Experiment-4.1

AIM:

1. Selector forms

a. Write a program to apply different types of selector forms

I. Simple selector (element, id, class, group, universal)
II. Combinator selector (descendant, child, adjacent sibling, general sibling)
III. Pseudo-class selector
IV. Pseudo-element selector
V. Attribute selector

DESCRIPTION:

CSS selector forms define **how to choose elements** in an HTML document for styling.

Types :

I. Simple Selectors

Simple selectors target elements directly by name, id, class, group, or universally **CSS Selector Forms**.

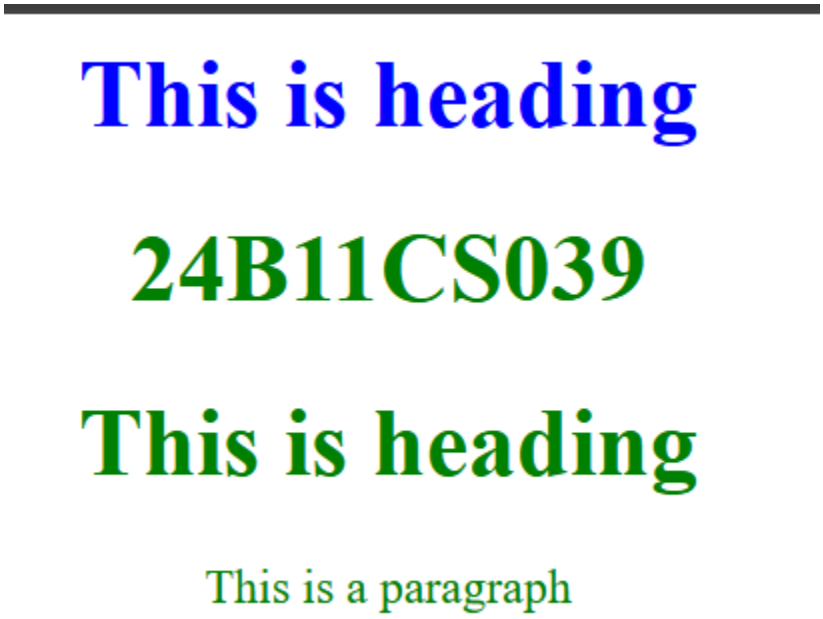
1. **Element Selector:** Selects all elements of a given tag.
2. **ID Selector:** Selects an element with a specific id.
3. **Class Selector:** Selects elements with a specific class.
4. **Group Selector:** Allows applying styles to multiple elements at once.
5. **Universal Selector:** Selects **all elements** on the page.

Program:

```
<html>
<body>
<head>
<style>
h1
{
color:green;
}

.h{
color:blue;
}
#s{
color:green;
}

*{
  text-align:center;
}

</style>
</head>
<body>
```

```
<h1 class="h">This is heading</h1>

<h1>24B11CS039</h1>
<h1>This is heading</h1>
<p id="s">This is a paragraph</p>

</body>
<html>
```

Output:



Result:

4.2
Description:
II. Combinator Selectors
Used to select elements based on relationships (parent-child, siblings, etc.)
1. **Descendant Selector (Space):** Selects elements inside another element.
2. **Child Selector (>)**:Selects **direct children** of an element.
3. **Adjacent Sibling Selector (+)**:Selects the first element **immediately after** a given element.
The **first <p> after <h1>** will be green.
4. **General Sibling Selector (~):** Selects **all siblings** after a given element.

Program:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title Combinator selectors></title>
    <style>
      div p{
        color: orange;
        font-size:16px;
      }
      /* child selector (>)*/
      div >ul{
        background-color:aqua ;
        padding:10px;
      }
      /* Adjacent sibling selector (+)*/
      h2 + p{
        color:blue;
        font-style: italic;
      }
      /* General sibling  selector(~)*/
      h3 ~ p {
        color:red;
        text-decoration: underline;
      }
    </style>

  </head>
  <body>
    <h1>Combinator selector demo</h1>
    <h4>24B11CS039</h4>  <!--relation between tags or elements-->
    <div>
      <p>This is descendant of div(styled using Descendant selector)</p>
      <section>
        <p>This is nested deeper inside div(still a descendant)</p>
      </section>
```

```
        <ul>
          <li> Child of div- styled with child selector</li>
          <li> Another list item</li>
        </ul>
      </div>

      <h2> Heading 2</h2>
      <p> This paragraph is an adjacent sibling of h2(Styled using + selector)</p>
      <p> This paragraph is not an adjacent to h2</p>

      <h3> Heading 3</h3>
      <p> This is general sibling of h3(styled using ~ selector).</p>
      <p> Another generalsiblng of h3.</p>
    </body>
</html>
```

Output:

Combinator selector demo

24B11CS039

This is descendant of div(styled using Descendant selector)

This is nested deeper inside div(still a descendant)

- Child of div- styled with child selector
- Another list item

Heading 2

This paragraph is an adjacent sibling of h2(Styled using + selector)

This paragraph is not an adjacent to h2

Heading 3

This is general sibling of h3(styled using ~ selector).

Another generalsiblng of h3.

Result:

4.3

III. Pseudo-Class Selector

- **Description:** Selects elements in a specific state (hovered, visited, first-child, etc.)
- **Syntax:**a:hover {
  color: orange;
  }
  p:first-child {
  font-style: italic;
  }

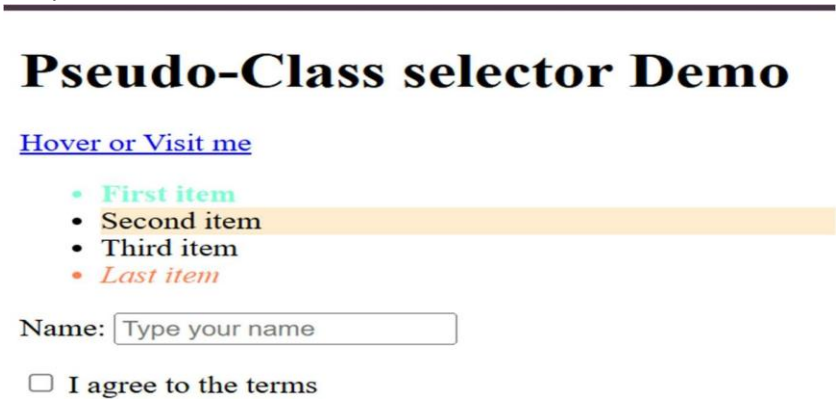
First example changes color when hovered, second styles first <p> inside a container.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Pseudo class selectors</title>
  <style>
    /*: hover- when mouse is over the element*/
    a:hover{
      color:red;
      text-decoration: underline;
    }
    /*: first -child - targets the first child element*/
    ul li:first-child{
      font-weight: bold;
      color:aquamarine;
    }
    /*last-child - targets the last child element*/
    ul li:last-child{
      font-style: italic;
      color: coral;
    }
    /* :nth-child - targets the nth child*/
    ul li:nth-child(2)
    {
      background-color: blanchedalmond;
    }
    /* :focus - applies when input is focused*/
    input:focus{
      border: 2px solid orange;
      outline: none;
    }
    /* checked - when checkbox is selected*/
    input[type="checkbox"]:checked+label{
      color: cornflowerblue;
      font-weight: bold;
    }
```

```
      /* :visited - visited links */
      a:visited{
        color: purple;
      }
    </style>
</head>
  <body>
      <h1> Pseudo-Class selector Demo</h1>
      <a href="https://www.codechef.com/login?destination=/dashboard">Hover or Visit
me</a>
      <ul>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
        <li>Last item</li>
      </ul>
      <form>
        <p>
          <label for="name">Name:</label>
          <input type="text" id="name" placeholder="Type your name">
        </p>
        <p>
          <input type="checkbox" id="agree">
          <label for="agree">I agree to the terms</label>
        </p>
      </form>
  </body>
</html>
```

Output:



Result:

