WEB DEVELOPMENT LAB

LAB MANUAL

Course Code: PE/CSE/24-P

B.Tech CSE — 8th Sem

Department	Computer Science & Engineering
Course Type	Professional Core Lab
Duration	2 Hours / Week
Assessment	Internal 50 | External 50

Pre-requisite: Basic programming skills and knowledge of surfing internet.

Course Overview

This lab course on web development involves learning web-based programming languages. It incorporates the development of web pages by structuring information provided for the website design. The objective of the lab course is to equip students to design web pages using modern web development tools.

Course Outcomes

By the end of the course, students will be able to:

- CO1. Implement object models for website design using modern tools like HTML, XML, and JavaScript etc. (LOTS: Level 3: Apply)
- CO2. Analyze the design of websites. (LOTS: Level 4: Analyze)
- CO3. Test the design of websites. (LOTS: Level 5: Evaluate)
- CO4. Design websites that consider socio-cultural values. (LOTS: Level 6: Create)
- CO5. Create a written report for website designed. (LOTS: Level 6: Create)

- CO6. Use ethical practices and socio-cultural values while designing websites. (LOTS: Level 3: Apply)

List of Experiments

1. Create a Simple Webpage Using HTML
2. Designing a Registration Form with Table and Hyperlinks
3. Designing a Page with Frames to Include Images and Videos
4. Adding a Cascading Style Sheet for Designing the Web Page
5. User Defined Function to Sort Array Values in Ascending Order
6. Dynamic Web Page with Form Validation Using JavaScript
7. Design a Catalogue in ASP
8. Event Handling Validation of Registration Form
9. Open a Window from Current Window on Mouse Over Event
10. Simple Application to Demonstrate Servlet Request and Response Object
11. Demonstrate Array Objects and Date Object's Predefined Methods
12. Display Calendar for Month and Year Selected from Combo Box
13. Create a Welcome Cookie and Display Different Image and Text
14. Demonstrate Request and Response Object Using HTML Form
15. Database Connection to Display All Values in the Table

Program **1**	**Write a program to create a simple webpage using basic HTML structure and tags.**

THEORY

HTML (HyperText Markup Language) is the standard markup language used to create web pages. Every HTML document starts with a doctype declaration followed by the <html> root element. The document is divided into two sections: <head> (metadata) and <body> (visible content). Basic HTML tags include headings (<h1>-<h6>), paragraphs (<p>), line breaks (
), horizontal rules (<hr>), and structural tags like and .

PROCEDURE

1. Open a text editor (Notepad, VS Code, etc.)
2. Write the basic HTML structure with <!DOCTYPE html>, <html>, <head>, and <body> tags
3. Add a <title> in the head section
4. Add headings, paragraphs, and other content in the body section
5. Save the file with .html extension
6. Open the file in a web browser to view the output

Source Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My First Webpage</title>
</head>
<body>
    <h1>Welcome to My First Webpage</h1>
    <h2>About Me</h2>
    <p>Hello! I am a BTech CSE student learning web development.</p>
    <hr>
    <h3>My Interests</h3>
```

```
    <p>I enjoy coding, problem solving, and designing websites.</p>
</body>
</html>
```

EXPECTED OUTPUT
A webpage displaying a heading 'Welcome to My First Webpage', sub-headings, paragraphs, and a horizontal rule is rendered in the browser.

Program 2	**Write a program to design a registration form using HTML tables and include hyperlinks.**

THEORY

HTML forms are used to collect user input. The <form> tag defines a form. Input elements like <input>, <select>, and <textarea> collect data. HTML tables (<table>, <tr>, <td>, <th>) provide a grid layout. Hyperlinks are created using the <a> tag with the href attribute, allowing navigation between pages or to external URLs. Form attributes like action and method control data submission.

PROCEDURE

1. Create an HTML file with a form tag
2. Use a table to align form labels and inputs
3. Add various input types: text, email, password, radio, checkbox
4. Include a submit button inside the form
5. Add hyperlinks to navigate between pages
6. Save and open in browser

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Registration Form</title></head>
<body>
<h2>Student Registration Form</h2>
<form action="#" method="post">
 <table border="1" cellpadding="10">
  <tr>
   <td><label>Name:</label></td>
   <td><input type="text" name="name" required></td>
  </tr>
  <tr>
   <td><label>Email:</label></td>
```

```
    <td><input type="email" name="email" required></td>
   </tr>
   <tr>
    <td><label>Password:</label></td>
    <td><input type="password" name="password"></td>
   </tr>
   <tr>
    <td><label>Gender:</label></td>
    <td>
     <input type="radio" name="gender" value="M"> Male
     <input type="radio" name="gender" value="F"> Female
    </td>
   </tr>
   <tr>
    <td colspan="2" align="center">
     <input type="submit" value="Register">
    </td>
   </tr>
  </table>
</form>
<br><a href="login.html">Already registered? Login here</a>
</body></html>
```

