**LAB 1**

**Title:** Write HTML code using basic HTML tags.

**Theory:**

**HTML:** HTML stands for Hyper Text Markup Language. It is the back bone of the web page and it's the standard language used to create and design web pages. HTML tell the browser how to display text, images and other forms of content.HTML consists of series of elements and tags that tell the web browser how to display the content.

For example:

<!DOCTYPE html>

<html>

<head>

<title>Sample Page</title>

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is a paragraph of text.</p>

</body>

</html>

Each tag, like <h1> for headings or <p> for paragraphs, has a specific role. HTML is essential for building the skeleton of a webpage, and it works in conjunction with CSS (Cascading Style Sheets) for styling and JavaScript for interactivity.

**HTML Tags :**

HTML tags are the building blocks of HTML, used to create elements on a webpage. They are enclosed in angle brackets (< >) and often come in pairs: an opening tag and a closing tag. Here are some common HTML tags and their purposes:

1. <html>…</html> - Defines the root of an HTML document.
2. <head>…</head> - Contains metadata and links to scripts and stylesheets.
3. <title>…</title> - Sets the webpage's title, shown in the browser’s title bar.
4. <body>…</body> - Contains the main content of the webpage.
5. <h1> to <h6> - Define headings, with <h1> being the highest level and <h6> the lowest.
6. <b>…</b> - Set in boldface.
7. <i> …</i>- Set in italics.
8. <center>…</center> - Center on the page horizontally.
9. <br> - Forces a line break here.
10. <p> - Defines a paragraph.
11. <hr> - Inserts a horizontal rule.
12. <a href=" ">..</a> - Defines a hyperlink.
13. <img scr=".."> - Embeds an image.
14. <ul>…</ul> and <ol>…</ol> - Define unordered (bullet points) and ordered (numbered) lists, respectively.
15. <li>…</li> - Represents a list item.

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Lab 1</title>

</head>

<body>

    <h1>Web Technology.</h1> <!--Heading-->

    <hr><!--Horizontal rule-->

    <p>HTML stand for <b>Hypertext Markup Language</b>. <!--Bold face-->

        <i>It is the backbone of webpage and    <!--Italics-->

         <br>it is the standatd language used to create and design webpage.</i></p>

    <center><b>HTML Tags</b></center> <!--Center on the page horizontally-->

    <ul>    <!--Unordered List-->

        <li>Head</li> <!--Lists-->

        <li>Body</li>

        <li>Title</li>

      </ul>

      <h2>HTML Tags</h2>

      <ol>  <!--Ordered List-->

        <li>Center</li>

        <li>a href</li>

        <li>img scr</li>

      </ol>

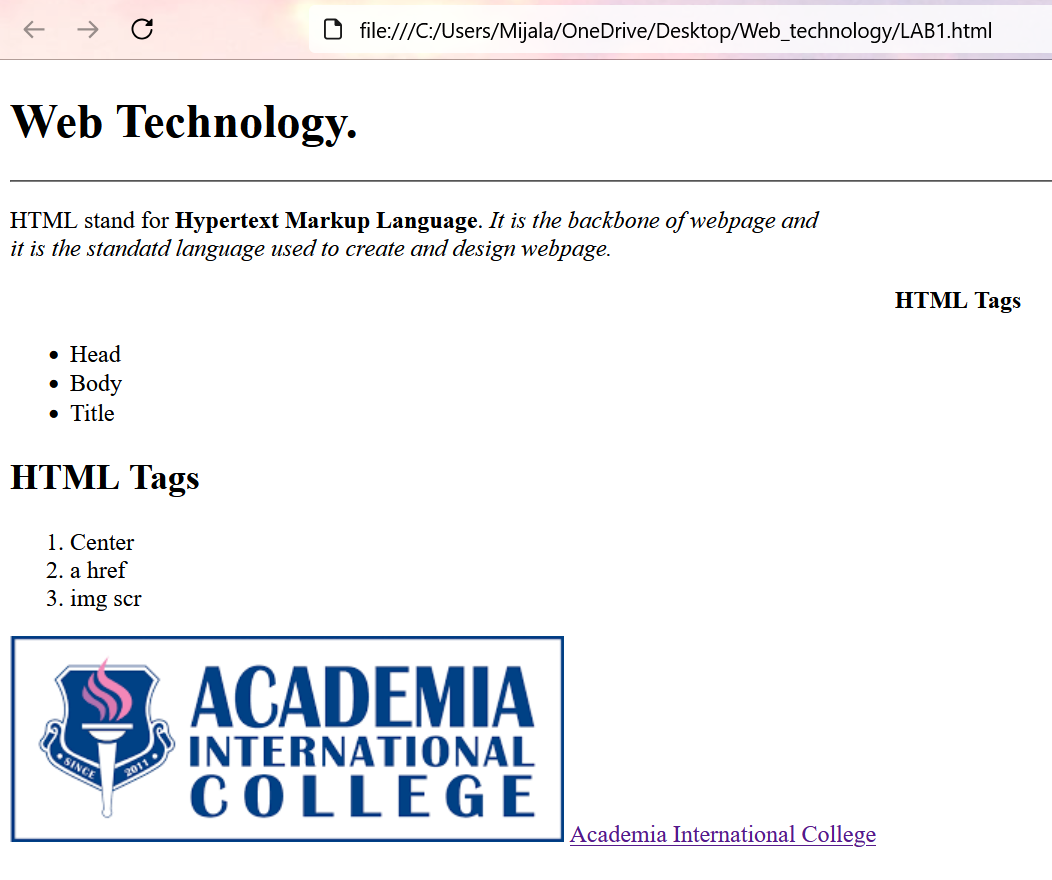
      <img src="logo.png" alt="blue college logo">   <!--Display an image-->

      <a href="https://www.academiacollege.edu.np/">Academia International College</a>  <!--Link to a website-->

</body>

</html>

**Output:**

****

**LAB 2**

**Title:** Write an HTML script to create a table with three rows and three columns. The second row and third column should contain “About us” with a link to [www.academiacollege.edu.np](http://www.academiacollege.edu.np/). The height of the second row should be set to 200px. Please comment on each line also.

**Theory:**

HTML tables allow us to arrange data into rows and columns. A table in HTML consists of table cells inside rows and columns. Each table cell is defined by a <td> and a </td> tag. Td stands for table data. Everything between <td> and </td> is the content of a table cell. A table cell can contain all sorts of HTML elements i.e. text, images, lists, links, etc. Each table starts with a <tr> and ends with </tr> tags where tr stands for table row.

th stands for table header.

HTML tables can have borders of different styles and shapes. CSS border property is used on tables, th, and the td element to add a border in the table. To avoid the double border in the table set CSS border-collapse property to collapse. This will cause the borders to collapse into a single border. With the border-radius property, the borders get rounded corners. With the border-style property, you can set the appearance of the border. With the border-color property, you can set the color of the border.

HTML tables can have different sizes for each column, row, or entire table. We use the style attribute with the width or height properties to specify the size of a table, row, or column.

To set the width of a table, add the style attribute to the <table> element.

To set the size of a specific column, add the style attribute on a <th> or <td> element.

