# Module 16)

# CSS in Full Stack Course

1. CSS Selectors & Styling

**Theory Assignment**

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

A CSS selector is the first part of the CSS rule. It is a pattern used to select the HTML elements that developer wants to style. The selector tells the browser which elements the CSS rules should apply to.

Element Selector: It targets all elements of a specific type.

Class Selector: It targets element with a specific class. In this class names start with a dot.

ID Selector: It targets a single unique element with a specific ID. ID selector start with #.

Examples of element, class, and ID selectors

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        /\* element selector \*/

        h4{

            color: brown;

        }

        /\* Class selector \*/

        .box{

            border: 4px solid black;

            height:50px;

            width: 180px;

            color: rgb(56, 105, 6);

            padding:8px;

        }

        /\* ID selector \*/

        #header{

            color:blue;

        }

    </style>

</head>

<body>

    <h4 class="p"> This is an element selector.  </h4>

    <p class="box"> This is a class selector start with a dot (.) </p>

    <div id="header"> This is a ID selector start with hash (#).</div>

</body>

</html>

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

CSS specificity is a set of rules the browser uses to determine which CSS rule to apply when multiple rules target the same element.

* When multiple styles apply to the same element, the rule with the highest specificity wins.
* CSS specificity decides which style rule wins when multiple rules target the same element. The more specific the selector, the higher its priority.

p {

color: red; /\* element selector \*/

}

.box {

color: green; /\* class selector \*/

}

#special {

color: blue; /\* ID selector \*/

}

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

**External CSS:** With an external style sheet, we can change the look of an entire website by changing just one file.

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

<head>

<link rel=”stylesheet” href=”file.css”>

</head>

* An external style sheet can be written in any text editior and must be saved with a .css extension.
* The external .css file should not contain any HTML tags.

Advantage:

* Best for large websites.
* Reusable across multiple HTML pages.
* Cleaner HTML and easier maintenance.

Disadvantage:

* Requires an extra HTTP request.
* Styles may not apply if the CSS file is missing or fails to load.

**Internal CSS:** An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Image collage</title>

    <style>

      \*{

            box-sizing:border-box

      }

        .diamond{

            width: 100px;

            height:100px;

            border-radius:4px;

            position: absolute;

            transform: rotateZ(45deg);

            overflow: hidden;

        }

        .diamond img{

            width: 100px;

            height:100px;

            border-radius:4px;

            object-fit: cover;

        }

        #img1{

            margin-top: 100px;

            margin-left: 100px;

        }

        #img2{

            margin-top: 100px;

            margin-left: 240px;

        }

    </style>

</head>

Advantage:

* Keeps styles in one place in the same file.
* Good for small websites or single-page projects.

Disadvantage:

* Not reusable across multiple pages.
* Increases page size if repeated on many pages.

**Inline CSS:** An inline style may be used to apply a unique style for a single element.

To use inline style, add the style attribute to the relavent element. The style attribute can contain any CSS property.

<h2 style="text-align: center;">Filter</h2>

Advantage:

* Quick and easy for small changes.
* Useful for testing or overriding styles.

Disadvantage:

* Not reusable.
* HTML file becomes messy because CSS is mixed directly inside HTML tags.
* Low maintainability.

**Lab Assignment**

4• Task:

Style the contact form (created in the HTML Forms lab) using external CSS.

The following should be implemented:

* Change the background color of the form.
* Add padding and margins to form fields.
* Style the submit button with a hover effect.
* Use class selectors for styling common elements and ID selectors for unique elements.

CSS STYLE:

         \*{

            box-sizing: border-box;

            margin:0px;

        }

        .row

        {

            display:flex;

            flex-wrap:wrap;

        }

         [ class\*= "col-"]

            {

                width:100%;

            }

        /\* class work on Desktop >768 \*/

        @media only screen and (min-width: 768px)

        {

            .col-1 {width: 8.33%;}

            .col-2 {width: 16.66%;}

            .col-3 {width: 25%;}

            .col-4 {width: 33.33%;}

            .col-5 {width: 41.66%;}

            .col-6 {width: 50%;}

            .col-7 {width: 58.33%;}

            .col-8 {width: 66.66%;}

            .col-9 {width: 75%;}

            .col-10 {width: 83.33%;}

            .col-11 {width: 91.66%;}

            .col-12 {width: 100%;}

        }

        body{

            background-image: url(img/form\ bg.jpg);

            background-size: cover;

            background-repeat: no-repeat;

            width:100%;

            height: 500px;

            display: flex;

            justify-content: center;

            align-items: center;

            color: white;

        }

       .form{

        background-color: black;

        opacity:70%;

        width: 50%;

        height: 400px;

        padding: 8px;

        justify-content: center;

        align-items: center;

       }

       .registration{

         font-family: sans-serif;

         padding: 5px;

         margin-top: 10px;