EXPECTED OUTPUT

A registration form displayed in a table format with text fields, radio buttons, and a submit button, along with a hyperlink to a login page.

Program 3	Write a program to design a webpage that embeds images and videos using HTML.

THEORY

HTML supports embedding multimedia content. Images are added using the <img> tag with src, alt, width, and height attributes. Videos are embedded using the <video> tag with controls, autoplay, and loop attributes, supporting MP4, WebM, and OGG formats. The <iframe> tag is used to embed external videos from platforms like YouTube. Responsive media uses CSS percentage widths to scale with the viewport.

PROCEDURE

1. Create an HTML file
2. Add images using the <img> tag with proper attributes
3. Embed a local video using the <video> tag
4. Embed an external video using <iframe>
5. Apply basic CSS to style the layout
6. Test in browser

Source Code

```
<!DOCTYPE html>
<html>
<head>
  <title>Media Page</title>
  <style>
    body { font-family: Arial; background: #f0f0f0; text-align: center; }
    img { width: 400px; height: 250px; border-radius: 8px; }
    video { width: 500px; margin: 20px; }
    iframe { width: 560px; height: 315px; border: none; }
  </style>
</head>
<body>
  <h2>Media Showcase</h2>
```

```
  <h3>Image</h3>
  <img src="landscape.jpg" alt="Beautiful Landscape">
  <h3>Local Video</h3>
  <video controls>
    <source src="sample.mp4" type="video/mp4">
    Your browser does not support HTML5 video.
  </video>
  <h3>YouTube Video</h3>
  <iframe src="https://www.youtube.com/embed/dQw4w9WgXcQ"
    allowfullscreen></iframe>
</body>
</html>
```

EXPECTED OUTPUT

A webpage displaying an image, a local video with controls, and an embedded YouTube video in an organized layout.

Program 4	**Write a program to apply CSS styles to enhance the visual design of a webpage.**

THEORY

CSS (Cascading Style Sheets) is used to control the presentation of HTML elements. There are three ways to apply CSS: inline (style attribute), internal (<style> tag in <head>), and external (.css file linked via <link>). CSS properties include color, background, font, margin, padding, border, and layout properties. The CSS Box Model describes how content, padding, border, and margin work together. Selectors target elements by tag name, class (.), or id (#).

PROCEDURE

1. Create an HTML file with various elements
2. Create an external CSS file (styles.css)
3. Link the CSS file using <link rel='stylesheet' href='styles.css'>
4. Apply styles: fonts, colors, borders, margins, padding
5. Add hover effects and pseudo-classes
6. Test responsiveness in browser

Source Code

```
/* styles.css */
* { box-sizing: border-box; margin: 0; padding: 0; }
body {
  font-family: 'Segoe UI', sans-serif;
  background-color: #f4f4f4;
  color: #333;
}
header {
  background-color: #1a3c6e;
  color: white;
  padding: 20px;
  text-align: center;
}
```

```
nav a {
  display: inline-block;
  padding: 10px 20px;
  color: white;
  background: #2980b9;
  text-decoration: none;
  margin: 5px;
  border-radius: 5px;
}
nav a:hover { background-color: #1abc9c; }
.container {
  max-width: 900px;
  margin: 30px auto;
  padding: 20px;
  background: white;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}
```

EXPECTED OUTPUT

A styled webpage with a colored header, styled navigation links with hover effects, and a centered content container with shadow.