4.4

IV. Pseudo-Element Selector

- Description: Selects parts of an element (like first line, first letter, etc.)
- Syntax:

```
p::first-line {
 font-weight: bold;
}
p::first-letter {
 font-size: 200%;
 color: red;
}
```

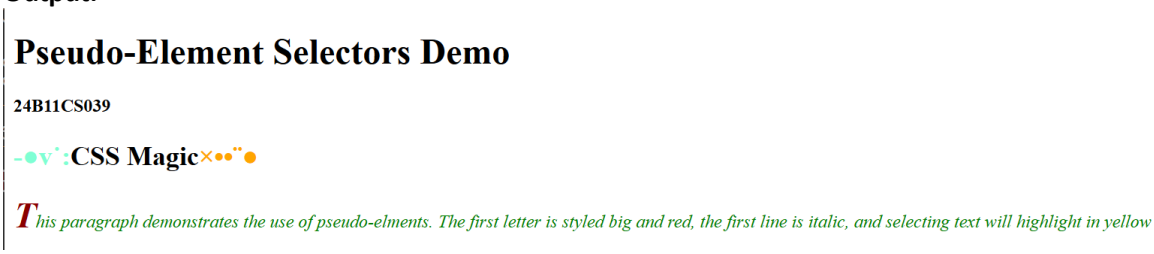
This makes first line bold and first letter big & red.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Pseudo Class Selectors</title>
   <style>
      /*::first-letter-style the first letter of a paragraph*/
      p::first-letter{
         font-size:  200%;
         font-weight: bold;
         color: darkred;
      }
      /*::first-line -style the first line of a paragraph*/
      p::first-line{
         font-style: italic;
         color: green;
      }
      /*:: before- insert content before an element*/
      h2::before{
         content: "";
         color: aquamarine;
      }

      /* ::after-insert content after an element*/
      h2::after{
         content: "";
         color: orange;
      }
       /* ::selection- style selected text*/
      ::selection{
         background-color: bisque;
         color: chartreuse;
      }
```

```
    </style>
</head>
<body>
    <h1>Pseudo-Element Selectors Demo</h1>
    <h4>24B11CS039</h4>
    <h2>CSS Magic</h2>
    <p> This paragraph demonstrates the use of pseudo-elments.
        The first letter is styled big and red, the first line is italic, and selecting text will highlight
in yellow</p>
</body>
</html>
```

Output:



Result:

4.5

V. Attribute Selector

- **Description:** Selects elements based on attributes or attribute values.
- **Syntax:**

```
input[type="text"] {
border: 2px solid blue;
}
a[target="_blank"] {
  color: purple;
}
```

First applies style to text input fields, second applies to links opening in new tabs.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Simple Attribute Selector</title>
   <style>
      input[type="text"]
   {
   background-color:lightyellow;
   }
   a[target = "_blank"] {
   color:red;
   }
   img[alt]{
   border:5px solid black;
   }
   </style>
</head>
<body>
   <h2> Attribute Selector Demo</h2>
   <h4>24B11CS039</h4>
   <input type="text" placeholder="Enter Your Name"><br><br>
   <a href="https://www.codechef.com/users/smack_trust_39">Code chef</a><br><br>
   <img src="c:\Users\DEEVENA\OneDrive\Documents\Downloads\IMG-20250427-
   WA0027.jpg "width="100" height="100">
</body>
</html>
```

Output:

Attribute Selector Demo

24B11CS039

Enter Your Name

Code chef

Result:

AIM:

5)CSS with Color, Background, Font, Text and CSS Box Model

a) Write a program to demonstrate the various ways you can reference a color in CSS.

DESCRIPTION:

The **color property in CSS** can be defined in multiple ways, allowing developers flexibility in how they choose and apply colors to elements. Colors can be referenced using:

1. **Named Colors:**
   - CSS supports predefined color names like red, blue, green, yellow, etc.
   - Example: color: red;
2. **Hexadecimal Values (#RRGGBB or #RGB):**
   - Uses a six-digit or three-digit hex code to represent red, green, and blue values.
   - Example: color: #ff0000; (red), color: #0f0; (green).
3. **RGB Values (rgb):**
   - Defines a color using red, green, and blue values ranging from 0 to 255.
   - Example: color: rgb(255, 0, 0); (red).
4. **RGBA Values (rgba):**
   - Similar to RGB but adds an alpha channel for transparency (0.0 to 1.0).
   - Example: color: rgba(0, 0, 255, 0.5); (semi-transparent blue).
5. **HSL Values (hsl):**
   - Defines colors using hue (0–360), saturation (%), and lightness (%).
   - Example: color: hsl(120, 100%, 50%); (green).
6. **HSLA Values (hsla):**
   - Same as HSL but with an alpha (transparency) component.
   - Example: color: hsla(240, 100%, 50%, 0.3); (transparent blue).

→In short, CSS allows colors to be defined in **different formats** depending on readability, flexibility, and design needs, making it easy for developers to control the look and feel of a webpage.

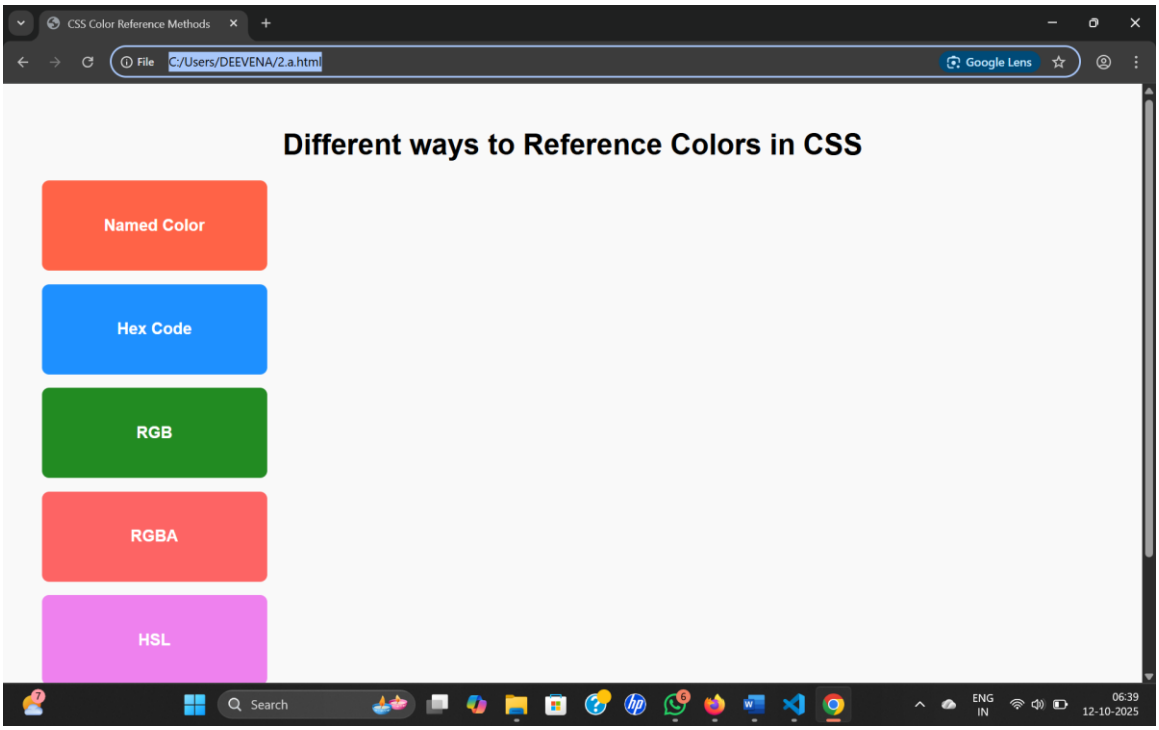
PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>CSS Color Reference Methods</title>
 <style>
   body {
     font-family:Arial,sans-serif;
     padding:20px;
     background-color:#f9f9f9;
     }
 h1 {
  text-align:center;
 }
 .color-box {
```