       }

        input[type="text"],

        input[type="email"],

        input[type="password"],

        input[type="tel"]{

        background: none;

        border: 2px solid gray;

        border-radius: 4px;

        width: 90%;

        margin: 35px 5px 5px 5px;

        padding: 8px;

       }

       input:hover{

        border-color: white;

        color: white;

       }

       .checkbox{

        text-align: left;

        margin: 25px 5px;

        position: relative;

        align-items: left;

       }

       .button{

        padding: 5px;

        display: block;

        margin:15px auto;

       }

       .button:hover{

        color: rgb(232, 223, 223);

        background-color:rgb(0, 0, 0);

       }

HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Registration Form</title>

     <!-- Bootstrap -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">

    <link href="Task\_1.4 Contact Form.css" rel="stylesheet">

</head>

<body>

    <div class="form">

        <form class="registration" action="#" method="POST">

                <h2 style="text-align: center;"> Registration</h2>

                <div class="row">

                    <div class="col-6">

                        <input type="text" placeholder="Full Name">

                        <input type="Email" placeholder="Email">

                        <input type="password" placeholder="Password">

                    </div>

                    <div class="col-6">

                        <input type="text" placeholder="Username">

                        <input type="tel" placeholder="Phone Number">

                        <input type="password" placeholder="Confirm Password">

                    </div>

                </div>

                <div class="checkbox">

                    <input type="checkbox"> Send me E-mails about the News.

                </div>

                <button class="button" > Register</button>

        </form>

    </div>

</body>

</html>

Output



1. 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?

The CSS Box Model is the fundamental concept that describes how elements are rendered on a web page. Every HTML Element is treated as a box consisting of:

* Content
* Padding
* Border
* Margin

**Content:**

* In this text, images or other inner elements appear.
* By using width and height its size can be control.

**Padding:**

* Padding is defines as the space between content and border.
* Increases the inner spacing, but also adds to the total size of the box

**Box-sizing: border-box;**

**Border:**

* A line around the padding and content.
* A border can be styled by using **border-style** property as dotted, dashed and many more.
* Here are some border property are as:
  + Border-radius
  + Border-color
  + Border-width
  + Border-style
  + Border-collapse
  + Border-left
  + Boder-right
  + Border-top
  + Border-bottom, etc.

**Margin:**

* Margin is used to give space outside of the border, also it is used to separate elements.
* It doesn’t affect the size of the element, but affects spacing between elements.

How does each affect the size of an element?

By default (box-sizing:content-box)

Total element size is calculated as:

**Total width = width + padding-left + padding-right + border-left + border-right**

**Total height = height + padding-top + padding-bottom + border-top + border-bottom**

To control this use:

**Box-sizing:border-box;**

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

In CSS, content-box is the default box-sizing property, while border-box is another option. The key difference lies in how the specified width and height properties are interpreted: with content box, the width and height apply only to the content area and padding and borders are added to the outside, increasing the overall size of the element. With border-box, the specified width and height include the content, padding and border, the padding and border are effectively contained within the specified dimensions, preventing them from increasing the element’s overall size.

**Content-box:** (default)

* The width and height properties define the size of the content area only.
* Padding and borders are added to the outside of the content area, increasing the overall size of the element.

**Border-box:**

* The width and height properties define the total size of the element, including content, padding and border.
* Padding and borders are contained within the specified width and height.

**Lab Assignment**

2.3 Task: Create a profile card layout using the box model. The profile card should include:

* A profile picture.
* The user’s name and bio.
* A button to "Follow" the user. Additional Requirements:
* Add padding and borders to the elements.
* Ensure the layout is clean and centered on the page using CSS margins.
* Use the box-sizing property to demonstrate both content-box and border-box on different elements.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        .wrap{

           display: flex;

           justify-content: center;

           align-items: center;

        }

       .border{

           background-color: rgb(199, 223, 245);

           box-sizing:border-box;

           border-style: double;

           width: 200px;

           height: 300px;

           margin: 5px;

           padding: 5px;

           display: flex;

           justify-content: center;

       }

       .image{

           width:65px;

           height:70px;

           border: 2px solid black;

       }

       .data{

            box-sizing:content-box;

            color: black;

            display: inline;

            text-align: center;

       }

       .p{

           text-align:left;

           font-size: 10px;

           display: block;

           line-height: 15px;

       }

    </style>

</head>

<body>

    <div class="wrap">

        <div class="border">

            <div class="data">

                <h2> Identity Card </h2>

                <img src="img/id.jpg" class="image">

                <h3> Assistant Professor <br> <b style="font-size: x-small;">(Computer Science) </b> </h3>