**HTML Code:**

<!DOCTYPE html><!--Document type and version of html-->

<html lang="en"><!--language used english-->

<head>

    <meta charset="UTF-8">

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

    <title>Table example</title> <!--Title of broswer tab-->

    <style>

        table{         /\*Table element inside style tag\*/

            width:50%;

            border-collapse: collapse;

            background-color: beige;

            font-family: Verdana, Geneva, Tahoma, sans-serif;

            font-style: italic;

        }

        td,th{

            border:1px solid;

            padding:8px;

            text-align:center;

        }

        .row2{  /\*creating class row2\*/

            height:200px;

        }

        .row3{ /\*creating class row3\*/

            height:100px

        }

    </style>

</head>

<body>

    <table>

        <tr> <!--table row🡪

            <th>S.N</th> <!--table header🡪

            <th>Institution Name</th>

            <th>Website</th>

        </tr>

        <tr class="row2"> <!--calling class row2 🡪

            <td>1</td><!--table data🡪

            <td>Academia International College</td>

            <td><a href="http://www.academiacollege.edu.np" target="blank">About Us</a></td><!--Link to cademia website🡪

         </tr>

        <tr class="row3" > <!--calling class row3 🡪

            <td></td>

            <td></td>

            <td>

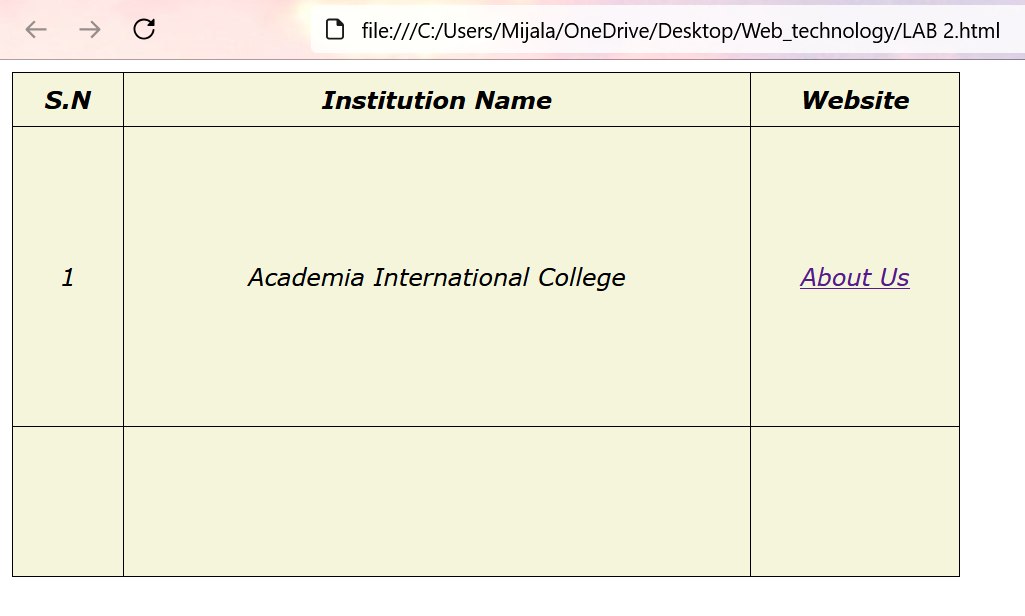
        </tr>

    </table>

</body>

</html>

**Output:**

****

**LAB 3**

**Title:** 1.Create an HTML page with its body containing a div. And also make another div for two different paragraphs. The div should contain an image within it. Create a link on the image to redirect to the URL [http://www.academiacollege.edu.np](http://www.academiacollege.edu.np/). Set the title of the page to “Academia College”. Add a Meta tag on the page  
2. Create an HTML page containing ordered and unordered lists. Set the value of the ordered list type to “A”. The list should start at “D”.

**Theory:**

The <div> element is a container for other HTML elements. The <div> element is by default a block element, meaning it takes all available width, and comes with line breaks before and after. The <div> element has no required attributes, but style, class, and id are common.

The <div> element is often used to group sections of a web page together. We can have many <div> containers on the same page.

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Academia College</title>

    <style>

        .image{

            height:150px;

            width:500px;

            background-color: goldenrod;

            padding: 10px;

            border:10px;

            margin:10px;

        }

        .paragraph{

            background-color: darksalmon;

            padding:10px;

            border:10px;

            text-align:left;

            margin:10px;

        }

    </style>

</head>

<body>

   <h2>Academia International College</h2>

    <div class="image">

    <img src="logo.png"alt="acadeima logo" >

    <a href="http://www.academiacollege.edu.np" target="\_blank">Academia college</a>

    </div>

    <div class="paragraph">

        <p>Welcome to Academia College, a center of excellence for higher education in Nepal</p>

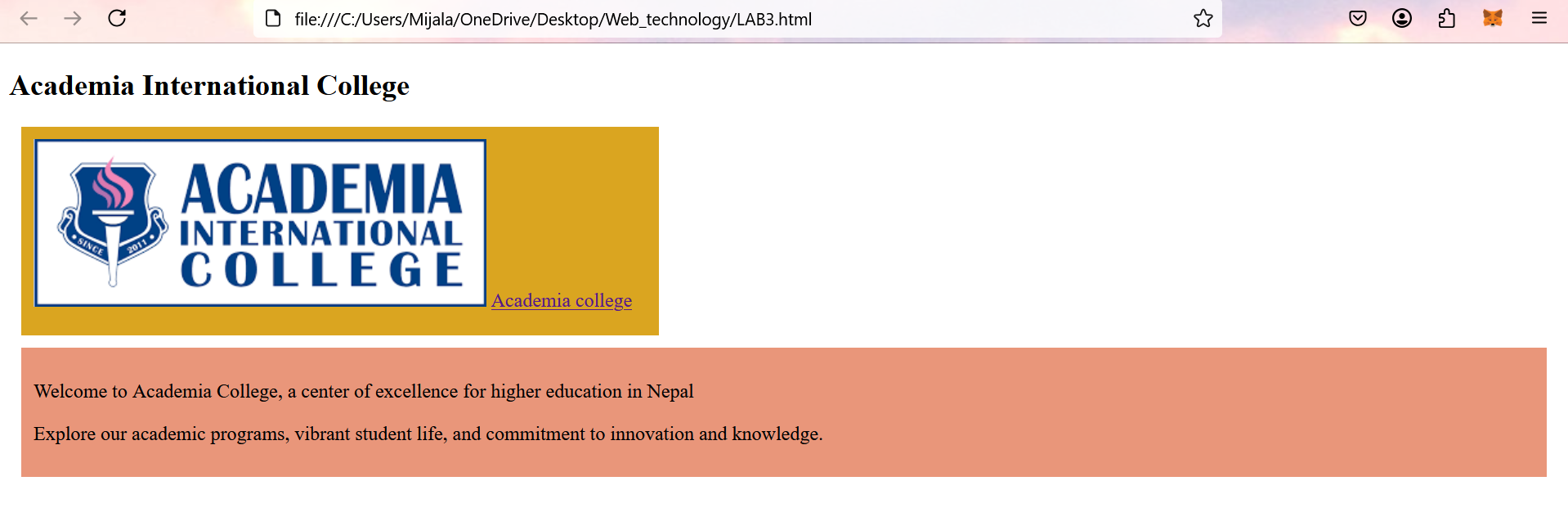
        <p>Explore our academic programs, vibrant student life, and commitment to innovation and knowledge.</p>

    </div>

</body>

</html>

**Output:**

****

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Page containing ordered and unordered list</title>