Program 5	**Write a program to use a JavaScript user-defined function to sort an array of values in ascending order on a webpage.**

THEORY

JavaScript allows developers to define custom functions using the function keyword. Arrays in JavaScript have built-in methods like sort(), but for numeric sorting, a comparator function must be provided: arr.sort((a,b) => a-b). User-defined functions encapsulate logic for reuse. The DOM (Document Object Model) allows JavaScript to dynamically update HTML content using methods like document.getElementById() and innerHTML.

PROCEDURE

1. Create an HTML file with an input area and button
2. Write a JavaScript function to read values
3. Implement bubble sort or use Array.sort() with comparator
4. Display sorted values on the webpage
5. Test with different input arrays

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Array Sorting</title></head>
<body>
<h2>Sort Array in Ascending Order</h2>
<input type="text" id="inputArr" placeholder="Enter numbers separated by commas">
<br><br>
<button onclick="sortArray()">Sort</button>
<p id="output"></p>

<script>
  function sortArray() {
    let input = document.getElementById('inputArr').value;
    let arr = input.split(',').map(Number);
```

```
    // Bubble Sort
    for (let i = 0; i < arr.length - 1; i++) {
      for (let j = 0; j < arr.length - i - 1; j++) {
        if (arr[j] > arr[j + 1]) {
          let temp = arr[j];
          arr[j] = arr[j + 1];
          arr[j + 1] = temp;
        }
      }
    }
    document.getElementById('output').innerHTML =
      'Sorted Array: ' + arr.join(', ');
  }
</script>
</body></html>
```

EXPECTED OUTPUT

User enters '5,3,8,1,4' and clicks Sort. The output displays: 'Sorted Array: 1, 3, 4, 5, 8'.

Program 6	**Write a program to design a dynamic web page with client-side form validation using JavaScript.**

THEORY

Form validation ensures that user input meets required criteria before submission. JavaScript can intercept form submission using the onsubmit event handler. Validation checks include: required fields not empty, email format (using regex /^[^\s@]+@[^\s@]+\.[^\s@]+$/), password length and strength, and number range checks. The return false statement prevents form submission when validation fails. Regex (Regular Expressions) are powerful pattern-matching tools in JavaScript.

PROCEDURE

1. Create an HTML registration form
2. Add a JavaScript validation function
3. Validate: name not empty, valid email format, password min 8 chars
4. Show error messages using innerHTML or alert
5. Prevent form submission if validation fails
6. Test with valid and invalid inputs

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Form Validation</title>
<style>
  .error { color: red; font-size: 13px; }
  input { padding: 8px; margin: 5px 0; width: 250px; }
</style></head>
<body>
<h2>Registration with Validation</h2>
<form onsubmit="return validate()">
  <input type="text" id="name" placeholder="Full Name"><br>
  <span id="nameErr" class="error"></span><br>
  <input type="email" id="email" placeholder="Email"><br>
```

```
  <span id="emailErr" class="error"></span><br>
  <input type="password" id="pwd" placeholder="Password (min 8 chars)"><br>
  <span id="pwdErr" class="error"></span><br>
  <button type="submit">Submit</button>
</form>
<script>
  function validate() {
    let valid = true;
    let name = document.getElementById('name').value.trim();
    let email = document.getElementById('email').value.trim();
    let pwd = document.getElementById('pwd').value;
    let emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    document.getElementById('nameErr').textContent =
      name === '' ? 'Name is required.' : '';
    if (name === '') valid = false;

    document.getElementById('emailErr').textContent =
      !emailRegex.test(email) ? 'Enter valid email.' : '';
    if (!emailRegex.test(email)) valid = false;

    document.getElementById('pwdErr').textContent =
      pwd.length < 8 ? 'Password min 8 characters.' : '';
    if (pwd.length < 8) valid = false;

    return valid;
  }
</script>
</body></html>
```

EXPECTED OUTPUT
Submitting with empty fields shows red error messages. Submitting with valid data allows form submission.