                <p class="p">

                    <b> Name:</b> Antima Namdev <br>

                    <b> Father Name:</b> Mr. Umashankar Namdev <br>

                    <b> DOB:</b> 15/10/1996 <br>

                </p>

            </div>

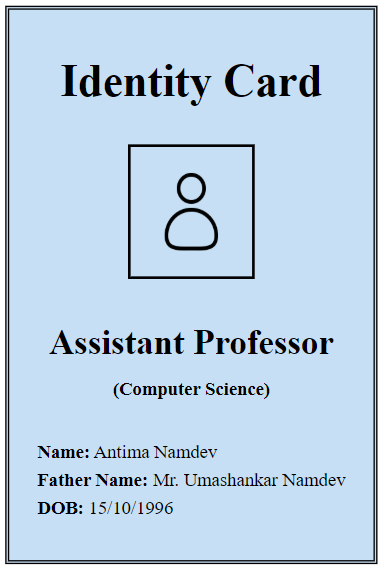
        </div>

    </div>

</body>

</html>

Output:



1. CSS Flexbox

**Theory Assignment**

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

CSS Flexbox is a layout model in CSS designed to arrange and align elements efficiently when the size of the content is unknown or dynamic. It helps to create responsive layouts that work well on all screen sizes as mobile, tablet or desktop.

**It is useful because:**

* Flexbox makes it easy to align items horizontally and vertically.
* It helps distribute space between items dynamically.
* It works well for both rows and columns.
* It replace older layout techniques like float, inline-block and complicated positioning.

Flexbox is especially helpful when creating layout like navigation bars, cards, profile sections or centered content.

**flex-container:**

A flex container is the parent element where flexbox layout starts. You make an element a flex container by applying the CSS property.

**Display:flex;**

**flex-item:** Flex items are the direct childern of the flex container. These are the elements that are arranged and aligned by flexbox.

**Some flexbox properties are:**

* **Flex-grow**
* **Flex-shrink**
* **Flex-basis**
* **Align-self**

|  |
| --- |
|  |
|  |
| |  |  |  | | --- | --- | --- | | **Property** | **Applies to** | **Description** | | display: flex | Flex container | Starts the Flexbox layout | | flex-direction | Flex container | Sets the direction (row or column) of items | | justify-content | Flex container | Aligns items on the main (horizontal) axis | | align-items | Flex container | Aligns items on the cross (vertical) axis | | flex | Flex item | Controls growth, shrink, and base size | |  |  |

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

In CSS flexbox, these three properties are used on the flex container to control the layout and alignment of flex items.

* **Justify-content:** This property is used to align flex items along the main axis (horizontally by default).

|  |  |
| --- | --- |
| **flex-start** | Items align to the left (start of the container). |
| **flex-end** | Items align to the right (end of the container). |
| **center** | Items are centered horizontally. |
| **space-between** | Items have equal space between them. |
| **space-around** | Items have equal space around them. |
| **space-evenly** | Equal space is distributed between and around items. |

* **Flex-direction:** This property is used to define the items are placed in a row or a column.

|  |  |
| --- | --- |
| **Row** | Items go left to right. |
| **Row-reverse** | Item go right to left. |
| **Column** | Items gotop to bottom. |
| **Column-reverse** | Item go top to bottom. |

* **align-items:** This property is used to align flex items along the cross axis (vertically by default).

|  |  |
| --- | --- |
| **flex-start** | Items align to top. |
| **flex-end** | Items align to the bottom. |
| **center** | Items are centered vertically. |
| **Stretch** | Items stretch to fill the container height. |
| **baseline** | Items align according to their text baseline. |

**Lab Assignment**

3.3 Task: Create a simple webpage layout using Flexbox. The layout should include:

* A header.
* A sidebar on the left.
* A main content area in the center.
* A footer. Additional Requirements:
* Use Flexbox to position and align the elements.
* Apply different justify-content and align-items properties to observe their effects.
* Ensure the layout is responsive, adjusting for smaller screens.

 <!DOCTYPE html>

 <html lang="en">

 <head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

        <style>

            \*{

                box-sizing: border-box;

                margin:0px;

            }

            .row

            {

                display:flex;

                flex-wrap:wrap;

            }

            [ class\*= "col-"]

                {

                    width:100%;

                }

            /\* class work on Desktop >768 \*/

            @media only screen and (min-width: 768px)

            {

                .col-1 {width: 8.33%;}

                .col-2 {width: 16.66%;}

                .col-3 {width: 25%;}

                .col-4 {width: 33.33%;}

                .col-5 {width: 41.66%;}

                .col-6 {width: 50%;}

                .col-7 {width: 58.33%;}

                .col-8 {width: 66.66%;}

                .col-9 {width: 75%;}

                .col-10 {width: 83.33%;}

                .col-11 {width: 91.66%;}

                .col-12 {width: 100%;}

            }

                    /\* <!-- •   A header.

        •   A sidebar on the left.