</head>

<body>

    <h2>Ordered List</h2>

    <ol type="A" start="4">

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

    </ol>

    <h2>Unordered List</h2>

    <ul>

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

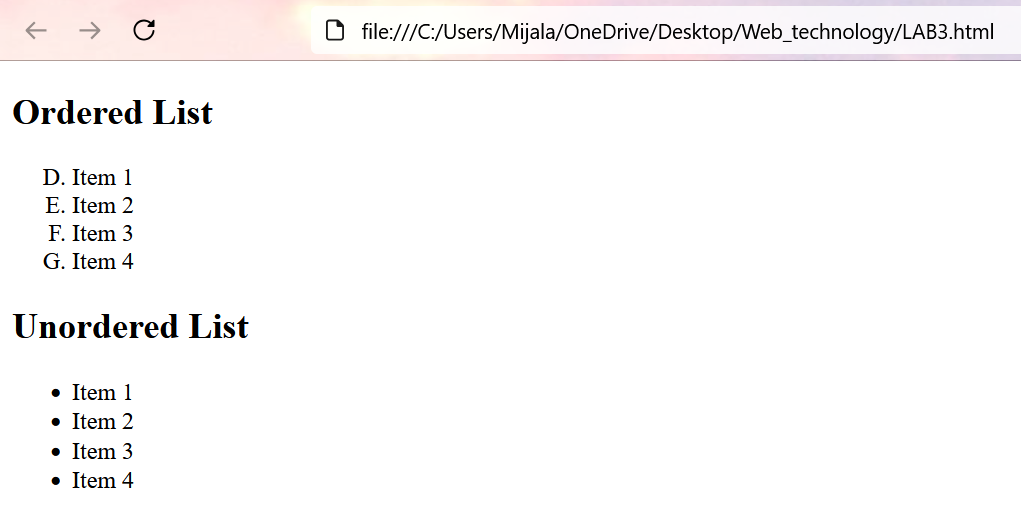
        <li>Item 4</li>

    </ul>

</body>

</html>

**Output:**

****

**LAB 4**

**Title:** Write a code using  HTML Events: Window Events, Form Element Events, Keyboard Events, Mouse Events.

**Theory:**

**Window Event:** Window events are triggered by actions that affect the entire browser window or document, such as loading, resizing, or closing a page (applies to the <body> tag).

**Examples:**

* onload: Fires when the page finishes loading.
* onresize: Triggers when the window is resized.
* onscroll: Fires when the user scrolls the page.
* <body onload="alert('Page loaded!')">

**Form Element Event:** Form events are triggered by user interactions with form elements like input boxes, dropdowns, checkboxes, and forms.

**Examples:**

* onchange: Fires when the value of an input or select changes.
* onsubmit: Fires when a form is submitted.
* onfocus: Fires when a field gets focus.
* onblur: Fires when a field loses focus.
* <form onsubmit="alert('Form submitted!')"><input type="text" onchange="alert('Input changed!')">

</form>

**Keyboard Event:** Keyboard events are triggered when the user interacts with the keyboard, like pressing or releasing a key.

**Examples:**

* onkeydown: Fires when a key is pressed down.
* onkeypress: Fires when a key is pressed (deprecated).
* onkeyup: Fires when a key is released.
* <input type="text" onkeydown="alert('Key pressed!')">

**Mouse Event:** Mouse events are triggered by interactions with the mouse, such as clicks, movement, or hovering over elements.

**Examples:**

* onclick: Fires when an element is clicked.
* onmouseover: Fires when the mouse pointer hovers over an element.
* onmouseout: Fires when the mouse pointer leaves an element.
* ondblclick: Fires on double-click.
* <div onclick="alert('Div clicked!')">Click Me</di**v>**

**HTML Code:**

<!DOCTYPE html>

<html>

<head>

  <title>HTML Events Demo</title>

  <style>

    /\*using # to target an element by its id.\*/

#mouseArea {

  width: 200px;

  height: 100px;

  border: 2px dashed dimgray;

  text-align: center;

  line-height: 100px;

  margin-top: 10px;

}

  </style>

</head>

<body onload="windowLoaded()" onresize="windowResized()">

  <h2>HTML Events Demo</h2>

  <!-- Window Events -->

  <p><strong>Window Events:</strong> Resize or reload the window to see effects in console.</p>

  <!-- Form Element Events -->

  <form onsubmit="return handleSubmit(event)">

    <label for="name">Name:</label>

    <input type="text" id="name" onfocus="inputFocused()" onchange="inputChanged()">

    <button type="submit">Submit</button>

  </form>

  <!-- Keyboard Events -->

  <p><strong>Keyboard Events:</strong> Type something in the input below.</p>

  <input type="text" onkeydown="keyPressed(event)" placeholder="Type here">

  <!-- Mouse Events -->

  <p><strong>Mouse Events:</strong> Hover or click inside the box below.</p>

  <div id="mouseArea"

       onmouseover="mouseOver()"

       onmouseout="mouseOut()"

       onclick="mouseClick()">

    Hover or Click me

  </div>

  <script>

    // Window Events

    function windowLoaded() {

      console.log("Window has loaded!");

    }

    function windowResized() {

      console.log("Window has been resized!");

    }

    // Form Element Events

    function inputFocused() {

      console.log("Input field is focused.");

    }

    function inputChanged() {

      console.log("Input value has changed.");

    }

    function handleSubmit(event) {

      event.preventDefault(); // Prevent actual form submission

      alert("Form submitted!");

      return false;

    }

    // Keyboard Events

    function keyPressed(event) {

      console.log("Key pressed:", event.key);

    }

    // Mouse Events

    function mouseOver() {

       document.getElementById("mouseArea").style.backgroundColor = "aliceblue";

    console.log("Mouse over the box.");

    }

    function mouseOut() {

      document.getElementById("mouseArea").style.backgroundColor = "white";

      console.log("Mouse left the box.");

    }

    function mouseClick() {

      alert("Box clicked!");

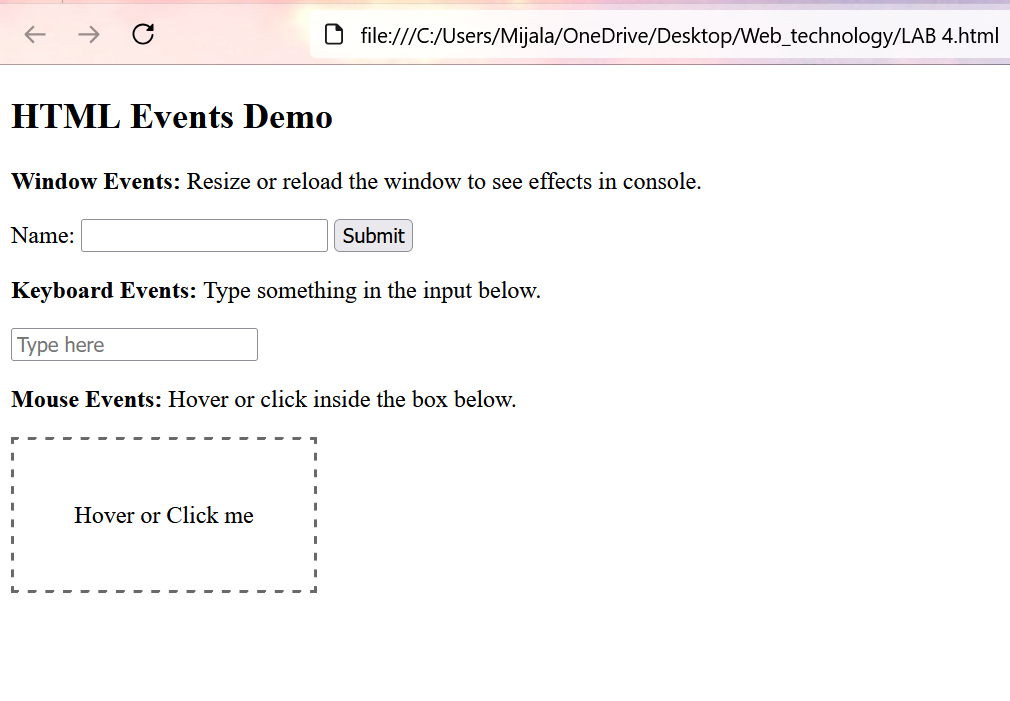
    }

  </script>

</body>

</html>

**Output:**

****

**LAB 5**

**Title:** 1. Write an HTML document demonstrating the three CSS types: inline, internal, and external. The document should include:

A heading styled using inline CSS.

