Module 3 – Frontend – CSS and CSS3

CSS Selectors & Styling

Theory Assignment

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

🡪**CSS**[class selectors](https://www.geeksforgeeks.org/css/css-class-selector/) are one of the most versatile tools in your [front-end development toolkit](https://www.geeksforgeeks.org/blogs/best-tools-for-front-end-web-development/). They allow you to apply [styles to multiple elements](https://www.geeksforgeeks.org/css/how-to-apply-styles-to-multiple-classes-at-once/) on a [webpage](https://www.geeksforgeeks.org/computer-networks/web-pages/), ensuring your [design](https://www.geeksforgeeks.org/websites-apps/importance-of-ui-ux-design/) remains consistent and easily maintainable.

**Syntax**

.class-name {  
 property: value;  
}

The CSS element Selector

The [element](https://www.w3schools.com/cssref/sel_element.php) selector selects HTML elements based on the element name.

Example

Here, all <p> elements on the page will be center-aligned, with a red text color:

p {  
  text-align: center;  
  color: red;  
}

The CSS id Selector

The [id](https://www.w3schools.com/cssref/sel_id.php) selector uses the id attribute of an HTML element to select a specific element.

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

The CSS rule below will be applied to the HTML element with id="para1":

#para1 {  
  text-align: center;  
  color: red;  
}

The CSS class Selector

The [class](https://www.w3schools.com/cssref/sel_class.php) selector selects HTML elements with a specific class attribute.

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

Example

In this example all HTML elements with class="center" will be red and center-aligned:

.center {  
  text-align: center;  
   color: red;  
}

• Question 2:-

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

🡪 Specificity is an advanced algorithm used by HTML browsers to define the CSS declaration which will be the most appropriate to an HTML element. It basically calculates the weights of all the CSS selectors to determine the proper rule.

Conflict Resolution:

When multiple rules target the same element and define conflicting properties (e.g., different color values), the browser resolves the conflict by:

* **Comparing Specificity:** The rule with the highest specificity score takes precedence.
* **Order of Appearance (Tie-breaker):** If two or more rules have the exact same specificity, the rule that appears later in the stylesheet (or in the order of linked stylesheets) will be applied, overriding earlier rules.
* **Inheritance:** Styles can also be inherited from parent elements, but inherited styles have lower specificity than explicitly defined styles on the element itself, even if the explicitly defined style has lower specificity than the inherited one.

• Question 3:-

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

🡪

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Inline CSS** | **Internal CSS** | **External CSS** |
| **Location** | It is used within HTML tag using the style attribute. | It is used within <head> section of HTML document. | It is used in a separate .css file. |
| **Selector Scope** | Affects a single element or a group of elements. | Affects multiple elements within the same HTML element. | Affects multiple HTML documents or an entire website. |
| **Reusability** | Not reusable. Styles need to be repeated for each element. | Can be reused on multiple elements within the same HTML document. | Can be reused on multiple HTML documents or an entire website. |
| **Priority** | Highest priority. Overrides internal and external styles. | Medium priority. Overrides external styles but can be overridden by inline styles. | Lowest priority. Can be overridden by both inline and internal styles. |

## 1. Inline CSS:-

* **Advantages:**
* Highest specificity, overriding other styles.
* Useful for quick, one-off styling or testing.
* No need for separate files.
* **Disadvantages:**
* Poor maintainability and scalability for larger projects.
* Leads to code duplication if the same style is applied to multiple elements.
* Mixes content and presentation, violating separation of concerns.

**2. Internal CSS:-**

* **Advantages:**
* Styles are contained within a single HTML file, suitable for single-page application or unique page designs.
* Avoids conflicts with styles on other pages.
* **Disadvantages:**
* Not reusable across multiple pages, leading to code duplication if the same Styles are needed elsewhere.
* Increases the size of the HTML file.

**3. External CSS:-**

* **Advantages:**
* Promotes reusability and consistency across multiple pages.
* Allows for easier global style changes by modifying a single file.
* **Disadvantages:**
* Requires an additional HTTP request to fetch the CSS file.
* Pages may not render correctly until the external CSS is loaded.

## CSS Box Model

## Theory Assignment

Question 1:

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

🡪 The CSS Box Model is a fundamental concept in web development that describes how HTML elements are rendered as rectangular boxes, each with distinct layers that contribute to its overall size and position.

The components of the CSS Box Model are:

**Content:**

This is the innermost area where the actual content of the element (text, images, videos, etc.) resides. Its dimensions are typically controlled by the width and height CSS properties.

* **Effect on size:** The width and height properties directly determine the size of the content area.

**Padding:**

This is the transparent space surrounding the content area, located between the content and the border. It provides "breathing room" for the content and prevents it from directly touching the border.

* **Effect on size:** Padding adds to the overall size of the element. For example, if an element has a width of 100px and padding of 10px on all sides, the total width of the content plus padding will be 120px (100px content + 10px left padding + 10px right padding).