```
    width:250px;
    height:100px;
    color:white;
    display:flex;
    align-items:center;
    justify-content:center;
    margin:15px;
    font-size:18px;
    font-weight:bold;
    border-radius:8px;
    box-shadow:0 4px 6px rgba(0,0,0,0,1);
   }

 /*1.Named color*/
  .named-color {
      background-color:tomato;/*CSS Named Color*/
      }

/*2.Hexadecimal notation*/
.hex-color {
   background-color:#1E90FF;/*Dodger Blue*/
  }

/*3.RGB notation*/
  .rgb-color {
    background-color:rgb(34,139,34);/*Forest  Green*/
   }

/*.RGBS notation(with transparency)*/
 .rgba-color {
    background-color:rgba(255,0,0,0.6);/*Red with transparency*/
   }

/*5.HSL notation*/
  .hsl-color {
      background-color:hsl(300,76%,72%);/*Light  Purple*/
    }

/*6.HSLA notation(with trancparency)*/
  .hsla-color {
    background-color:hsla(39,100%,50%,0.7);/*Orange with transparency*/
    }
</style>
 </head>
<body>

  <h1>Different ways to Reference Colors in CSS</h1>
```

```
  <div class="color-box named-color">Named Color</div>
  <div class="color-box hex-color">Hex Code</div>
  <div class="color-box rgb-color">RGB</div>
  <div class="color-box rgba-color">RGBA</div>
  <div class="color-box hsl-color">HSL</div>
  <div class="color-box hsla-color">HSLA</div>
</body>
</html>
```

OUTPUT:



RESULT:

5)b.

AIM:Write a CSS rule that places a background image halfway down the page, tilting it horizontally. The image should remain in place when the user scrolls up or down.

DESCRIPTION:

The **background-image property in CSS** allows you to add images behind HTML elements. To place a background image halfway down the page, fix it so it doesn't move while scrolling, and repeat it horizontally, we use a combination of background-position, background-repeat, and background-attachment.

- **background-image** → specifies the image file to be used as background.
- **background-position** → positions the image on the page; using center 50% places it halfway vertically and centered horizontally.
- **background-repeat** → controls tiling; setting it to repeat-x repeats the image horizontally across the page.
- **background-attachment** → fixes the background image so it remains in place while the user scrolls.

→ In short, this CSS rule ensures the background image appears halfway down, stretches horizontally through tiling, and stays fixed in position even when the page is scrolled.

PROGRAM:

```
<!doctype html>

<html>

<head>

<meta charset="utf-8" />

<title>Fixed tilted background (horizontal rotate)</title>

<style>

/* Container content so you can scroll the page */

body {

  min-height: 300vh; /* make page scrollable to test fixed behaviour */

 margin: 0;

  font-family: system-ui, Arial, sans-serif;

}

/* Fixed background image element */.tilted-bg {
```

```
  position: fixed;
  left: 50%;           /* center horizontally */
  top: 50vh;           /* halfway down the viewport */
  transform: translate(-50%, -50%) rotate(-12deg);
  width: 60vw;         /* size the image element */
  height: 40vh;
  background-image: url('spasskaya_tower.jpg'); /* place image in same folder */
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  pointer-events: none; /* so it won't block clicks */
  border-radius: 12px;
  box-shadow: 0 10px 30px rgba(0,0,0,0.25);
  z-index: 9999;
  opacity: 0.95;
  will-change: transform;
}
/* sample page content */
main {
  padding: 2rem;
}
</style>
</head>
<body>
  <div class="tilted-bg" aria-hidden="true"></div>
  <main>
    <h1>Page Content</h1>
    <p>Scroll to see the tilted background remain fixed halfway down the viewport.</p>
```

```
    <p>Adjust <code>width</code>, <code>height</code>, and rotation angle in the CSS as needed.</p>


<!-- Add content to scroll -->
    <p style="margin-top:140vh">This line is roughly below the fold to show scrolling.</p>
  </main>
</body>
</html>
```

OUTPUT:

RESULT:

AIM:

5)c.Write a program using the following terms related to CSS font and text:

i. font-size ii. font-weight iii. font-style

DESCRIPTION:

1. **font-size**
    - The font-size property in CSS is used to define the size of the text on a webpage.
    - It can be given in different units such as pixels (px), em (em), rem (rem), percentage (%), or predefined keywords like small, medium, large.
    - Example: font-size: 20px; makes the text larger, while font-size: 12px; makes it smaller.
2. **font-weight**
    - The font-weight property controls the **thickness (boldness)** of text characters.
    - It can take values like normal, bold, lighter, bolder, or numeric values ranging from 100 (thin) to 900 (extra bold).
    - Example: font-weight: bold; makes the text bold, while font-weight: 300; makes it lighter.
3. **font-style**
    - The font-style property specifies whether the text should appear in **normal**, **italic**, or **oblique** form.
    - normal shows plain text, italic makes the text slanted and stylized, and oblique also slants text but in a less decorative way than italic.
    - Example: font-style: italic; makes text slanted for emphasis.

→In summary, these three CSS properties (font-size, font-weight, and font-style) are essential for controlling **how text looks** on a webpage, making it more readable, attractive, and suitable for design requirements.

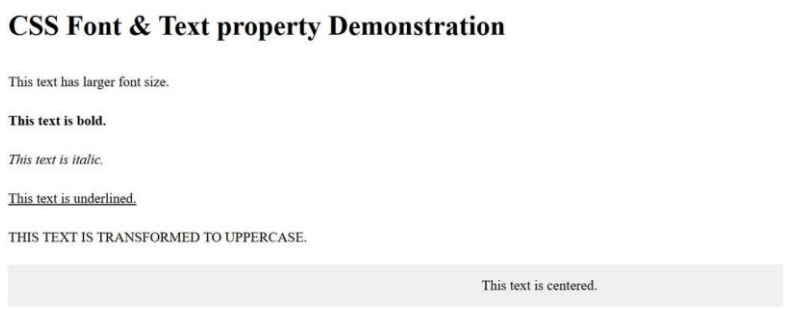
PROGRAM:

```
<!DOCTYPE html>

<html>

<head>

    <meta charset="UTP+8">
```

```
<title>CSS Font and Text Properties</title>
  <style>
    body {
      .font-family: Arial, sans-serif;
      padding: 20px;
      line-height:1.8;
    }
  /* 1. font-size */
      .font-size {
          front-size:24px; /* Increase text size */
      }
  /* 2. font-weight */
      .font-weight {
         font-weight: bold; /* Make text bold */
      }
  /* 3. font-style */
      .font-style {
          font-style: italic; /* Make text italic */
      }
/* 4. text-decoration */
      .text-decoration {
          text-decoration: underline; /* Add underline */
      }
  /* 5. text-transform */
      .text-transform {
         text-transform: uppercase; /* convert all letters to uppercase */
       }
```

