. **Building a Data Table**: Grid is better suited for building data tables that require precise control over rows and columns.

III. **Designing a Responsive Image Gallery**: Grid is better suited for designing responsive image galleries that adapt to different screen sizes and devices.

IV. **Creating a Complex Web Application**: Grid is better suited for creating complex web applications that require precise control over layout and responsive design.

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

  Ans.

  ➢ **Grid Template Columns and Rows:**

The grid-template-columns and grid-template-rows properties are used to define the structure of a grid container. They specify the number and size of columns and rows in the grid.

➢ **Grid Template Columns:**

- grid-template-columns: Specifies the number and size of columns in the grid.
- Values:
  - length (e.g., 100px, 20%): Specifies a fixed width for each column.
  - fr (e.g., 1fr, 2fr): Specifies a fractional unit that represents a share of the available space.
  - repeat() (e.g., repeat(3, 1fr)): Repeats a pattern of column sizes.
  - auto (e.g., auto): Automatically sizes columns based on content.

➢ **Grid Template Rows:**

- grid-template-rows: Specifies the number and size of rows in the grid.
- Values:
- length (e.g., 100px, 20%): Specifies a fixed height for each row.
- fr (e.g., 1fr, 2fr): Specifies a fractional unit that represents a share of the available space.
- repeat() (e.g., repeat(3, 1fr)): Repeats a pattern of row sizes.
- auto (e.g., auto): Automatically sizes rows based on content.

Examples:

```css
CSS
.grid-container {
  display: grid;
  grid-template-columns: 100px 200px 300px; /* Three columns with fixed widths */
  grid-template-rows: 100px 200px; /* Two rows with fixed heights */
}
```

```css
CSS
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* Three columns with equal fractional
widths */
  grid-template-rows: repeat(2, 1fr); /* Two rows with equal heights */
}
```

- **Grid Gap :** The grid-gap property is used to specify the size of the gap between grid cells.

- grid-gap: Specifies the size of the gap between grid cells.
- Values:
  - length (e.g., 10px, 20%): Specifies a fixed size for the gap.
  - normal (e.g., normal): Uses the default gap size.

Example: Using Grid Template Columns and Rows with Grid Gap

```css
CSS
#container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 1fr);
  grid-gap: 10px; /* 10px gap between grid cells */
}

.grid-item {
  background-color: #ccc;
  padding: 20px;
}
```

```html
HTML
<div Id="container">
  <div class="box">Item 1</div>
  <div class="box">Item 2</div>
  <div class="box">Item 3</div>
  <div class="box">Item 4</div>
  <div class="box">Item 5</div>
  <div class="box">Item 6</div>
</div>
```

Responsive Web Design with Media Queries

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

  Ans.

  ➢ **Media Queries in CSS:**

  Media queries are a fundamental concept in CSS that allows you to define different styles for different devices or screen sizes. They enable you to create responsive designs that adapt to various screen sizes, devices, and orientations.

  ❖ **What are Media Queries?**

  A media query is a CSS technique that allows you to specify a set of rules that apply only when certain conditions are met, such as:

  - Screen size (width, height, or both)
  - Device type (e.g., desktop, tablet, mobile)
  - Orientation (e.g., portrait, landscape)
  - Resolution (e.g., high-definition, low-definition)

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

Media queries are crucial for responsive design because they enable you to:

1) **Adapt to different screen sizes**: By defining different styles for different screen sizes, you can ensure that your website or application looks great on various devices, from small mobile screens to large desktop monitors.
2) **Improve user experience**: Media queries allow you to tailor your design to specific devices and screen sizes, providing a better user experience for your visitors.
3) **Enhance accessibility**: By adapting your design to different devices and screen sizes, you can ensure that your website or application is accessible to a wider range of users, including those with disabilities.
4) **Increase flexibility**: Media queries enable you to create flexible designs that can adapt to different devices, screen sizes, and orientations, making it easier to maintain and update your website or application.

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

CSS
```
 *{
  font-size: 20px;
}

/* Media query for screens smaller than 600px */
@media screen and (max-width: 600px) {
  .para {
    font-size: 16px;
  }
}
```

HTML
```
<body>
<p class= “para”>This is an example</p>
</body>
```

Typography and Web Fonts

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

Web-Safe Fonts vs Custom Web Fonts

Characteristics	*Web-Safe Fonts*	*Custom Web Fonts*
Compatibility	Widely supported, works on most devices	May require additional setup, may not work on all devices
Uniqueness	Common, may not stand out	Unique, can create distinctive visual identity
Design Control	Limited design control	More design control, wide range of font styles and variations
Performance	Generally faster to load	May impact page load times and performance
Examples	Arial, Calibri, Helvetica, Times New Roman, Verdana	Google Fonts, Font Squirrel, independent font designers
Use Cases	Simple and widely supported font solution, limited design requirements	Unique and distinctive visual identity, specific design requirements
Setup	Easy to set up, no additional setup required	May require additional setup and optimization

- **Reasons to Use Web-Safe Fonts:** You might use a web-safe font over a custom font for several reasons:

1) **Compatibility**: Web-safe fonts are widely supported and work on most devices, making them a reliable choice for ensuring that your website looks consistent across different platforms.

2) **Ease of Use**: Web-safe fonts are easy to set up and require no additional setup or optimization, making them a convenient choice for developers who want to focus on other aspects of their project.

3) **Performance**: Web-safe fonts are generally faster to load and don't impact page load times and performance, making them a good choice for websites that require fast loading speeds.

4) **Fallback Option**: Web-safe fonts can be used as a fallback option in case a custom font fails to load or is not supported by a particular device or browser.

5) **Simple Design Requirements**: If your website has simple design requirements and you don't need a unique and distinctive visual identity, web-safe fonts may be a suitable choice.

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

- **Font-Family Property in CSS:**

  The font-family property in CSS is used to specify the font family for a given element or group of elements. It defines the name of the font family, which can include multiple font names separated by commas.

- Here's an example of how to use the font-family property:

```css
body {
 font-family: Arial, Helvetica, sans-serif;
}
```

In this example:

- Arial is the primary font family.
- Helvetica is the secondary font family (used if Arial is not available).
- sans-serif is the generic font family (used if neither Arial nor Helvetica is available).

- **How Font-Family Works?**

When you specify multiple font families in the font-family property, the browser will attempt to use the first font family listed. If that font is not available, it will move on to the next font family listed, and so on.

Generic Font Families: There are several generic font families that can be used in the font-family property:

- serif (e.g., Times New Roman)
- sans-serif (e.g., Arial, Helvetica)
- monospace (e.g., Courier New)
- cursive (e.g., Comic Sans)
- fantasy (e.g., Impact)

These generic font families can be used as a fall back option in case the specified font families are not available.

Applying a Custom Google Font to a Webpage?

To apply a custom Google Font to a webpage, follow these steps:

Step 1: Choose a Font

1) Go to the Google Fonts website (**fonts.google.com**).
2) Browse through the font library and select a font that you like.
3) Click on the font to view its details.

Step 2: Get the Font Code

1) Click on the "Get font" button.
2) Select the font styles and weights you want to use.
3) Click on the "Embed" tab.
4) Copy the font code, which includes the `<link>` tag and the font family name.

Step 3: Add the Font Code to Your HTML

1) Add the `<link>` tag to the `<head>` section of your HTML document.
2) Replace the `href` attribute with the font code you copied.

(Using @import to Add a Google Font : You can use the `@import` rule to add a Google Font to your CSS code.)