Program 7	**Write a Program to design a product catalogue webpage using ASP (Active Server Pages).**

THEORY

ASP (Active Server Pages) is a server-side scripting technology by Microsoft. ASP files have .asp extension and can contain HTML and server-side VBScript or JScript. The server processes ASP code before sending the response to the client. Response.Write() outputs content to the browser. ASP allows dynamic page generation, database connectivity, and session management. Modern equivalent is ASP.NET, but classic ASP remains foundational for understanding server-side concepts.

PROCEDURE

1. Set up IIS (Internet Information Services) or an ASP server
2. Create a new file with .asp extension
3. Use HTML for structure and ASP tags <% %> for server-side code
4. Create a product array or connect to a database
5. Loop through products and display as a catalogue
6. Style with CSS and test on server

Source Code

```
<%@ Language="VBScript" %>
<!DOCTYPE html>
<html>
<head>
  <title>Product Catalogue</title>
  <style>
    .product { border:1px solid #ccc; padding:15px; margin:10px;
               display:inline-block; width:200px; border-radius:8px; }
    h1 { color: #1a3c6e; text-align:center; }
  </style>
</head>
<body>
<h1>Product Catalogue</h1>
```

```
<%
  Dim products(3,2)
  products(0,0) = "Laptop" : products(0,1) = "$999" : products(0,2) = "High-performance"
  products(1,0) = "Smartphone" : products(1,1) = "$599" : products(1,2) = "5G enabled"
  products(2,0) = "Tablet" : products(2,1) = "$399" : products(2,2) = "10-inch display"
  products(3,0) = "Headphones" : products(3,1) = "$149" : products(3,2) = "Noise cancelling"

  Dim i
  For i = 0 To 3
    Response.Write "<div class='product'>"
    Response.Write "<h3>" & products(i,0) & "</h3>"
    Response.Write "<p><b>Price:</b> " & products(i,1) & "</p>"
    Response.Write "<p>" & products(i,2) & "</p>"
    Response.Write "</div>"
  Next
%>
</body></html>
```

EXPECTED OUTPUT

A product catalogue page displaying four product cards (Laptop, Smartphone, Tablet, Headphones) with their names, prices, and descriptions.

Program 8	**Write a program to implement event handling and validation on a registration form using JavaScript.**

THEORY

JavaScript event handling allows functions to execute in response to user actions. Events include onclick, onchange, onblur, onfocus, onsubmit, and onkeyup. Event listeners can be added using addEventListener() or inline HTML event attributes. The onblur event triggers when an element loses focus—ideal for field-level validation. Real-time validation using oninput or onkeyup provides immediate feedback as users type, improving user experience.

PROCEDURE

1. Create an HTML form with multiple input fields
2. Attach onblur events to each field for real-time validation
3. Write separate validation functions for each field
4. Add onsubmit event to the form for final validation
5. Display success or error messages dynamically
6. Test all event scenarios

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Event Handling Form</title></head>
<body>
<h2>Event-Driven Registration</h2>
<form id="regForm" onsubmit="return finalValidate()">
  <label>Username:</label>
  <input type="text" id="uname" onblur="checkUsername()" oninput="clearMsg('unameMsg')">
  <span id="unameMsg"></span><br><br>

  <label>Age:</label>
  <input type="number" id="age" onblur="checkAge()">
  <span id="ageMsg"></span><br><br>
```

```
  <label>Phone:</label>
  <input type="text" id="phone" onkeyup="checkPhone()">
  <span id="phoneMsg"></span><br><br>