```
 /* 6. text-align */
      .text-align {
         text-align: center; /* center align text */
         background-color: #f0f0f0;
         padding: 10px;
       }
</style>
</head>
 <body>
      <h1>CSS Font & Text property Demonstration</h1>
      <p class="font-size">This text has larger font size.</p>
      <p class="font-weight">This text is bold.</p>
      <p class="font-style">This text is italic.</p>
      <p class="text-decoration">This text is underlined.</p>
      <p class="text-transform">This text is transformed to uppercase.</p>
      <p class="text-align">This text is centered.</p>
</body>
</html>
```

OUTPUT:



AIM:

5)d.Write a program, to explain the importance of CSS Box model using

i. Content ii. Border iii. Margin iv. padding

DESCRIPTION:

The **CSS Box Model** is the fundamental concept that describes how every element on a webpage is structured and how spacing is applied. Each element is treated as a rectangular box made up of four layers:

1. **Content**
    - This is the innermost area of the box, where the actual text, image, or other media is displayed.
    - Example: The words inside a paragraph or an image inside a div are part of the content.
2. **Padding**
    - Padding is the space between the **content** and the **border** of the element.
    - It creates breathing room inside the box without affecting the border.
    - Example: Adding padding: 20px; increases the distance between the text and its surrounding border.
3. **Border**
    - The border wraps around the **padding** and the **content**.
    - It can be styled with color, thickness, and patterns (solid, dotted, dashed, etc.).
    - Example: border: 2px solid black; adds a black outline around the element.
4. **Margin**
    - The margin is the outermost layer of the box.
    - It creates space **outside** the border, separating the element from other elements.
    - Example: margin: 15px; pushes the element away from neighboring boxes.

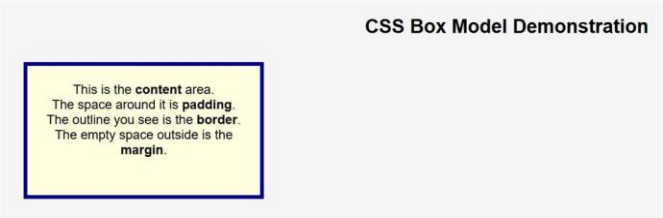
PROGRAM:

```
<!DOCTYPE html>

,html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>CSS Box Model Example</title>
  <style>
        .box {
          width: 250px;
          height: 120px;
          background-color: lightyellow;        /* i. Content */
          padding: 20px;                        /* iv. Padding */
          border: 5px solid darkblue;           /* ii. Border */
          margin: 30px;                         /* iii.Margin */
          font-size: 16px;
          text-align: center;
        }
body {
                background-color: #f5f5f5;
                font-family: Arial, sans-serif;
        }
    </style>
  </head>

<body>
  <h2 style="text-align:center;">CSS Box Model Demonstration</h2>
 <div class="box">
        This is the <b>content</b> area.<br>
        The space around it is <b>padding</b>.</br>
        The outline you see is the <b>border</b>.<br>
```

```
        The empty space outside is the <b>margin</b>.
    </div>
</body>
</html>
```

OUTPUT:



RESULT:

Experiment-6.1

AIM: Write a program to embed internal and external JavaScript in a web page

DESCRIPTION:

Internal JavaScript:

Written inside the HTML file using <script> tag.
Usually placed inside <head> or <body>.
Useful for small scripts.

External JavaScript

Saved in a separate .js file and linked using <script src="...">.
Useful for reusability and better code organization.

JavaScript I/O

Input: Using prompt() to take user input.
Output: Using document.write(), alert(), or console.log().

Type Conversion

parseInt() → converts string to integer.
parseFloat() → converts string to floating point.
Number() → converts string to number.
toString() → converts number to string.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Embedded</title>
  <!--External Javascript-->
  <script src="script.js"></script>
</head>
<body>
  <h3>24B11CS039</h3>
  <h1>Welcome to Javascript Demo</h1>
  <button onclick="internalFunction()">Run Internal JS</button>
  <button onclick="externalFunction()">Run External JS</button>
  <!--Internal Javascript-->
  <script>
    function internalFunction(){
      alert("Hello from internal javascript");
    }
  </script>
</body>
</html>
```

OUTPUT

24B11CS039

Welcome to Javascript Demo

Run Internal JS | Run External JS

RESULT:

Experiment-6.2

AIM: Write a program to explain the different ways for displaying output.

DESCRIPTION:

JavaScript provides multiple ways to display output on a webpage or to the user:

1. Using document.write()

This method writes content directly into the HTML document.

It is often used for testing or quick demonstrations.

2. Using console.log()

This method displays output in the browser's console (accessible by pressing F12 or opening developer tools).

It is mainly used by developers for debugging and testing code.

3. Using alert()

This method displays a popup dialog box with a message.

Showing urgent messages, alerts, or confirmations that require immediate user attention.

4. Using prompt()

prompt(): Displays a dialog box that asks the user for input. Whatever the user types can then be displayed or stored in a variable.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <h3>24B11CS039</h3>
  <h1>JavaScript Output Methods</h1>
  <button onclick="showAlert()">Show Alert</button>
  <button onclick="logToConsole()">Log to Console</button>
  <button onclick="writeToDocument()">write To Document</button>
  <button onclick="updateDOM()">Update DOM</button>
  <div id="outputArea">output will appear here...</div>
  <script>
    //1.Alert box
    function showAlert()
    {
      alert("This is an alert box!");
    }
    //2.console.log
    function logToConsole(){
      console.log(" This message is logged to the console.");
    }
    //3.document.write
    function writeToDocument(){
     document.write("This text is written directly to the document.");
    }
```

```
      //4.upadate dom
      function updateDOM(){
         document.getElementById("outputArea").innerText="This text was inserted using
DOM manipulation.";
      }
   </script>
</body>
</html>
```

OUTPUT:

RESULT:

Experiment-6.3

AIM: Write a program to explain the different ways for taking input.

DESCRIPTION:

JavaScript allows us to collect input from users in multiple ways. These methods can be simple (like dialog boxes) or more advanced (like HTML forms). Each method has its own purpose and usage.

1. Using prompt()

The simplest way to take input directly from a user.

When called, it opens a dialog box with a text field where the user can type something.

2. Using HTML Form Fields

The most common and professional way to collect user input.

Inputs can be taken through <input>, <textarea>, <select>, and other form elements.

3. Using Event Listeners (Keyboard/Mouse Events)

JavaScript can capture user input through events like keypress, keyup, keydown, or mouse click.

4. Using Command-Line Arguments (in Node.js)

If JavaScript is executed outside the browser (using Node.js), input can be taken from the command line.

5. Using External Files :

JavaScript can also take input indirectly by reading data from:

Text files / JSON files

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <style>
      body{
         font-family: Arial, Helvetica, sans-serif; padding: 20px;
         input,button{
            margin:10px 0;display: block;
         }
         #output{
            margin-top: 20px; font-weight: bold;
         }
      }
   </style>
</head>
<body>
  <h3>24B11CS039</h3>
  <h1>Ways to take input in Javascript</h1>
  <!--1.Text input field-->
  <label>Enter your name:</label>
  <input type="text" id="nameInput">
  <button onclick="getInputFromField()">Submit Name</button>
  <!--2.Prompt box-->
```

```
 <button onclick="getInputFromPrompt()">Enter Age (Prompt)</button>
 <!--3.Form Input-->
 <form onsubmit="getInputFromForm(event)">
  <label>Enter your Email</label>
  <input type="email" id="emailInput">
  <button type="submit">Submit Email</button>
 </form>
 <!---4.Dropdown Selection-->
  <label>Select your country</label>
  <select id="countrylist">
   <option value="India">India</option>
   <option value="USA">USA</option>
   <option value="Japan">Japan</option>
  </select>
  <button onclick="getInputFromDropdown()">Submit Country</button>
  <div id="output">Your input will appear here.</div>
  <script>
   //1.Input from text field
   function getInputFromField()
   {
      const name=document.getElementById("nameInput").value;
      document.getElementById("output").innerText="Name:"+name;
   }
   //2.Input from Prompt box
   function getInputFromPrompt(){
      const age=prompt("Please enter the age:");
      document.getElementById("output").innerText="Age:"+age;
   }
   //3.Input from form
   function getInputFromForm(event){
      event.preventDefault();//prevent page reload
      const email=document.getElementById("emailInput").value;
      document.getElementById("output").innerText="Email:"+email;
   }
   //4.Input from drop down selection
   function getInputFromDropdown(){
     const country=document.getElementById("countrylist").value;
     document.getElementById("output").innerText="Country:"+country;
   }
  </script>
</body>
</html>
```

OUTPUT:

RESULT:

Experiment-6.4

AIM: Create a webpage which uses prompt dialogue box to ask a voter for his name and age. Display the information in table format along with either the voter can vote or not.