A paragraph styled using a class selector in internal CSS.

A <div> styled using an ID selector in internal CSS.

A reference to an external CSS file named styles.css.

2. Write an HTML and CSS code demonstrating the CSS Box Model with margin, padding, border, and content.  
  
3. Write an HTML and CSS code demonstrating the Normal Flow Box Layout, including basic box layout, display property, padding, and margin.

**Theory:**

CSS is the language we use to style an HTML document. CSS describes how HTML elements should be displayed.

**1. Inline CSS:**

* CSS is written directly inside an HTML element’s style attribute.
* Affects only that specific element.
* Used for quick styling or testing.

Example: <p style="color: red; font-size: 16px;">This is inline CSS</p>

**2. Internal CSS:**

* CSS is written within a <style> tag inside the <head> section of the HTML document.
* Styles apply to the whole page (all elements that match the selectors).

Example:

<html>

<head>

<style>

p {

color: blue; font-size: 18px;

}

</style>

</head>

<body>

<p>This is internal CSS</p> </body> </html>

**3. External CSS:**

* CSS is written in a separate .css file, which is linked to the HTML using the <link> tag.
* Best for larger websites or reusing styles across multiple pages.

Example:

HTML file (index.html):

<head>

<link rel="stylesheet" href="styles.css">

</head>

External CSS file (styles.css):

p {

color: green;

font-size: 20px;

}

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>CSS using inline,internal and external </title>

    <!--link to external css-->

    <link rel="stylesheet" href="style.css">

    <style>

/\*Internal CSS\*/

        .styleparagraph{

            color:beige;

            font-size:30px;

        }

        #styled-div {

            background-color: lightblue;

            padding: 20px;

            border: 2px solid darkblue;

        }

</style>

</head>

<body>

    <!-- Inline CSS -->

    <h1 style="color: blue; text-align: center;">Inline CSS Styled Heading</h1>

<!-- Internal CSS using class selector -->

    <p class="styled-paragraph">This paragraph is styled using internal CSS with a class selector.</p>

    <!-- Internal CSS using ID selector -->

    <div id="styled-div">This div is styled using internal CSS with an ID selector.</div>

 <footer>

        This footer is styled using external CSS.

    </footer>

</body>

</html>

**CSS Code:**

footer {

    background-color: lightskyblue;

    color: white;

    text-align: center;

    padding: 10px;

    margin-top: 40px;

    font-style: italic;

}

**Output:**

****

**Theory:** In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins.

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>CSS Box Model Demo</title>

    <link rel="stylesheet" href="style1.css">

</head>

<body>

    <h1>CSS Box Model Example</h1>

    <div class="box">

        This is the content inside the box.

    </div>

</body>

</html>

**CSS Code:**

body {

    font-family: Arial, sans-serif;

    background-color: lightgray;

    padding: 20px;

}

h1 {

    color:black;

}

.box {

    width: 300px;

    background-color: white;

    color:black;

    /\* Padding: space between content and border \*/

    padding: 20px;

    /\* Border: visible edge around the element \*/

    border: 3px solid green;

    /\* Margin: space outside the border \*/

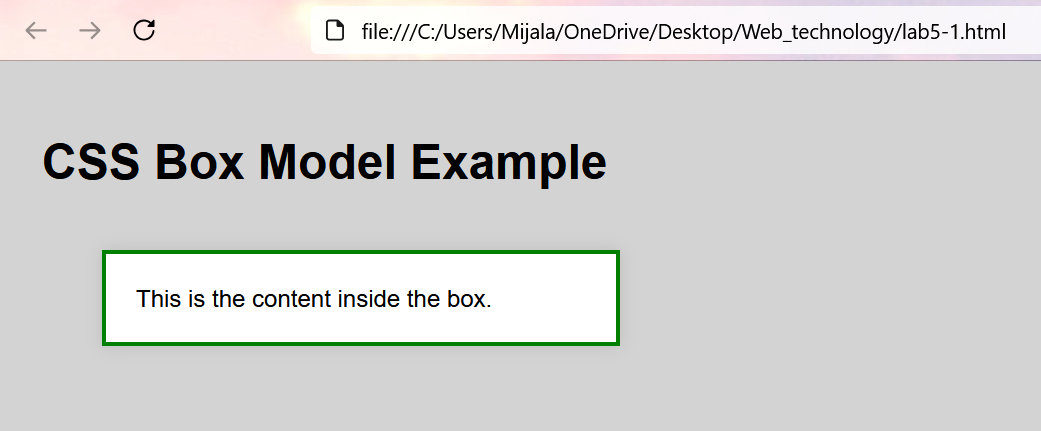
    margin: 40px;

    /\* Just for visualization \*/

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

**Output:**

****

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>Normal Flow Box Layout</title>

  <link rel="stylesheet" href="style2.css">

</head>

<body>

  <h1>Normal Flow Box Layout</h1>

  <div class="box box1">

    <p>This is a block-level box (Box 1).</p>

  </div>

  <div class="box box2">

    <p>This is another block-level box (Box 2).</p>

  </div>

  <span class="inline-box">Inline Box 1</span>

  <span class="inline-box">Inline Box 2</span>

</body>

</html>

**CSS Code:**

/\* Common box styles \*/

.box {

  display: block; /\* Default for divs \*/

  background-color: lightblue;

  padding: 20px;

  margin-bottom: 20px;

  border: 2px solid royalblue;

  width: 300px;

}

/\* Second box with different background to differentiate \*/

.box2 {

  background-color: lightgreen;

}

/\* Inline boxes \*/

.inline-box {

  display: inline;

  background-color: gold;

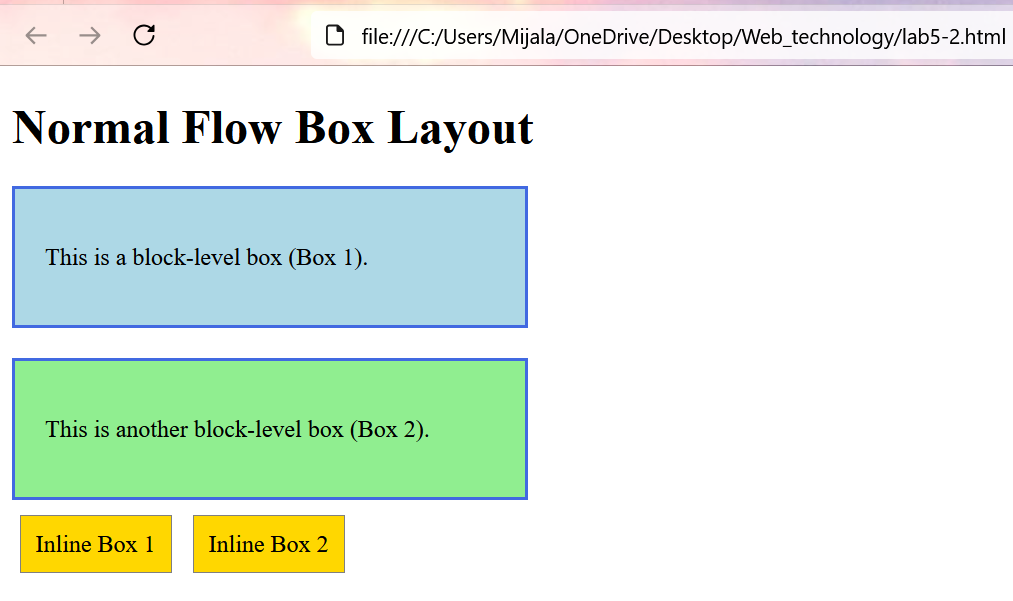
  padding: 10px;

  margin: 5px;

  border: 1px solid gray;

}

**Output:**

****

**LAB 6**

**Title:** 1)Write a web page using HTML and CSS to demonstrate Box Shadows, different Text Effects, and shadow;

* Write an HTML & CSS code snippet to create a box with a red shadow effect and a border-radius of 15px.
* Modify the given code to add multiple box shadows with different colors and positions.
* Create a new <div> element with a neon text glow effect using text-shadow.
* Write CSS to apply a hover effect that changes the box shadow color when the mouse is over the box.
* Create an HTML button with a 3D depth effect using box-shadow.
* Create a gradient background for the box along with a soft shadow effect.
* Apply a shadow effect only on the right side of the box.
* Write a CSS rule that applies different text shadows for h1 and p elements.