  <button type="submit">Register</button>
</form>

<script>
  function checkUsername() {
    let u = document.getElementById('uname').value;
    let msg = document.getElementById('unameMsg');
    msg.style.color = u.length >= 4 ? 'green' : 'red';
    msg.textContent = u.length >= 4 ? '✓ Valid' : '✗ Min 4 chars';
  }
  function checkAge() {
    let a = parseInt(document.getElementById('age').value);
    let msg = document.getElementById('ageMsg');
    msg.style.color = (a >= 18 && a <= 60) ? 'green' : 'red';
    msg.textContent = (a >= 18 && a <= 60) ? '✓ Valid' : '✗ Age 18-60';
  }
  function checkPhone() {
    let p = document.getElementById('phone').value;
    let msg = document.getElementById('phoneMsg');
    msg.style.color = /^\d{10}$/.test(p) ? 'green' : 'red';
    msg.textContent = /^\d{10}$/.test(p) ? '✓ Valid' : '10 digits required';
  }
  function clearMsg(id) { document.getElementById(id).textContent = ''; }
  function finalValidate() { checkUsername(); checkAge(); checkPhone(); return false; }
</script>
```

</body></html>

EXPECTED OUTPUT
Real-time validation: username field shows green checkmark after 4+ characters; age shows red error for invalid range; phone validates 10 digits on keyup.

Program 9	**Write a program to open a new browser window/popup when the user performs a Mouse Over event.**

THEORY

JavaScript's window object provides methods to control browser windows. window.open(url, name, specs) opens a new browser window. The specs parameter controls the window dimensions: 'width=400,height=300,left=100,top=100'. The onmouseover event fires when the mouse pointer enters an element. window.close() closes the current window. setTimeout() can auto-close popup windows after a delay. The window.focus() method brings the opened window to the foreground.

PROCEDURE

1. Create an HTML file with an element that triggers the event
2. Attach onmouseover event to the element
3. Use window.open() to open a new popup window
4. Pass dimensions and position in the specs parameter
5. Optionally add a button in the popup to close it
6. Test in browser (disable popup blocker if needed)

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Mouse Over Window</title>
<style>
  .hover-box {
    width:200px; padding:20px; background:#1a3c6e;
    color:white; text-align:center; cursor:pointer;
    border-radius:8px; font-size:18px;
  }
</style></head>
<body>
<h2>Hover Over the Box to Open a Window</h2>
<div class="hover-box" onmouseover="openWin()" onmouseout="closeWin()">
```

```
  Hover Me!
</div>

<script>
  let popup;
  function openWin() {
    popup = window.open('', 'popup',
      'width=400,height=300,top=100,left=200,resizable=no');
    popup.document.write(`
      <html><body style='font-family:Arial; text-align:center; padding:30px;'>
        <h2 style='color:#1a3c6e;'>Hello from Popup!</h2>
        <p>This window opened on Mouse Over event.</p>
        <button onclick='window.close()'>Close</button>
      </body></html>
    `);
    popup.focus();
  }
  function closeWin() {
    if (popup && !popup.closed) popup.close();
  }
</script>
</body></html>
```

EXPECTED OUTPUT

Hovering over the blue box opens a popup window with a message. Moving the mouse away closes the popup.

Program 10	Write a program to create a simple Java Servlet that demonstrates the use of HttpServletRequest and HttpServletResponse objects.

THEORY

Java Servlets are server-side programs that handle HTTP requests and generate responses. The HttpServletRequest object provides methods to retrieve client data: getParameter(), getHeader(), getMethod(), and getRemoteAddr(). The HttpServletResponse object is used to send data back to client: setContentType(), getWriter(), and setStatus(). Servlets extend HttpServlet and override doGet() or doPost() methods. They run inside a Servlet container like Apache Tomcat.

PROCEDURE

1. Install JDK and Apache Tomcat server
2. Create a Dynamic Web Project in Eclipse/IntelliJ
3. Create an HTML form to send data to the servlet
4. Create a Java class extending HttpServlet
5. Override doPost() method
6. Use request.getParameter() to read form data
7. Use response.getWriter() to send HTML response
8. Deploy on Tomcat and test

Source Code

```
// HelloServlet.java

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.WebServlet;


@WebServlet("/hello")

public class HelloServlet extends HttpServlet {

  protected void doPost(HttpServletRequest request,

     HttpServletResponse response)
```

```
    throws ServletException, IOException {

  response.setContentType("text/html");
  PrintWriter out = response.getWriter();

  String name = request.getParameter("username");
  String method = request.getMethod();
  String ip = request.getRemoteAddr();

  out.println("<html><body>");
  out.println("<h2>Servlet Response</h2>");
  out.println("<p>Hello, " + name + "!</p>");
  out.println("<p>Method: " + method + "</p>");
  out.println("<p>Client IP: " + ip + "</p>");
  out.println("</body></html>");
  out.close();
 }
}
```

EXPECTED OUTPUT

A servlet response page displaying the user's name from the form, the HTTP method (POST), and the client's IP address.