        •   A main content area in the center.

        •   A footer. Additional Requirements:

        •   Use Flexbox to position and align the elements.

        •   Apply different justify-content and align-items properties to observe their effects.

        •   Ensure the layout is responsive, adjusting for smaller screens.

        --> \*/

        body{

            text-align: center;

            font-family: sans-serif;

            font-size: large;

        }

        nav{

            background-color: mediumturquoise;

            width: 100%;

            height: 50px;

            text-align: center;

            padding: 8px;

        }

        aside{

            background-color: rgb(232, 209, 115);

            width: 100%;

            height: 400px;

            padding: 50px;

            line-height: 25px;

        }

        main{

            background-color: cornflowerblue;

            width: 100%;

            height: 400px;

            padding: 50px;

            line-height: 25px;

        }

        footer{

            background-color: black;

            color: white;

            padding: 5px;

        }

        </style>

</head>

 <body>

    <div class="row">

        <nav>

            <h1> Annu's Fashion</h1>

        </nav>

    </div>

    <div class="row">

        <div class="col-3">

            <aside>

                <h2> Sidebar </h2>

                <br>

                <ol style="text-align: left;">

                    <li> Home </li>

                    <li> About us </li>

                    <li> Service </li>

                    <li> Contact us </li>

                </ol>

            </aside>

        </div>

        <div class="col-9">

            <main>

                <h2> About us </h2>

                <br>

                <span>

                    Ecommerce can feel a little impersonal at times. Shoppers experience your brand through a screen instead of in person, where they can touch and handle the products before making a purchase. As a result of this detachment, shoppers might forget your brand or even mix you up with a competitor. To avoid this, it’s important […]

                </span>

                <br>

                <a href="#"> Click here for more information</a>

            </main>

        </div>

    </div>

    <footer>

       © 2025 My Website

    </footer>

 </body>

 </html>

Output



1. 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 powerful two-dimensional layout system in CSS that allows developers to create complex web page layouts using row and columns. With grid, you can precisely control the placement and sizing of elements in both row and columns or horizontal and vertical directions.

**Key Features of CSS Grid:**

* Works in rows and columns
* Allows you to define grid tracks and place items anywhere in the grid.
* Provide full control over layout structure, including spacing, alignment and overlap.

|  |  |  |
| --- | --- | --- |
| **Flexbox vs Grid** | | |
| **Feature** | **Flexbox** | **CSS Grid** |
| Layout Direction | One-dimensional  (row or column) | Two-dimensional  (row and column) |
| Best for | Aligning items in a single line | Creating full-page or section layouts |
| Item Placement | Items placed in order  (flow-based) | Items can be placed anywhere in the grid |
| Alignment | Powerful along a single axis | Powerful along both axis |
| Content or Layout-First | Content-first layout | Layout-first design |

**use Grid over Flexbox:**

* When you need to create complex page layouts with both rows and columns.
* When your layout is not linear
* When we want to align items precisely in both directions or create overlapping content.
* When the layout depends more on the grid structure than the flow of content.

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

This three properties are part of CSS Grid and are used to define the layout structure of a grid container.

### **1. grid-template-columns**

This sets **how many columns** you want and how wide each column should be.

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

### **2. grid-template-rows**

This sets **how many rows** you want and how tall each row should be.

**grid-template-rows: 100px 150px;**

### **3. grid-gap**

This sets the **space between rows and columns** — like a gap between boxes.

**grid-gap: 20px;**

You can also set row and column gaps separately: row-gap: 10px;

**column-gap: 30px;**

**Lab Assignment**

4.3 Task: Create a 3x3 grid of product cards using CSS Grid.

Each card should contain:

* A product image.
* A product title.
* A price.

Additional Requirements:

* Use grid-template-columns to create the grid layout.
* Use grid-gap to add spacing between the grid items.
* Apply hover effects to each card for better interactivity.

1. Responsive Web Design with Media Queries

Theory Assignment

1. What are media queries in CSS, and why are they important for responsive design?
2. Write a basic media query that adjusts the font size of a webpage for screens smaller than 600px.

Lab Assignment

• Task: Build a responsive webpage that includes:

* A navigation bar.
* A content section with two columns.
* A footer. Additional Requirements:
* Use media queries to make the webpage responsive for mobile devices.
* On smaller screens (below 768px), stack the columns vertically.
* Adjust the font sizes and padding to improve readability on mobile.

1. 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?
2. What is the font-family property in CSS? How do you apply a custom Google Font to a webpage?

Lab Assignment

• Task: Create a blog post layout with the following:

* A title, subtitle, and body content.
* Use at least two different fonts (one for headings, one for body content).
* Style the text to be responsive and easy to read.

Additional Requirements:

* Use a custom font from Google Fonts.
* Adjust line-height, font-size, and spacing for improved readability.