**Theory:**

**Box Shadows:** The CSS box-shadow property applies one or more shadows to an element. In its simplest use, we specify only a horizontal and a vertical shadow. The shadow's default color is the current text color.

Example: Specify a horizontal and a vertical shadow

div {  
  box-shadow: 10px 10px;  
}

The color parameter defines the color of the shadow.

Example: Specify a color for the shadow

div {  
  box-shadow: 10px 10px lightblue;  
}

The blur parameter defines the blur radius. The higher the number, the more blurred the shadow will be.

Example: Add a blur effect to the shadow

div {  
  box-shadow: 10px 10px 5px lightblue;  
}

The spread parameter defines the spread radius. A positive value increases the size of the shadow, a negative value decreases the size of the shadow.

Example: Set the spread radius of the shadow

div {  
  box-shadow: 10px 10px 5px 12px lightblue;

}

An element can also have multiple shadows

Example:

div {  
  box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;  
}

**Text Shadows:** The CSS text-shadow property applies a shadow to text. In its simplest use, we only specify the horizontal shadow (2px) and the vertical shadow (2px).

Example:

h1 {  
  text-shadow: 2px 2px;  
}

Add a color to the shadow

h1 {  
  text-shadow: 2px 2px red;  
}

Add a blur effect to the shadow

h1 {  
  text-shadow: 2px 2px 5px red;  
}

The following example shows white text with black shadow

h1 {  
  color: white;  
  text-shadow: 2px 2px 4px black;  
}

The following example shows a red neon glow shadow

h1 {  
  text-shadow: 0 0 3px red;  
}

To add more than one shadow to the text, you can add a comma-separated list of shadows. The following example shows a red and blue neon glow shadow:

h1 {  
  text-shadow: 0 0 3px red, 0 0 5px blue;  
}

**Text Effect:**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [text-justify](https://www.w3schools.com/cssref/css3_pr_text-justify.php) | Specifies how justified text should be aligned and spaced |
| [text-overflow](https://www.w3schools.com/cssref/css3_pr_text-overflow.php) | Specifies how overflowed content that is not displayed should be signaled to the user |
| [word-break](https://www.w3schools.com/cssref/css3_pr_word-break.php) | Specifies line breaking rules for non-CJK scripts |
| [word-wrap](https://www.w3schools.com/cssref/css3_pr_word-wrap.php) | Allows long words to be able to be broken and wrap onto the next line |
| [writing-mode](https://www.w3schools.com/cssref/css3_pr_writing-mode.php) | Specifies whether lines of text are laid out horizontally or vertically |

**3D Transform:**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [transform](https://www.w3schools.com/cssref/css3_pr_transform.php) | Applies a 2D or 3D transformation to an element |
| [transform-origin](https://www.w3schools.com/cssref/css3_pr_transform-origin.php) | Allows you to change the position on transformed elements |
| [transform-style](https://www.w3schools.com/cssref/css3_pr_transform-style.php) | Specifies how nested elements are rendered in 3D space |
| [perspective](https://www.w3schools.com/cssref/css3_pr_perspective.php) | Specifies the perspective on how 3D elements are viewed |
| [perspective-origin](https://www.w3schools.com/cssref/css3_pr_perspective-origin.php) | Specifies the bottom position of 3D elements |
| [backface-visibility](https://www.w3schools.com/cssref/css3_pr_backface-visibility.php) | Defines whether or not an element should be visible when not facing the screen |

**CSS Gradients:**

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines three types of gradients:

* Linear Gradients (goes down/up/left/right/diagonally)
* Radial Gradients (defined by their center)
* Conic Gradients (rotated around a center point)

CSS Linear Gradients

To create a linear gradient, we must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax:

background-image: linear-gradient(*direction*, *color-stop1*, *color-stop2, ...*);

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Box Shadows & Text Effects Demo</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background-color: lightgray;

      padding: 30px;

      text-align: center;

    }

    h1 {

      text-shadow: 2px 2px 4px black;

      color: gray;

    }

    p {

      text-shadow: 1px 1px 2px rgb(7, 7, 7);

      color: black;

    }

    /\* Box with red shadow and border-radius \*/

    .red-shadow-box {

      width: 300px;

      margin: 20px auto;

      padding: 20px;

      background-color: white;

      border-radius: 15px;

      box-shadow:0 0 15px red;

    }

    /\* Multiple colored box shadows \*/

    .multi-shadow-box {

      width: 300px;

      margin: 20px auto;

      padding: 20px;

      background-color: white;

      border-radius: 15px;

      box-shadow:

        5px 5px 10px red,

        -5px -5px 10px blue,

        5px -5px 10px green;

    }

    /\* Neon text glow effect \*/

    .neon-text {

      font-size: 2rem;

      color: aqua;

      text-shadow:

        0 0 5px aqua,

        0 0 10px aqua,

        0 0 20px aqua,

        0 0 40px aqua;

        margin: 40px 0;

    }

    /\* Hover effect to change box shadow \*/

    .red-shadow-box:hover {

      box-shadow: 0 0 15px blue;

    }

    /\* 3D effect button \*/

    .button-3d {

      padding: 12px 24px;

      font-size: 16px;

      background-color: green;

      color: white;

      border: none;

      border-radius: 8px;

      box-shadow: 0 4px darkgreen;

      cursor: pointer;

      transition: transform 0.2s, box-shadow 0.2s;

    }

    .button-3d:active {

      transform: translateY(4px);

      box-shadow: 0 1px darkgreen;

    }

    /\* Gradient background + soft shadow \*/

    .gradient-box {

      width: 300px;

      margin: 20px auto;

      padding: 20px;

      border-radius: 15px;

      background: linear-gradient(to right, pink, peachpuff);

      box-shadow: 0 10px 20px gray;

    }

    /\* Right side shadow only \*/

    .right-shadow-box {

      width: 300px;

      margin: 20px auto;

      padding: 20px;

      background-color: white;

      border-radius: 10px;

      box-shadow: 10px 0 10px gray;

    }

  </style>

</head>

<body>

  <h1>CSS Shadow & Text Effects</h1>

  <p>This page demonstrates various shadow effects using color names in CSS.</p>

  <div class="red-shadow-box">Box with red shadow and rounded corners</div>

  <div class="multi-shadow-box">Box with multiple colored shadows</div>

  <div class="neon-text">Neon Glow Text</div>

  <button class="button-3d">3D Shadow Button</button>

  <div class="gradient-box">Box with Gradient & Soft Shadow</div>

  <div class="right-shadow-box">Box with Shadow on the Right Side</div>

</body>

</html>

**Output:**

****

2. Write HTML and CSS code to change the background color of a webpage when the screen width is less than 600px.  
Create a responsive navigation bar that converts into a dropdown menu for screens smaller than 768px.  
Write a media query to hide an image when the screen width is less than 480px.  
Use media queries to increase font size for h1 headings when the screen width exceeds 1024px.  
Write a media query to change the text alignment to center on screens smaller than 768px.  
Write a media query that changes the padding and margins of a container when viewed on a mobile device.