Program 11	**Write a program to demonstrate JavaScript's built-in Array and Date object methods on a webpage.**

THEORY

JavaScript provides built-in objects with powerful predefined methods. Array methods include: push() (add to end), pop() (remove from end), shift() (remove from front), unshift() (add to front), splice() (add/remove), slice() (extract), sort() (sort), filter() (filter), map() (transform), and forEach() (iterate). The Date object provides: getDate(), getMonth() (0-indexed), getFullYear(), getHours(), getMinutes(), getSeconds(), toLocaleDateString(), and toLocaleTimeString() for date and time operations.

PROCEDURE

1. Create an HTML file
2. Demonstrate at least 5 Array methods with output
3. Create a Date object and demonstrate date methods
4. Display a real-time clock using setInterval()
5. Show formatted date and time
6. Display all results on the webpage

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Array & Date Objects</title></head>
<body>
<h2>Array Object Methods</h2>
<div id='arrayOutput'></div>
<h2>Date Object Methods</h2>
<div id='dateOutput'></div>
<h3>Live Clock: <span id='clock'></span></h3>

<script>
  // Array Methods Demo
  let fruits = ['Apple', 'Banana', 'Cherry'];
```

```
  fruits.push('Date');
  fruits.unshift('Avocado');
  let sliced = fruits.slice(1, 3);
  let upper = fruits.map(f => f.toUpperCase());
  let longNames = fruits.filter(f => f.length > 5);

  document.getElementById('arrayOutput').innerHTML = `
    <p>Original + push + unshift: ${fruits.join(', ')}</p>
    <p>slice(1,3): ${sliced.join(', ')}</p>
    <p>map (uppercase): ${upper.join(', ')}</p>
    <p>filter (length>5): ${longNames.join(', ')}</p>
  `;

  // Date Methods Demo
  let d = new Date();
  document.getElementById('dateOutput').innerHTML = `
    <p>Full Date: ${d.toLocaleDateString()}</p>
    <p>Year: ${d.getFullYear()}, Month: ${d.getMonth()+1}, Day: ${d.getDate()}</p>
    <p>Day of Week: ${['Sun','Mon','Tue','Wed','Thu','Fri','Sat'][d.getDay()]}</p>
  `;