DESCRIPTION:

It validates the inputs (not empty, age is a sensible number).
It calculates eligibility (e.g., "Can Vote" if age ≥ 18; otherwise "Cannot Vote").
It displays the results in a table on the page with the columns: *Name*, *Age*, *Eligibility*, and optionally *Notes/Reason*.
If the user cancels or provides invalid input, the page shows a friendly message and does not add a bad row.
Input collection details

Name prompt

The prompt message clearly asks for the voter's full name.
After the user responds, the script trims whitespace from the start and end.

Age prompt

The prompt message asks for Age in years (numbers only).
After the user responds, the script trims whitespace and attempts to convert the string to a number.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Voter Eligibility checker</title>
   <style>
     body{
       font-family: Arial,sans-serif;
       padding: 20px;
     }
     table{
       border-collapse: collapse;
       width:50%;
       margin-top: 20px;
     }
     th,td{
       border: 1px solid #333;
       padding: 20px;
       text-align: center;
     }
th{
background-color:softblue
}
   </style>
</head>
<body>
   <h3>24B11CS039</h3>
```

```
<h1> Voter Eligibility checker</h1>
<button onclick="checkvoter()">Check Eligibility </button>
<div id="result"></div>
<script>
   function checkvoter()
   {
      const name=prompt("Enter Name");
      const age=prompt("Enter Age");
      if(name===null || age===null || name.trim()==="" || isNaN(age)){
         alert("Invalid. Try again Later");
         return;
      }

      const ageNum=parseInt(age);
      const eligibility = ageNum >=18 ? "Eligibility to Vote" : "Not Eligible to Vote";

      const tableHTML =`
      <table>
         <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Eligibility</th>
         </tr>
         <tr>
            <td>${name}</td>
            <td>${ageNum}</td>
            <td>${eligibility}</td>
         </tr>
       </table>
    `;
     document.getElementById("result").innerHTML=tableHTML;
    }
  </script>
</body>
</html>
```

OUTPUT:

24B11CS039

Voter Eligibility checker

Check Eligibility

Name	Age	Eligibility
deevena	19	Eligibility to Vote

RESULT:

Experiment-7.1

AIM: write a program using document object properties and methods.
DESCRIPTION: The Document Object Model (DOM) represents the structure of an HTML document.
Using the document object, we can access and manipulate elements, attributes, and content of a web page dynamically.
Common properties of the document object include title, URL, and lastModified.
Common methods include write(), getElementById(), getElementsByTagName(), and createElement().
This program demonstrates how to use document object properties to display webpage information and methods to modify the page dynamically.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document Object Example</title>
</head>
<body>
  <h3>24B11CS039</h3>
  <h1 id="heading">Hello World!</h1>
  <p>This is a demo of document object.</p>

  <button onclick="showInfo()">Show Document Info</button>

  <div id="output"></div>

  <script>
    function showInfo() {
      // Using document properties
      let title = document.title;
      let URL = document.URL;
      let lastModified = document.lastModified;

      // Using document methods
      document.getElementById("heading").style.color = "brown"; // Change heading color

      document.getElementById("output").innerHTML =
        "Title: " + title + "<br>" +
        "URL: " + URL + "<br>" +
        "Last Modified: " + lastModified;
    }
  </script>
</body>
</html>
```

OUTPUT:

24B11CS039

Hello World!

This is a demo of document object.

Show Document Info

Title: Document Object Example
URL: file:///C:/Users/DEEVENA/2.a.html
Last Modified: 10/12/2025 06:48:07

RESULT:

Experiment-7.2

AIM: write a program using window object properties and methods

DESCRIPTION: In JavaScript, the window object represents the browser window.
It is the global object in client-side JavaScript — meaning all global variables, functions, and objects belong to it.

Common Window Object Properties:

- window.innerWidth – Returns the width of the browser window's content area.
- window.innerHeight – Returns the height of the browser window's content area.
- window.location – Returns the current URL.
- window.document – Refers to the document loaded in the window.

Common Window Object Methods:

- window.alert() – Displays an alert dialog box.
- window.confirm() – Displays a dialog box with OK and Cancel buttons.
- window.prompt() – Displays a dialog box that prompts the user for input.
- window.open() – Opens a new browser window.
- window.setTimeout() – Executes a function after a specified delay.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Window Object Example</title>
</head>
<body>
 <h3>24B11CS039</h3>
 <h1>Window Object Demo</h1>

 <button onclick="showWindowInfo()">Show Window Info</button>
 <button onclick="openNewWindow()">Open New Window</button>
 <button onclick="closeNewWindow()">Close New Window</button>

 <div id="output"></div>

 <script>
  let newWin; // to store reference of opened window

  function showWindowInfo() {
   // Using window properties
   let width = window.innerWidth;
   let height = window.innerHeight;
   let locationHref = window.location.href;

   // Display info
   document.getElementById("output").innerHTML =
    "Window Width: " + width + "px<br>" +
    "Window Height: " + height + "px<br>" +
    "Current URL: " + locationHref;
  }
```

```
  function openNewWindow() {
    // Using window.open()
    newWin = window.open("https://www.google.com", "Example", "width=400,height=300");
  }

  function closeNewWindow() {
    // Using window.close()
    if (newWin) {
      newWin.close();
    }
  }
 </script>
</body>
</html>
```

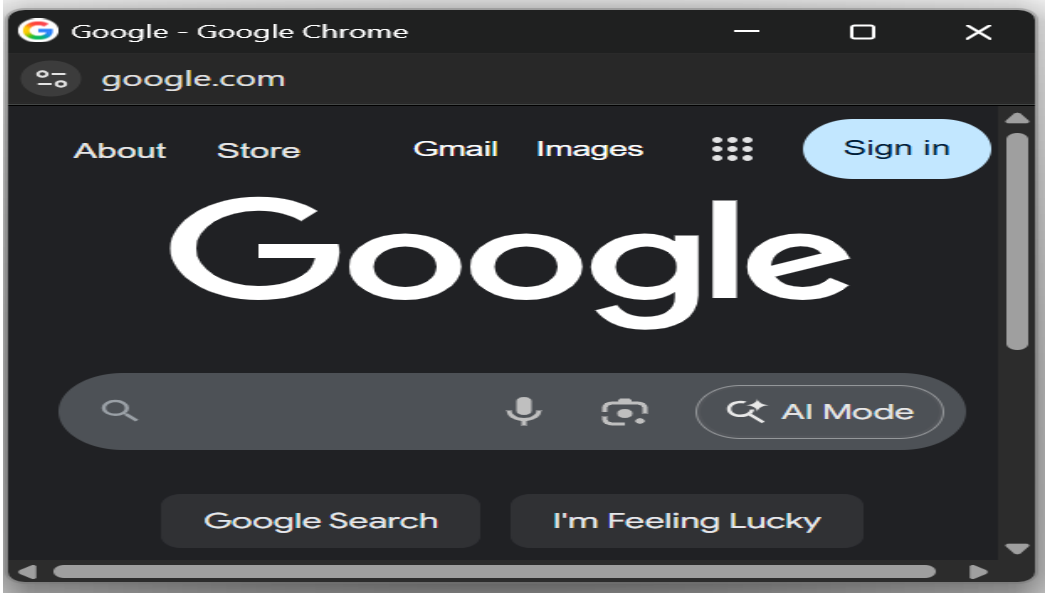
OUTPUT:

24B11CS039

Window Object Demo

Show Window Info | Open New Window | Close New Window

Window Width: 1280px
Window Height: 665px
Current URL: file:///C:/Users/DEEVENA/2.a.html



RESULT:

Experiment-7.3

AIM: write a program using array object properties and methods

DESCRIPTION: In JavaScript, the Array object is used to store multiple values in a single variable.

It provides several properties and methods to work with data easily.

Common Array Properties:

- length – Returns the number of elements in an array.

Common Array Methods:

- push() – Adds an element at the end.
- pop() – Removes the last element.
- unshift() – Adds an element at the beginning.
- shift() – Removes the first element.
- sort() – Sorts the array.
- reverse() – Reverses the order of elements.
- join() – Combines elements into a string.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Array object example</title>
</head>
<body>
  <h3>24B11CS039</h3>
  <h1>Array object Example</h1>
  <p id="output"></p>
  <script>
    let fruits=["banana","mango","grapes"];
    let length=fruits.length;
    fruits.push("orange");      //add an element at the end
    fruits.unshift("Papaya");  //adds an element at the beggining
    let last=fruits.pop();     //removes the last element
    let first=fruits.shift();  //removes the first element
    let sorted=fruits.sort();  //sort the array
    let joined=fruits.join(", ");//joins elements as a string


    document.getElementById("output").innerHTML=
   "<b>Original length:</b>"+length+"<br>"+
   "<b>After push & unshift:</b> papaya,banana,mango,grapes,orange<br>"+
   "<b>Removed Last:</b>"+last+"<br>"+
   "<b>Removed first:</b>"+first+"<br>"+
   "<b>Sorted array:</b>"+sorted+"<br>"+
   "<b>Joined as string:</b>"+joined;
  </script>
```

```
</body>
</html>
```