**Border:**

This is a line that surrounds the padding and content areas, defining the visible edge of the element. Borders can be styled with various colors, widths, and styles.

* **Effect on size:** The border adds to the overall size of the element. If an element has a width of 100px, padding of 10px, and a border of 2px, the total width of content plus padding plus border will be 124px (100px content + 10px left padding + 10px right padding + 2px left border + 2px right border).

**Margin:**

This is the transparent space outside the border, separating the element from other neighboring elements. Margins do not have a background color and are used for controlling spacing between elements on a page.

* **Effect on size:** Margin does not directly add to the element's rendered size (its width and height as perceived by the browser for layout purposes). Instead, it creates space around the element, influencing its position relative to other elements.
* Question 2:-

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

🡪 **1**.**Content-box (**Default):

* This is the default value for most HTML elements.
* When using content-box, the width and height properties you set for an element only apply to the content area.
* Any padding and border you add to the element will be added on top of this specified width and height, increasing the element's total size.

Example: If you set width: 100px; padding: 10px; border: 5px; the element's content area will be 100px wide, but its total visible width (including padding and border) will be 100px + (2 \* 10px padding) + (2 \* 5px border) = 130px.

**2.** **border-box:**

* When using border-box, the width and height properties you set for an element include the content, padding, and border.
* The browser automatically adjusts the size of the content area to accommodate the padding and border within the specified width and height, ensuring the element's total size remains as defined.

Example: If you set width: 100px; padding: 10px; border: 5px; with box-sizing: border-box;, the element's total visible width will be exactly 100px. The content area will be reduced to 100px - (2 \* 10px padding) - (2 \* 5px border) = 70px.

* The default value for box-sizing is content-box

CSS Flexbox

Theory Assignment

* Question 1: -

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

🡪 CSS Flexbox, or the Flexible Box Layout module, is a one-dimensional CSS layout model designed for arranging items in rows or columns, distributing space, and aligning content within a container. It simplifies the creation of flexible and responsive web designs, particularly for user interface components.

Usefulness for Layout Design:

* **Responsiveness:**

It allows items to "flex" their sizes, growing to fill available space or shrinking to prevent overflow, making it easier to adapt layouts to different screen sizes and orientations.

**Order Control:**

The order of flex items can be easily changed independently of their source order in the HTML, which is useful for responsive design or accessibility

Terms:

* **Flex-container:** This is the parent element that holds the flex items. To enable Flexbox layout, the display property of this element is set to flex or inline-flex. All direct children of the flex-container automatically become flex-items.

.flex-container {

display: flex; /\* or inline-flex \*/

}

* **Flex-item:** These are the direct child elements within a flex-container. They are the elements whose layout and positioning are controlled by the Flexbox properties applied to their parent (the flex-container) and to themselves.

<div class="flex-container">

<div class="flex-item">Item 1</div>

<div class="flex-item">Item 2</div>

<div class="flex-item">Item 3</div>

</div>

• Question 2:-

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

🡪 1. justify-content

* **Purpose:**

Controls how flex items are distributed and aligned along the main axis of the flex container.

* **How it works:**

It defines the alignment of items when there is extra space in the container along the main axis.

* **Common Values:**
  + flex-start (default): Items are aligned at the beginning of the main axis.
  + flex-end: Items are aligned at the end of the main axis.
  + center: Items are aligned in the center of the main axis.
  + space-between: Space is distributed evenly between items, with no space on the ends.
  + space-around: Space is distributed around items, with half the space at the ends.
  + space-evenly: Space is distributed evenly across all items, including between items and at the ends.

2. align-items

* **Purpose:**

Controls how flex items are aligned along the cross axis (perpendicular to the main axis) within their flex line.

* **How it works:**

It positions items on the cross axis, such as vertically in a row-based layout or horizontally in a column-based layout.

* **Common Values:**
  + stretch (default): Items are stretched to fill the height of the cross axis.
  + flex-start: Items are aligned at the beginning of the cross axis.
  + flex-end: Items are aligned at the end of the cross axis.
  + center: Items are aligned in the center of the cross axis.
  + baseline: Items are aligned based on their text baselines.

3. flex-direction

* **Purpose:** Sets the direction of the flex container's main axis, which in turn determines the direction of the cross axis.
* **How it works:** It establishes the primary flow of items within the container.
* **Common Values:**
  + row (default): Items are arranged in a horizontal row, with the main axis running horizontally.
  + row-reverse: Items are arranged in a horizontal row, but in reverse order.
  + column: Items are arranged in a vertical column, with the main axis running vertically.
  + column-reverse: Items are arranged in a vertical column, but in reverse order.

CSS Grid

Theory Assignment

* Question 1:-

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

🡪 CSS Grid

CSS Grid is a two-dimensional layout system that allows for the precise arrangement of content in both rows and columns simultaneously. It provides a structured approach to defining the overall page layout, enabling developers to create complex and responsive designs with explicit control over item placement and sizing within a grid.