  // Live Clock
  setInterval(() => {
    document.getElementById('clock').textContent = new Date().toLocaleTimeString();
  }, 1000);
</script>
</body></html>
```

EXPECTED OUTPUT
Webpage showing array manipulation results and a live clock updating every second with current date information.

Program 12	**Write a program to display a calendar for a selected month and year using JavaScript.**

THEORY

A calendar can be generated dynamically using JavaScript's Date object. The key methods are: new Date(year, month, 1) to get the first day of a month, getDay() to find what weekday a date falls on, and a loop to iterate through all days. The calendar is built as an HTML table with 7 columns (Sun-Sat). JavaScript Date handles month boundaries automatically—new Date(year, month, 0).getDate() returns the last day of the previous month.

PROCEDURE

1. Create combo boxes (select) for month and year
2. Write a generateCalendar() function
3. Use Date object to find the first day of selected month
4. Calculate total days in the month
5. Build an HTML table for the calendar
6. Highlight the current date
7. Display on the page

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Calendar</title>
<style>
  table { border-collapse:collapse; }
  td,th { width:40px; height:40px; text-align:center; border:1px solid #ccc; }
  th { background:#1a3c6e; color:white; }
  .today { background:#f39c12; color:white; border-radius:50%; }
</style></head>
<body>
<h2>Dynamic Calendar</h2>
<select id='month' onchange='showCal()'>
```

```
  <option value='0'>January</option><option value='1'>February</option>
  <option value='2'>March</option><option value='3'>April</option>
  <option value='4'>May</option><option value='5'>June</option>
  <option value='6'>July</option><option value='7'>August</option>
  <option value='8'>September</option><option value='9'>October</option>
  <option value='10'>November</option><option value='11'>December</option>
</select>
<select id='year' onchange='showCal()'></select>
<div id='cal'></div>

<script>
  let yrSel = document.getElementById('year');
  for (let y = 2000; y <= 2030; y++) {
    yrSel.innerHTML += `<option value='${y}'${y==new Date().getFullYear()?'
selected':''}>${y}</option>`;
  }
  document.getElementById('month').value = new Date().getMonth();

  function showCal() {
    let m = parseInt(document.getElementById('month').value);
    let y = parseInt(document.getElementById('year').value);
    let today = new Date();
    let first = new Date(y, m, 1).getDay();
    let days = new Date(y, m+1, 0).getDate();
    let html =
'<table><tr><th>Su</th><th>Mo</th><th>Tu</th><th>We</th><th>Th</th><th>Fr</th><th>Sa</th></
tr><tr>';
    for (let i = 0; i < first; i++) html += '<td></td>';
    for (let d = 1; d <= days; d++) {
      let isToday = d==today.getDate() && m==today.getMonth() && y==today.getFullYear();
      html += `<td class='${isToday?"today":""}'>${d}</td>`;
```

```
    if ((first+d)%7===0) html += '</tr><tr>';
   }
   html += '</tr></table>';
   document.getElementById('cal').innerHTML = html;
  }
 showCal();
</script>
</body></html>
```

EXPECTED OUTPUT

A calendar table is displayed for the selected month and year. Today's date is highlighted in orange.

Program 13	**Write a program to create a welcome cookie that displays different content based on whether the user is visiting for the first time or returning.**

THEORY

Cookies are small text files stored in the user's browser by websites. JavaScript can create, read, and delete cookies using document.cookie. A cookie is set as: document.cookie = 'name=value; expires=date; path=/'. Cookies persist across browser sessions (if given an expiry). They are used for authentication, preferences, and tracking. The Date object is used to set cookie expiration. document.cookie returns all cookies as a single string that must be parsed.

PROCEDURE

1. Create an HTML page
2. Write a function to set a cookie with expiry date
3. Write a function to read a specific cookie by name
4. On page load, check if welcome cookie exists
5. If first visit: show welcome message and set cookie
6. If returning: show returning user message with different content
7. Test by clearing cookies and revisiting

Source Code

```
<!DOCTYPE html>
<html>
<head><title>Cookie Demo</title>
<style>
  .welcome { background:#d5f5e3; padding:20px; border-radius:8px; border-left:5px solid #27ae60; }
  .returning { background:#d6eaf8; padding:20px; border-radius:8px; border-left:5px solid #2980b9; }
</style></head>
<body>
<div id='message'></div>
<img id='banner' src='' alt='Banner' style='width:300px; display:none'>
<button onclick='clearCookie()'>Clear Cookie (Simulate New User)</button>
```

```
<script>
  function setCookie(name, value, days) {
    let d = new Date();
    d.setTime(d.getTime() + days*24*60*60*1000);
    document.cookie = name+'='+value+';expires='+d.toUTCString()+';path=/';
  }
  function getCookie(name) {
    let cookies = document.cookie.split(';');
    for (let c of cookies) {
      let [k,v] = c.trim().split('=');
      if (k === name) return v;
    }
    return null;
  }
  function clearCookie() {
    setCookie('visited','',0); location.reload();
  }

  window.onload = function() {
    let visited = getCookie('visited');
    let msg = document.getElementById('message');
    if (!visited) {
      msg.className = 'welcome';
      msg.innerHTML = '<h2>Welcome, New Visitor!</h2><p>Thanks for visiting us for the first time!</p>';
      setCookie('visited','yes',7);
    } else {
      msg.className = 'returning';
      msg.innerHTML = '<h2>Welcome Back!</h2><p>Great to see you again!</p>';
    }
```

```
  };
</script>
</body></html>
```

EXPECTED OUTPUT
First visit shows a green 'Welcome, New Visitor!' box. Refreshing the page shows a blue 'Welcome Back!' box, confirming cookie persistence.