OUTPUT:

24B11CS039

Array Object Example

Original length: 3
After push & unshift: Papaya, banana, mango, grapes, orange
Removed Last: orange
Removed First: Papaya
Sorted Array: banana,grapes,mango
Joined as String: banana, grapes, mango

RESULT

Experiment-7.4

AIM: write a program using math object properties and methods

DESCRIPTION: The Math object in JavaScript provides mathematical constants and functions.

It is not a constructor, so you don't need to create it — you can use its methods directly as Math.methodName().

Common Math Properties:

- Math.PI – Returns the value of π (3.14159...).
- Math.E – Returns Euler's number (2.718...).

Common Math Methods:

- Math.round() – Rounds to the nearest integer.
- Math.floor() – Rounds down to the nearest integer.
- Math.ceil() – Rounds up to the nearest integer.
- Math.sqrt() – Returns the square root.
- Math.pow(x, y) – Returns x raised to the power of y.
- Math.random() – Returns a random number between 0 and 1.
- Math.max() / Math.min() – Returns the largest/smallest number.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Math Object Example</title>
</head>
<body>
 <h3>24B11CS039</h3>
 <h2>Math Object Example</h2>

 <p id="output"></p>

 <script>
  // Math Object Properties
  let piValue = Math.PI;
  let eValue = Math.E;
  let sqrt2Value = Math.SQRT2;

  // Math Object Methods
  let sqrtResult = Math.sqrt(16);
  let powerResult = Math.pow(2, 5);
  let absResult = Math.abs(-25);
  let roundResult = Math.round(4.7);
  let ceilResult = Math.ceil(4.2);
  let floorResult = Math.floor(4.8);
  let randomResult = Math.random();
  let maxResult = Math.max(3, 9, 2, 15, 7);
  let minResult = Math.min(3, 9, 2, 15, 7);

  // Display results
  document.getElementById("output").innerHTML =
   "<b>Properties:</b><br>" +
   "PI: " + piValue + "<br>" +
```

```
    "E: " + eValue + "<br>" +
    "SQRT2: " + sqrt2Value + "<br><br>" +

    "<b>Methods:</b><br>" +
    "Square Root of 16: " + sqrtResult + "<br>" +
    "2^5 (Power): " + powerResult + "<br>" +
    "Absolute of -25: " + absResult + "<br>" +
    "Round(4.7): " + roundResult + "<br>" +
    "Ceil(4.2): " + ceilResult + "<br>" +
    "Floor(4.8): " + floorResult + "<br>" +
    "Random Number (0–1): " + randomResult + "<br>" +
    "Max(3, 9, 2, 15, 7): " + maxResult + "<br>" +
    "Min(3, 9, 2, 15, 7): " + minResult;
  </script>
</body>
</html>
```

OUTPUT:

24B11CS039

Math Object Example

Properties:
PI: 3.141592653589793
E: 2.718281828459045
SQRT2: 1.4142135623730951

Methods:
Square Root of 16: 4
2^5 (Power): 32
Absolute of -25: 25
Round(4.7): 5
Ceil(4.2): 5
Floor(4.8): 4
Random Number (0–1): 0.43620672712826647
Max(3, 9, 2, 15, 7): 15
Min(3, 9, 2, 15, 7): 2

RESULT:

Experiment-7.5

AIM: write a program using string object properties and methods

DESCRIPTION: The String object in JavaScript is used to work with text.
It provides properties and methods to perform operations such as finding length, changing case, extracting substrings, searching, and replacing text.

Common String Property:

- length – Returns the number of characters in a string.

Common String Methods:

- toUpperCase() – Converts to uppercase.
- toLowerCase() – Converts to lowercase.
- charAt(index) – Returns the character at the given position.
- indexOf(substring) – Returns the position of the first occurrence of a substring.
- substring(start, end) – Extracts characters between two indices.
- replace(old, new) – Replaces part of a string.
- concat() – Joins two or more strings.
- trim() – Removes extra spaces.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>String Object Example</title>
</head>
<body>
 <h3>24B11CS039</h3>
 <h2>JavaScript String Object Example</h2>

 <p id="output"></p>

 <script>
  // Create a String
  let text = "Hello JavaScript World!";

  // Using String Property
  let length = text.length; // property

  // Using String Methods
  let upper = text.toUpperCase();         // Convert to uppercase
  let lower = text.toLowerCase();         // Convert to lowercase
  let trimmed = text.trim();              // Remove extra spaces
  let sliced = text.slice(6, 16);         // Extract part of string
  let replaced = text.replace("World", "Programming"); // Replace word
  let charAtPos = text.charAt(7);         // Character at position 7
  let index = text.indexOf("JavaScript");  // Find index of word
  let splitted = text.split(" ");         // Split into array by spaces

  // Display the results
  document.getElementById("output").innerHTML =
   "<b>Original String:</b> " + text + "<br>" +
   "<b>Length:</b> " + length + "<br>" +
```

```
    "<b>Uppercase:</b> " + upper + "<br>" +
    "<b>Lowercase:</b> " + lower + "<br>" +
    "<b>Trimmed:</b> " + trimmed + "<br>" +
    "<b>Sliced (6–16):</b> " + sliced + "<br>" +
    "<b>Replaced:</b> " + replaced + "<br>" +
    "<b>Character at 7:</b> " + charAtPos + "<br>" +
    "<b>Index of 'JavaScript':</b> " + index + "<br>" +
    "<b>Splitted (Array):</b> " + splitted.join(", ");
  </script>
</body>
</html>
```

OUTPUT:

24B11CS039

JavaScript String Object Example

Original String: Hello JavaScript World!
Length: 23
Uppercase: HELLO JAVASCRIPT WORLD!
Lowercase: hello javascript world!
Trimmed: Hello JavaScript World!
Sliced (6–16): JavaScript
Replaced: Hello JavaScript Programming!
Character at 7: a
Index of 'JavaScript': 6
Splitted (Array): Hello, JavaScript, World!

RESULT:

Experiment-7.6

AIM: write a program using regex object properties and methods

DESCRIPTION: The RegExp (Regular Expression) object in JavaScript is used to search for patterns in strings.

It provides properties and methods for pattern matching and text validation.

Common RegExp Properties:

- source – Returns the text of the regular expression.
- global – Returns true if the global (g) flag is set.
- ignoreCase – Returns true if the case-insensitive (i) flag is set.
- multiline – Returns true if the multiline (m) flag is set.

Common RegExp Methods:

- test() – Tests if a pattern exists in a string (returns true or false).
- exec() – Executes a search and returns the first match.
- Used with String methods:
  - match() – Returns an array of matches.
  - replace() – Replaces matched text.
  - search() – Returns position of match.