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

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

  <title>Responsive Design Example</title>

  <style>

    body {

      margin: 0;

      font-family: Arial, sans-serif;

      background-color: lightgray;

    }

    h1 {

      font-size: 24px;

    }

    nav {

      background-color: black;

      overflow: hidden;

    }

    nav a {

      float: left;

      display: block;

      color: white;

      text-align: center;

      padding: 14px 16px;

      text-decoration: none;

    }

    .dropdown-menu {

      display: none;

      background-color: black;

    }

    .dropdown-menu a {

      display: block;

    }

    .container {

      padding: 20px;

      margin: 20px;

      background-color: white;

    }

    img.responsive-img {

      width: 100%;

      height: auto;

    }

    /\* 1. Background color change for screen width < 600px \*/

    @media screen and (max-width: 599px) {

      body {

        background-color: pink;

      }

    }

    /\* 2. Responsive Navbar into Dropdown below 768px \*/

    @media screen and (max-width: 768px) {

      nav a {

        display: none;

      }

      .dropdown-menu {

        display: block;

      }

    }

    /\* 3. Hide image when screen width < 480px \*/

    @media screen and (max-width: 479px) {

      img.responsive-img {

        display: none;

      }

    }

    /\* 4. Increase h1 font size above 1024px \*/

    @media screen and (min-width: 1025px) {

      h1 {

        font-size: 48px;

      }

    }

    /\* 5. Center text alignment below 768px \*/

    @media screen and (max-width: 768px) {

      .container {

        text-align: center;

      }

    }

    /\* 6. Change padding and margin on mobile devices (max 600px) \*/

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

      .container {

        padding: 10px;

        margin: 10px;

      }

    }

  </style>

</head>

<body>

  <!-- Navigation -->

  <nav>

    <a href="#">Home</a>

    <a href="#">About</a>

    <a href="#">Contact</a>

    <div class="dropdown-menu">

      <a href="#">Menu</a>

    </div>

  </nav>

  <!-- Content -->

  <div class="container">

    <h1>Responsive Design in Action</h1>

    <p>This container's layout and typography change based on screen size.</p>

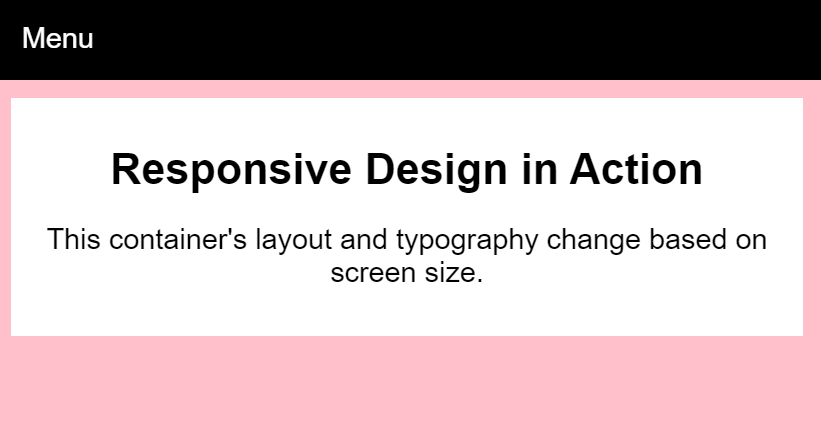
  </div>

</body>

</html>

**Output:**

****



3) Write a responsive web page using Bootstrap.

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

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

  <title>Responsive Bootstrap Page</title>

  <!-- Bootstrap 5 CSS CDN -->

  <link

    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"

    rel="stylesheet"

  />

</head>

<body>

  <!-- Navigation Bar -->

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

    <div class="container-fluid">

      <a class="navbar-brand" href="#">MySite</a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"

              data-bs-target="#navbarNav" aria-controls="navbarNav"

              aria-expanded="false" aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarNav">

        <ul class="navbar-nav ms-auto">

          <li class="nav-item"><a class="nav-link active" href="#">Home</a></li>

          <li class="nav-item"><a class="nav-link" href="#">About</a></li>

          <li class="nav-item"><a class="nav-link" href="#">Services</a></li>

          <li class="nav-item"><a class="nav-link" href="#">Contact</a></li>

        </ul>

      </div>

    </div>

  </nav>

  <!-- Hero Section -->

  <div class="container text-center my-5">

    <h1 class="display-4">Welcome to My Responsive Website</h1>

    <p class="lead">This page adapts perfectly to any screen size using Bootstrap!</p>

    <a href="#" class="btn btn-primary">Learn More</a>

  </div>

  <!-- Image and Text Section -->

  <div class="container my-5">

    <div class="row align-items-center">

      <div class="col-md-6">

<img src="flower.jpeg" width="150" height="100"class="img-fluid rounded" alt=" " />

</div>

      <div class="col-md-6">

        <h2>Why Choose Us?</h2>

        <p>

          We build responsive, beautiful websites using modern tools like Bootstrap.

          Our layout automatically adjusts for mobile, tablet, and desktop screens.

        </p>

        <a href="#" class="btn btn-success">Get Started</a>

      </div>

    </div>

  </div>

  <!-- Footer -->

  <footer class="bg-dark text-white text-center py-3">

    &copy; 2025 MySite. All rights reserved.

  </footer>

  <!-- Bootstrap 5 JS Bundle CDN -->

  <script

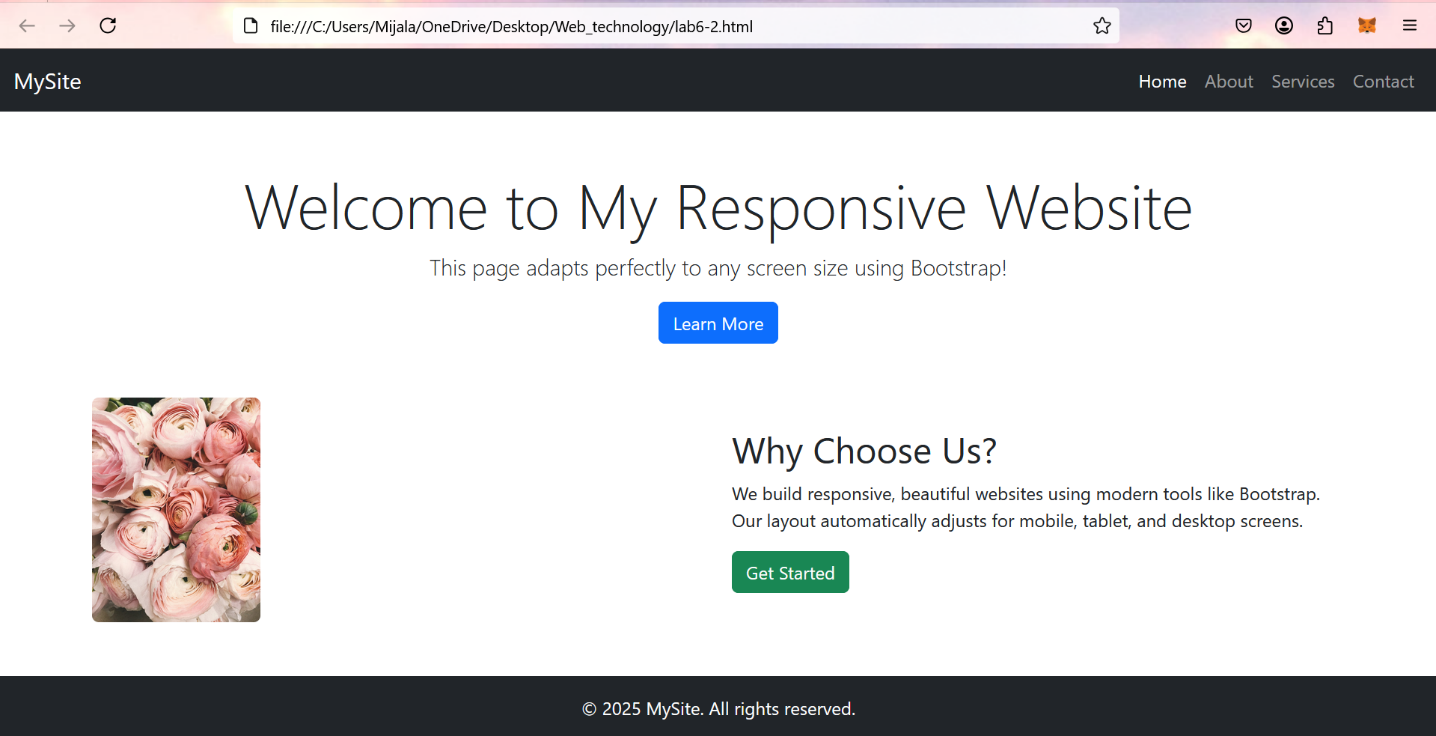
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"