Program 14	**Write a program to demonstrate HTTP request and response handling using an HTML form and server-side processing.**

THEORY

HTML forms use GET or POST methods to send data to a server. In GET requests, form data is appended to the URL as query parameters (?key=value&key2=value2), visible in the address bar and limited in length. In POST requests, data is sent in the HTTP request body, not visible in URL, and supports larger data. The server-side script reads these using request objects (request.getParameter() in Java, $_POST/$_GET in PHP). The response is then sent back as HTML.

PROCEDURE

1. Create an HTML page with a form using POST method
2. Create a server-side script (PHP/Java Servlet)
3. Read form data using request object
4. Process the data (validate, compute, etc.)
5. Generate a response using response object
6. Display response in the browser

Source Code

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head><title>Request/Response Demo</title></head>
<body>
<h2>HTTP Request & Response Demo</h2>
<form action="process.php" method="post">
  <label>First Name: <input type="text" name="fname"></label><br><br>
  <label>Last Name: <input type="text" name="lname"></label><br><br>
  <label>Age: <input type="number" name="age"></label><br><br>
  <input type="submit" value="Submit">
</form>
```

```
<!-- process.php -->
<?php
  $fname = htmlspecialchars($_POST['fname']);
  $lname = htmlspecialchars($_POST['lname']);
  $age   = intval($_POST['age']);

  echo '<h2>Server Response</h2>';
  echo '<p><b>Full Name:</b> ' . $fname . ' ' . $lname . '</p>';
  echo '<p><b>Age:</b> ' . $age . '</p>';

  if ($age >= 18) {
    echo '<p style="color:green">You are eligible to vote.</p>';
  } else {
    echo '<p style="color:red">You are not eligible to vote.</p>';
  }
?>
```

EXPECTED OUTPUT

Submitting the form with 'John', 'Doe', 20 shows: 'Full Name: John Doe', 'Age: 20', and a green 'You are eligible to vote.' message.

Program 15	**Write a program to establish a database connection using Java/PHP and display all records from a database table on a webpage.**

THEORY

Database connectivity allows web applications to store and retrieve persistent data. In PHP, MySQLi or PDO extensions are used. PHP's mysqli_connect(host, user, pass, db) establishes a connection. mysqli_query() executes SQL queries. mysqli_fetch_assoc() retrieves rows. In Java, JDBC (Java Database Connectivity) is used with DriverManager.getConnection(), Statement, and ResultSet objects. Always close connections after use to free resources. SQL's SELECT * FROM table_name retrieves all records.

PROCEDURE

Install XAMPP/WAMP (for PHP+MySQL) or configure MySQL

Create a database and table: CREATE TABLE students(id, name, branch, year)

Insert sample records using INSERT INTO

Create a PHP file to connect to database

Execute SELECT query to fetch all records

Display records in an HTML table

Test in browser

Source Code

```
<?php
// Database Connection
$host = 'localhost';
$user = 'root';
$pass = '';
$db   = 'college';

$conn = mysqli_connect($host, $user, $pass, $db);

if (!$conn) {
  die('Connection failed: ' . mysqli_connect_error());
```

```
}

$sql = 'SELECT * FROM students';
$result = mysqli_query($conn, $sql);

echo '<h2>Student Records</h2>';
echo '<table border="1" cellpadding="8" cellspacing="0">';
echo '<tr style="background:#1a3c6e;color:white">';
echo '<th>ID</th><th>Name</th><th>Branch</th><th>Year</th></tr>';

while ($row = mysqli_fetch_assoc($result)) {
  echo '<tr>';
  echo '<td>' . $row['id'] . '</td>';
  echo '<td>' . $row['name'] . '</td>';
  echo '<td>' . $row['branch'] . '</td>';
  echo '<td>' . $row['year'] . '</td>';
  echo '</tr>';
}
echo '</table>';
mysqli_close($conn);
?>
```

EXPECTED OUTPUT

A styled HTML table displaying all student records from the database with columns: ID, Name, Branch, and Year.