PROGRAM:

```
<html>
<head>
	<title>Contact us Page</title>
	<script type="text/javascript">
		//Validation for fields
		function validation(){
			var nm=document.getElementById("name_id").value;
			var em=document.getElementById("email_id").value;
			var su=document.getElementById("subject_id").value;
			var me=document.getElementById("message_id").value;
			if(nm=="" && em=="" && su=="" && me=="")
			{
				alert("Please enter all fields");
				return false;
			}
			else if(nm=="")
			{
				alert("Enter Name");
				return false;
			}
			else if(nm.length<4)
			{
				alert("Name should not be below 6 character");
				return false;
			}
			else if(em=="")
			{
				alert("Enter Email");
```

```
                return false;
            }
            else  if(!/^[A-z0-9]{6,}[@][A-z]{4,5}[.][A-z]{3}$/.test(em))
            {
                alert("Enter valid Email ID");
                return false;
            }
            else if(su=="")
            {
                alert("Enter Subject");
                return false;
            }
            else if(me=="")
            {
                alert("Enter message");
                return false;
            }

            else if(ms.length<=20)
            {
                alert("message should below 20 characters ");
                return false;
            }
            else{
                return true;
            }
        }


    </script>
</head>
<h3>24B11CS005</h3>
<form onSubmit="return validation()" action="registration_app.php" method="POST">
    <table>
        <tr>
            <td colspan="2"><h2>Contact us</h2></td>
        </tr>
        <tr>
            <td>Name</td>
            <td><input type="text" name="name" id="name"/></td>
        </tr>
        <tr>
            <td>Email ID</td>
            <td><input type="text" name="email" id="email_id" /></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="text" name="password" id="password_id" /></td>
```

```
		</tr>
		<tr>
			<td>Phone number</td>
			<td><input type="text" name="phone" id="phone" /></td>
		</tr>

		<tr>
			<td>Address</td>
			<td><textarea name="address" id="address"></textarea></td>
		</tr>
		<tr>
			<td></td>
			<td><input type="submit" value="SUBMIT" /><input type="reset"
value="CLEAR" /></td>
		</tr>
	</table>
</form>
</html>
```

OUTPUT:

24B11CS039

Contact Us

Name: deeevena

Email ID: thee@gmail.com

Password:

Phone Number: 9652443578

Address: kakinada

SUBMIT CLEAR

RESULT:

Experiment-7.7

AIM: write a program using date object properties and methods

DESCRIPTION: The Date object in JavaScript is used to work with dates and times.
It can display the current date, extract date components (like day, month, year), and even modify date values.

Common Date Methods:

- getDate() – Returns the day of the month (1–31).
- getMonth() – Returns the month (0–11).
- getFullYear() – Returns the year.
- getDay() – Returns the day of the week (0–6).
- getHours(), getMinutes(), getSeconds() – Return time components.
- setFullYear(), setMonth(), setDate() – Set specific parts of a date.
- toDateString() – Returns a readable date string.
- toLocaleString() – Returns date and time based on locale.

PROGRAM:

```
<!DOCTYPE html>
<head>
   <title>Date Object Example</title>
</head>
<body>
  <h3>24B11CS039</h3>
  <h2>JavaScript Date Object Example</h2>
  <p id="dateInfo"></p>
   <script>

     //Create a new date object
      let currentDate=new Date();

    //Using date object methods
      let year=currentDate.getFullYear();
      let month=currentDate.getMonth()+1;        //get the month (0-11), so +1
      let date=currentDate.getDate();
      let day=currentDate.getDay();
      let hours=currentDate.getHours();
      let  minutes=currentDate.getMinutes();
      let  seconds=currentDate.getSeconds();
      let time=currentDate.getTime();

    //Array for week days
      let days=["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"]

    //Display the values
     document.getElementById("dateInfo").innerHTML=
     "Current Date:" + date + "/" + month + "/" + year + "<br>" +
     "Day of the week:" + days[day] + "<br>" +
```

```
    "Current Time:" + hours + ":" + minutes + ":" + seconds + "<br>" +
    "Milliseconds since jan 1, 1970: " + time;
   </script>
  </body>
</html>
```

OUTPUT:

24B11CS039

JavaScript Date Object Example

Current Date: 12/10/2025
Day of the Week: Sunday
Current Time: 7:6:34
Milliseconds since Jan 1, 1970: 1760232994438

RESULT:

Experiment-7.8

AIM: write a program to explain user defined object properties,methods,accessors,construcutors and display

DESCRIPTION: In JavaScript, you can create user-defined objects using a constructor function or object literals.

Objects can have:

- Properties – variables that hold data.
- Methods – functions that define behavior.
- Accessors – get and set methods for controlled access to properties.
- Constructors – special functions used to initialize objects.

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
   <title>User Defined Object Example</title>
</head>
<body>
  <h3>24B11CS039</h3>
 <h2>User Defined Object Example</h2>
 <p id="Output"></p>

  <script>
    //constructor function to define a user-defined object
    function Student(name,age,marks)  {

      //Properties
       this.name=name;
       this.age=age;
       this.marks=marks;
     //Method

      this.displayDetails=function()   {
     return "Name:"+this.name+", Age:"+this.age+", Marks:"+this.marks;
      };

     //Accessor(Getter)
      this.getGrade=function() {
      if(this.marks>=90) return  "A";
      else if(this.marks>=75) return "B";
      else if(this.marks>=50) return "C";
      else return "fail";
     };
    //Accessor(setter)
     this.setMarks=function(newMarks) {
     this.marks=newMarks;
    };
```

```
  }

   //Create an object using the constructor
    let student1=new Student("Hema",18,82);

   //Use setter to update marks
    student1.setMarks(92);

  //Display details and grade
   document.getElementById("Output").innerHTML=
   student1.displayDetails()+"<br>"+
   "Grade:"+student1.getGrade();
 </script>
</body>
</html>
```

OUTPUT:

24B11CS039

User Defined Object Example

Name: Hema, Age: 18, Marks: 92
Grade: A

RESULT:

Experiment-8.1

AIM: Write a program which asks the user to enter three integers, obtains the numbers from the user and outputs HTML text that displays the larger number followed by the words "LARGER NUMBER" in an information message dialog. If the numbers are equal, output HTML text as "EQUAL NUMBERS".

DESCRIPTION: This program is designed using HTML and JavaScript to compare three integers entered by the user. When the user clicks a button, they are prompted (using prompt()) to input three numbers. These inputs are then converted to integers using parseInt().

The program first checks if all three numbers are equal. If they are, it displays the message "EQUAL NUMBERS" using both an alert() message dialog and by writing it to the HTML page using innerHTML.

If the numbers are not all equal, the program calculates the largest of the three using simple conditional comparisons (if statements). It then displays the largest number followed by the message "LARGER NUMBER" (e.g., "25 LARGER NUMBER") in both the alert dialog and on the web page.

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
   <title>Larger Number Finder</title>
  <script>
function findLargerNumber () {
    //Take three numbers from user
     let num1=parseInt(prompt("Enter first integer:"));
     let num2=parseInt(prompt("Enter second integer:"));
     let num3=parseInt(prompt("Enter third integer:"));

     let message="";

    //Check if all numbers are equal
    if (num1===num2 && num2===num3) {
    message="EQUAL NUMBERS";
    } else {
    //find the largest
    let Largest=Math.max(num1,num2,num3);
    message=Largest+" is the LARGER NUMBER";
    }
    //Display result in an info dialog
    alert(message);
    }
  </script>
</head>
<body onload="findLargerNumber()">
</body>
</html>
```

OUTPUT:

RESULT:

Experiment-8.2

AIM: Write a program to display week days using switch case**.**

DESCRIPTION: This program is developed using HTML and JavaScript to display the name of a weekday based on a number input from the user. The user is prompted to enter a number from 1 to 7, where each number represents a day of the week:

- 1 → Monday
- 2 → Tuesday
- ...
- 7 → Sunday

The switch-case statement is used to evaluate the input and match it to the correct day. If the input does not fall within the 1 to 7 range, the program displays an error message: "Invalid input! Please enter a number between 1 and 7."