Flexbox (Flexible Box Layout)

Flexbox is a one-dimensional layout system, primarily used for distributing and aligning items within a single row or a single column. It excels at managing spacing, alignment, and ordering of items along a single axis, making it ideal for smaller, component-level layouts.

When to use Grid over Flexbox:

* **Overall Page Layout:** For defining the main structure of a webpage, such as headers, footers, sidebars, and content areas, where precise control over row and column alignment is crucial.
* **Complex Overlapping Layouts:** When elements need to overlap or span multiple rows and columns within a defined grid.
* **Explicit Placement:** When you need to explicitly place items into specific grid cells or areas.
* Question 2:-

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

🡪 The CSS Grid Layout module provides properties to define the structure and spacing of a grid container.

**Grid-template-columns:-**

This property defines the number of columns in a grid layout and specifies the width of each column.

* **Values:** Can be a space-separated list of lengths (e.g., PX, em, and rem), percentages, Fr units (fractional units representing a portion of the available space), auto, and min-content, max content or fit-content ().
* **Example:**

Code

.container {  
 display: grid;  
 grid-template-columns: 100px 1fr 20%; */\* First column 100px, second takes remaining space, third 20% of container width \*/*  
 }

**Grid-template-rows:-**

This property defines the number of rows in a grid layout and specifies the height of each row.

* **Values:** Similar to grid-template-columns, it accepts lengths, percentages, Fr units, auto, and min-content, max content or fit-content ().
* **Example:**

Code

.container {  
 display: grid;  
 grid-template-rows: auto 50px 2fr; */\* First row auto-sized, second 50px, third takes 2x the remaining space \*/*  
 }

**Grid-gap (or gap):-**

This shorthand property defines the size of the gaps (gutters) between grid rows and columns. It combines row-gap and column-gap.

* **Values:** Can be a single value for both row and column gaps or two values where the first is for row-gap and the second for column-gap.
* **Example:**

Code

.container {  
 display: grid;  
 gap: 20px; */\* 20px gap between both rows and columns \*/*  
 */\* or \*/*  
 gap: 10px 30px; */\* 10px row gap, 30px column gap \*/*  
 }

Responsive Web Design with Media Queries

Theory Assignment:-

* Question 1:-

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

🡪 Media queries in CSS are a feature that allows the application of different styles to a webpage based on the characteristics of the device or media being used to display it. They are initiated with the @media at-rule and contain conditions based on media types (e.g., screen, print) and media features (e.g., max-width, orientation, resolution).

Here is an example of a media query:

@media screen and (max-width: 768px) {  
 */\* Styles applied when the screen width is 768px or less \*/*  
 body {  
 font-size: 16px;  
 }  
 .container {  
 flex-direction: column;  
 }  
}

Media queries are important for responsive design because they enable websites to adapt their layout and appearance dynamically to different screen sizes and device types. This ensures an optimal viewing experience for users, regardless of whether they are accessing the site on a mobile phone, tablet, or desktop computer.

* Question 2:-

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

🡪A basic media query to adjust the font size of a webpage for screens smaller than 600px utilizes the @media rule with the max-width media feature.

@media screen and (max-width: 600px) {

body {

font-size: 14px; /\* Adjust this value as desired for smaller screens \*/

}

/\* You can also target specific elements within this media query \*/

h1 {

font-size: 2em; /\* Example: adjust heading font size \*/

}

}

In this example:

* @media screen and (max-width: 600px) targets screens with a maximum width of 600 pixels. The screen media type ensures these styles apply only to screen devices.
* body { font-size: 14px ; } sets the base font size for the entire body element to 14 pixels when the screen width is 600 pixels or less.
* Specific elements like h1 can also have their font sizes adjusted within the same media query, providing more granular control over typography for smaller screens.

Typography and Web Fonts

Theory Assignment:-

* 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?

🡪 Web-safe fonts are typefaces that are universally pre-installed on the majority of operating systems and devices. Examples include Arial, Times New Roman, and Verdana. These fonts are readily available to a user's browser, eliminating the need for external downloads.

Custom web fonts, in contrast, are typefaces that are not pre-installed on user devices. Instead, they are loaded from external servers (e.g., Google Fonts, Adobe Fonts) or hosted directly on the website's server when a webpage is accessed. This allows for greater design flexibility and unique branding.

* Question 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 for an element. It accepts a comma-separated list of font names, acting as a "fall back" system. If the browser cannot find or support the first font listed, it attempts to use the next one in the list. It is recommended to always end the list with a generic font family (e.g., serif, sans-serif, monospace) to ensure the browser can always find a suitable fallback font if none of the specific fonts are available. Font names with multiple words must be enclosed in quotation marks, such as "Times New Roman".