  ></script>

</body>

</html>

**Output:**

****

**LAB 7**

**Title:** Write a script using variables (var, const and let). Also show the output using console.log() and innerHTML.

**Theory:**

**Var**

* Used in older JavaScript (not commonly used now).
* Has function scope — works inside the whole function where it's declared.
* Can be re-declared and reassigned.
* May cause bugs due to its wide scope, so it's usually avoided in modern code.

**Const**

* Also introduced in ES6 (modern JavaScript).
* Has block scope like let.
* Cannot be reassigned or re-declared after the first assignment.
* Use const for values that should stay the same (e.g., fixed names, settings).

**Let**

* Introduced in modern JavaScript (ES6).
* Has block scope — works only inside { } where it's declared.
* Can be reassigned, but not re-declared in the same block.
* Use let when the variable value needs to change (e.g., in loops or conditions).

**Example:**

<script>

var name = "John"; // can be re-declared and re-assigned

let age = 25; // can be re-assigned but not re-declared

const country = "India"; // cannot be re-assigned or re-declared

console.log(name);

console.log(age);

console.log(country);

</script>

**console.log()**

* Used to print messages to the browser’s console (for debugging).
* You won’t see the output on the web page, but in the developer tools console (F12 → Console tab).

**innerHTML**

* Used to change or get the content inside an HTML element from JavaScript.

**HTML Code:**

<!DOCTYPE html>

<html>

<head>

  <title>Simple Addition</title>

</head>

<body>

  <h2>Calculate a + b = c</h2>

  <p id="output"></p>

  <script>

    // Using var for a and b

    var a = 11;

    const b = 6;

    // Using let for the result c

    let c = a + b;

    // Using const for a constant message

    const message = "The result of " + a + " + " + b + " is: ";

    // Console Output

    console.log(message + c);

    // Web Page Output

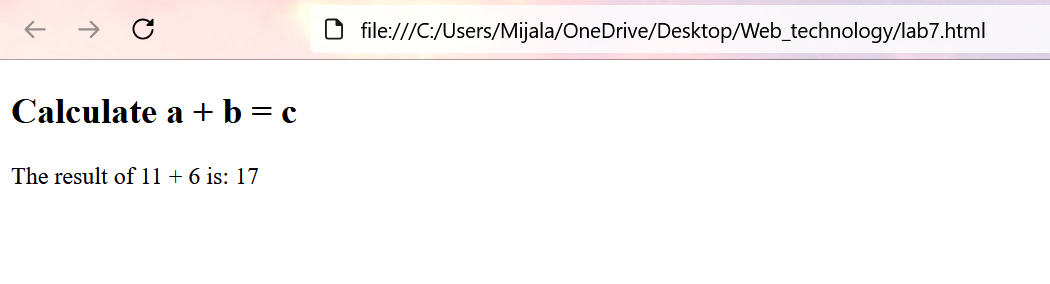
    document.getElementById("output").innerHTML = message + c;

  </script>

</body>

</html>

**Output:**

****

**LAB 8**

**Title:** Create an HTML signup form with fields Name, Email, Password, and Age, Validate the form using JavaScript. Write functions for validating each of the elements. All of the fields should not be empty. The Email address should be a valid email, the password should be of length at least 6 and should start with the alphabet and end with a digit. The age should be between 8 and 60.

**HTML Code:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Signup Form with Validation</title>

  <style>

    body { font-family: Arial; }

    .error { color: red; }

    .success { color: green; }

    .output { margin-top: 20px; padding: 10px; border: 1px solid #ccc; }

  </style>

</head>

<body>

  <h2>Signup Form</h2>

  <form onsubmit="return validateForm()">

    <label>Name: <input type="text" id="name"></label><br><br>

    <label>Email: <input type="email" id="email"></label><br><br>

    <label>Password: <input type="password" id="password"></label><br><br>

    <label>Age: <input type="number" id="age"></label><br><br>

    <div id="error" class="error"></div><br>

    <button type="submit">Submit</button>

  </form>

  <!-- Output Section -->

  <div id="output" class="output" style="display: none;"></div>

  <script>

    function validateName(name) {

      return name.trim() !== "" ? null : "Name cannot be empty.<br>";

    }

    function validateEmail(email) {

      const pattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;

      if (email.trim() === "") return "Email cannot be empty.<br>";

      if (!pattern.test(email)) return "Invalid email format.<br>";

      return null;

    }

    function validatePassword(password) {

      const pattern = /^[a-zA-Z].{4,}\d$/;

      if (password === "") return "Password cannot be empty.<br>";

      if (password.length < 6) return "Password must be at least 6 characters long.<br>";

      if (!pattern.test(password)) return "Password must start with a letter and end with a digit.<br>";

      return null;

    }

    function validateAge(age) {

      if (age === "") return "Age cannot be empty.<br>";

      if (age < 8 || age > 60) return "Age must be between 8 and 60.<br>";

      return null;

    }

    function validateForm() {

      const name = document.getElementById("name").value;

      const email = document.getElementById("email").value;

      const password = document.getElementById("password").value;

      const age = parseInt(document.getElementById("age").value);

      let error = "";

      error += validateName(name) || "";

      error += validateEmail(email) || "";

      error += validatePassword(password) || "";

      error += validateAge(age) || "";

      if (error) {

        document.getElementById("error").innerHTML = error;

        document.getElementById("output").style.display = "none";

        return false;

      }

      // Show output

      document.getElementById("error").innerHTML = "";

      const output = `

        <div class="success"><strong>Form submitted successfully!</strong></div>

        <p><strong>Entered Details:</strong></p>

        Name: ${name}<br>

        Email: ${email}<br>

        Age: ${age}<br>

      `;

      document.getElementById("output").innerHTML = output;

      document.getElementById("output").style.display = "block";

      return false; // Prevent actual form submission

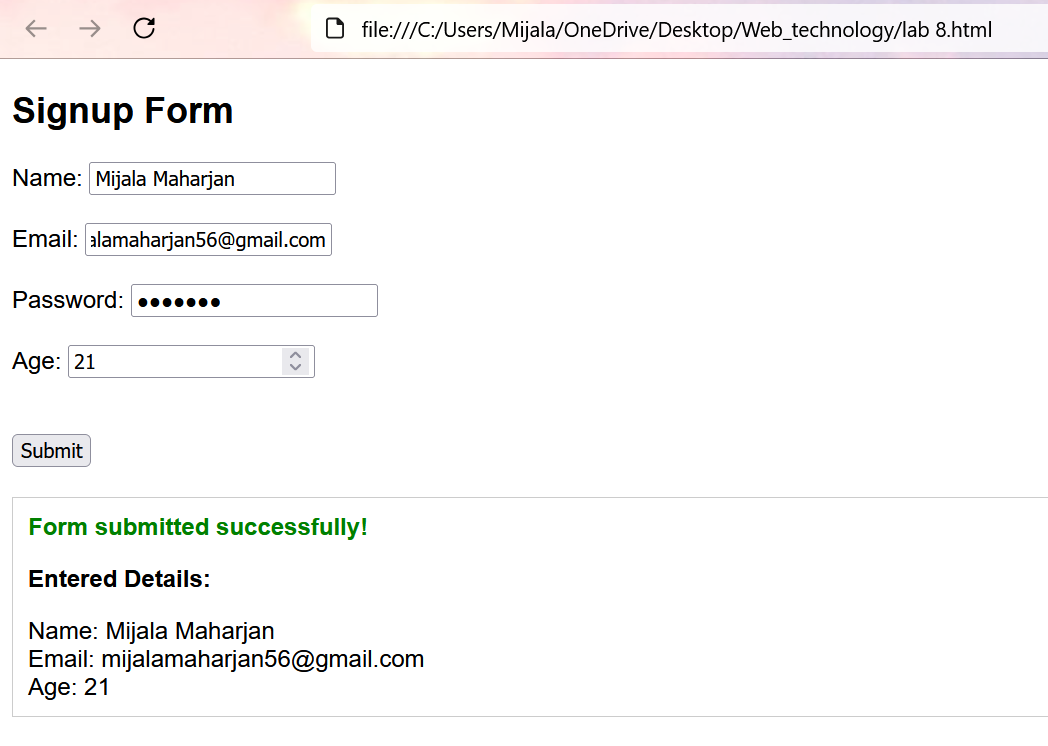
    }

  </script>

</body>

</html>

**Output:**

****

**LAB 9**

**Title:** Write a PHP script to create a multidimensional array name Product that will contain pcode, pname. and price. Initialize the array with at least three instances. Also, Write an HTML script to display the initialized values of the array in an HTML Table.

**Theory:**

A multidimensional array in PHP is an array that contains one or more arrays inside it. It's used to store complex data in a structured way, like rows and columns in a table. Each inner array can be either indexed or associative. For example, a list of products where each product has a code, name, and price can be stored as a multidimensional array. You can access the data using nested keys or indexes and loop through it using foreach for easy data display or processing.

**PHP Code:**

<?php

//Creating a multidomensional array named $Product

$Product=array(

    array(

        "pcode"=>"01", //Product Code

        "pname"=>"Phone", //Product Name

        "price"=>"60000",), //Product Price

        array(

            "pcode"=>"02",

            "pname"=>"Laptop",

            "price"=>"105500",

        ),

        array(

            "pcode"=>"03",

            "pname"=>"Tablet",

            "price"=>"12000",

        )

    );

    //Displaying Products details.

echo "Product code is ".$Product[0]["pcode"].

    " and product name is ".$Product[0]["pname"].

    " and price is :".$Product[0]["price"];

    echo"<br>";

echo "Product code is ".$Product[1]["pcode"].

    " and product name is ".$Product[1]["pname"].

    " and price is :".$Product[1]["price"];

    echo"<br>";

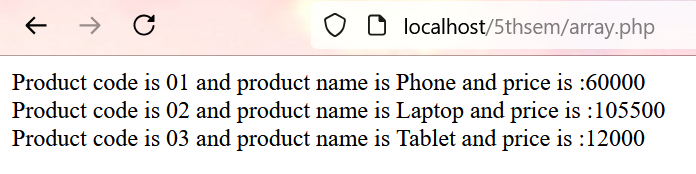
echo "Product code is ".$Product[2]["pcode"].

    " and product name is ".$Product[2]["pname"].

    " and price is :".$Product[2]["price"];

?>

**Output:**

****

**LAB 10**

**Title:** Write a PHP program to Connect Database, Create, Select, Delete, Update Records in a table and also insert multiple data.

**Theory:**

1. **Connect to database**

To connect PHP with a MySQL database, you use the new mysqli() function by passing the server name, username, password, and database name. This establishes a connection, and it’s important to check if the connection was successful using $conn->connect\_error.

1. **Create a Table**

Once connected, you can create a table using an SQL CREATE TABLE statement.

1. **Insert records into table**

To add data to a table, you use the INSERT INTO statement. It can be used to insert a single row or multiple rows at once by listing multiple sets of values.

1. **Select records from table**

The SELECT statement is used to retrieve data from a table. You can fetch all records or filter specific ones using WHERE clauses.

1. **Update records**

To modify existing records, the UPDATE statement is used. You specify the columns to change and the condition using WHERE to ensure only the intended rows are updated.

1. **Delete records**

The DELETE statement and a WHERE clause are used to remove records from a table. This ensures only the targeted rows are deleted. Without WHERE, all rows would be removed.

**PHP Code:**

<?php

$servername="localhost";

$username="root";

$password="";

//creating a connection here with mysql

$conn=mysqli\_connect($servername,$username,$password);

//checking connection here

if(!$conn){

    die("connection failed:".mysqli\_connect\_error());

}

//SQL qurey to ceating a database in MySQL

$sql="CREATE DATABASE School";

if(mysqli\_query($conn,$sql)){

    echo"Database created successfully";

}else{

    echo"Error!creating database:".mysqli\_error($conn);

}

mysqli\_close($conn);

?>

Creating a table

<?php

$sql="CREATE TABLE students(

    id int auto\_increment primary key,

    stname varchar(30) not null,

    email varchar(40),

    mobile varchar(10) not null

)";

if(mysqli\_query($conn,$sql)){

    echo"Table student created sucesfully";

}else{

    echo"Error!creating table:".mysqli\_error($conn);

}

mysqli\_close($conn)

?>

Insert records into table

<?php

//SQL query to inserting data in students table

$sql="INSERT INTO students(stname,email,mobile)

VALUES('mijala','mijala56@gmail.com','9894893009')";

if(mysqli\_query($conn,$sql)){

    echo"New record inserted successfully";

}else{

    echo"Error:".$sql."<br>".mysqli\_error($conn);

}

mysqli\_close($conn);

?>

Insert multiple record into table

<?php

//SQL query to inserting data in students table

$sql="INSERT INTO students(stname,email,mobile)

VALUES('Bijen','bijen56@gmail.com','9894893009'),

('Tisha','tisha@gamil.com','987654272'),

('Salina','salina@gmail.com','8726562287')";

if(mysqli\_query($conn,$sql)){

    echo"New record inserted successfully";

}else{

    echo"Error:".$sql."<br>".mysqli\_error($conn);

}

mysqli\_close($conn);

?>

Select records from table

<?php

//SQL query to deleting record

$sql="DELETE FROM students WHERE id=3";

if(mysqli\_query($conn,$sql)){

    echo"Record deleted successfully";

}else{

    echo"Error!deleted record.".mysqli\_error($conn);

}

mysqli\_close($conn);

?>

Update records

<?php

//SQL Query update data.

$sql="UPDATE students SET stname='Reeva'WHERE mobile='8726562287'";

if(mysqli\_query($conn,$sql)){

    echo"Record updated successfully";

}else{

    echo"Error!updating record.".mysqli\_error($conn);

}

mysqli\_close($conn);

?>

Delete records

<?php

//SQL Query Delete data.

$sql="DELETE FROM students WHERE id=3";

if(mysqli\_query($conn,$sql)){

   echo"Record deleted sucessfully";

}else{

    echo"Error deleting record.".mysqli\_error($conn);

}

mysqli\_close($conn);

?>

**Output:**

****