This exercise demonstrates:

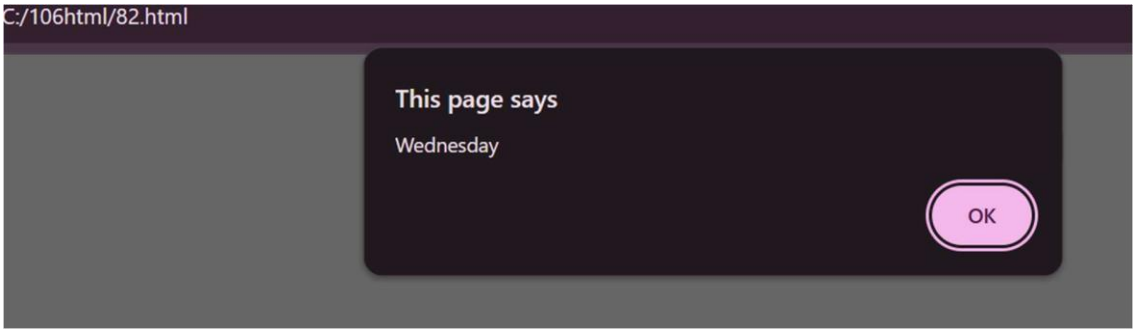
- Use of switch-case for multi-branch selection.
- Simple user interaction using prompt().
- Output using alert() and HTML display (innerHTML).

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
  <title>Week Days with switch case</title>
 <script>

function showWeekDay() {
    let daynumber=parseInt(prompt("Enter a number(1-7) for the weekday:"));
    let dayName="";

    switch(daynumber) {
    case 1:
        dayName="Monday";
         break;
    case 2:
        dayName="Tuesday";
         break;
    case 3:
        dayName="Wednesday";
         break;
    case 4:
        dayName="Thursday";
         break;
    case 5:
        dayName="Friday";
         break;
    case 6:
        dayName="Saturday";
         break;
    case 7:
        dayName="Sunday";
```

```
            break;
      default:
            dayName="Invalid";
       }

     //Display result in dialog box
       alert(dayName);
      // Also display result inside webpage
        document.getElementById("output").innerHTML =
            "<h2>" + dayName + "</h2>";
            }
</script>
<body onload ="showWeekDay()">
<div id="output" style="text-align:center;margin-top:20px;font-family:Lucida
Calligraphy;"></div>
</body>
</html>
```

OUTPUT:



RESULT:

EXPERIMENT-8.3

AIM:

Write a program to print 1 to 10 numbers using for, while and do-while loops.

DESCRIPTION:

This program demonstrates how to print numbers from 1 to 10 using three types of JavaScript loops — for, while, and do-while. Each button calls a function that runs the respective loop and displays the numbers in the Output section.
For Loop: Repeats from 1 to 10 using a counter inside the loop header.
While Loop: Runs while the condition (i <= 10) is true.
Do-While Loop: Executes at least once before checking the condition.
Example Output: For Loop: 1 2 3 4 5 6 7 8 9 10

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<title>Loop Demo</title>
</head>
<body>
<h1>24B11CS039</h1>
<h2>Print 1 to 10 Using Loops</h2>
<button onclick="printUsingFor()">Print Using For Loop</button>
<button onclick="printUsingWhile()">Print Using While Loop</button>
<button onclick="printUsingDoWhile()">Print Using Do-While Loop</button>
<h3>Output:</h3>
<p id="output"></p>
<script>
 function printUsingFor() {
  let result = "For Loop: ";
  for (let i = 1; i <= 10; i++) {
  result += i + " ";
  }
  document.getElementById("output").innerHTML = result;
 }
 function printUsingWhile() {
  let i = 1;
  let result = "While Loop: ";
  while (i <= 10) {
   result += i + " ";
   i++;
  }
  document.getElementById("output").innerHTML = result;
 }
 function printUsingDoWhile() {
  let i = 1;
  let result = "Do-While Loop: ";
  do {
```

```
    result += i + " ";
    i++;
   } while (i <= 10);
   document.getElementById("output").innerHTML = result;
  }
</script>
</body>
</html>
```

OUTPUT:

24B11CS039

Print 1 to 10 Using Loops

Print Using For Loop | Print Using While Loop | Print Using Do-While Loop

Output:

Do-While Loop: 1 2 3 4 5 6 7 8 9 10

RESULT:

EXPERIMENT-8.4

AIM:

Write a program to print data in object using for-in, for-each and for-of loops

DESCRIPTION:

This program demonstrates how to iterate through an object in JavaScript using three different loop methods — for-in, forEach, and for-of.

When the webpage loads, it creates a person object with properties: name, age, and city.

The program then displays each property and its value using all three loop types.

for-in loop: Iterates over the keys of the object.

forEach loop: Uses Object.keys() to loop through each key.

for-of loop: Uses Object.entries() to get key-value pairs.

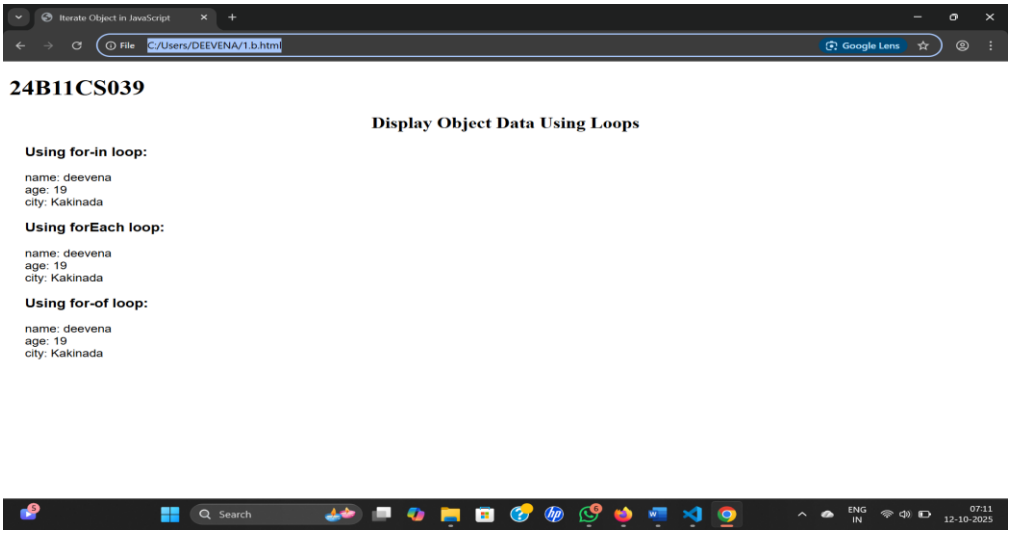
Output: It shows the property names and values three times — once for each loop method.

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Iterate Object in JavaScript</title>
 <script>

  function displayObjectData() {
   const person = {
    name: "deevena",
    age: 19,
    city: "Kakinada"
   };

   let output = "";

   // Using for-in (loops over keys)
   output += "<h3>Using for-in loop:</h3>";
   for (let key in person) {
    output += key + ": " + person[key] + "<br>";
   }

   // Using forEach (on Object.keys)
   output += "<h3>Using forEach loop:</h3>";
   Object.keys(person).forEach(function(key) {
   output += key + ": " + person[key] + "<br>";
   });

   // Using for-of (on Object.entries)
   output += "<h3>Using for-of loop:</h3>";
   for (let [key, value] of Object.entries(person)) {
    output += key + ": " + value + "<br>";
   }
```

```
    document.getElementById("result").innerHTML = output;
   }
  </script>
</head>
<body onload="displayObjectData()">
<h1>24B11CS039</h1>
  <h2 style="text-align:center;">Display Object Data Using Loops</h2>
  <div id="result" style="margin:20px; font-family:Arial; font-size:16px;">
  </div>
</body>
</html>
```

OUTPUT:



RESULT: