|  |  |
| --- | --- |
| **Sl.No** | **List of programs** |
| 1. | Design the following HTML page that contains <p>,<a>,<img>,<h1>…<h6>,  <div>,<color selector>,<date time selector>,<month selector>,<url selector>.  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>HTML Elements Example</title>  </head>  <body>  <h1>HTML Elements Example</h1>  <p>This is a paragraph element.</p>  <a href="https://www.google.com">This is an anchor element (link).</a>  <img src=" image.jpg" alt="Image description">  <h2>This is a level 2 heading.</h2>  <h3>This is a level 3 heading.</h3>  <h4>This is a level 4 heading.</h4>  <h5>This is a level 5 heading.</h5>  <h6>This is a level 6 heading.</h6>  <div>  <p>This is a div element.</p>  </div>  <input type="color" id="colorPicker">  <label for="dateTime">Select a date and time:</label>  <input type="datetime-local" id="dateTime" name="dateTime">  <label for="month">Select a month:</label>  <input type="month" id="month" name="month">  <label for="url">Enter a URL:</label>  <input type="url" id="url" name="url">  </body>  </html> |
| 2. | * Write JavaScript to validate the following fields of the Registration page.  1. First Name (Name should contain alphabets and the length should not be less than 6 characters). 2. Password (Password should not be less than 6 characters length). 3. E-mail id (should not contain any invalid and must follow the standard pattern name@domain.com) 4. Mobile Number (Phone number should contain 10 numbers)   <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Registration Page</title>  </head>  <body>  <h1>Registration Page</h1>  <form id="registrationForm" onsubmit="return validateForm()">  <label for="firstName">First Name:</label>  <input type="text" id="firstName" name="firstName" required>  <br>  <label for="password">Password:</label>  <input type="password" id="password" name="password" required>  <br>  <label for="email">E-mail:</label>  <input type="email" id="email" name="email" required>  <br>  <label for="mobileNumber">Mobile Number:</label>  <input type="text" id="mobileNumber" name="mobileNumber" required>  <br>  <input type="submit" value="Submit">  </form>  <script>  function validateForm() {  const firstName = document.getElementById('firstName').value;  const password = document.getElementById('password').value;  const email = document.getElementById('email').value;  const mobileNumber = document.getElementById('mobileNumber').value;  const namePattern = /^[A-Za-z]{6,}$/;  const passwordPattern = /^.{6,}$/;  const emailPattern = /^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/;  const mobilePattern = /^\d{10}$/;  if (!namePattern.test(firstName)) {  alert("First Name should contain alphabets and be at least 6 characters long.");  return false;  }  if (!passwordPattern.test(password)) {  alert("Password should be at least 6 characters long.");  return false;  }  if (!emailPattern.test(email)) {  alert("Invalid E-mail format. Please enter a valid E-mail address.");  return false;  }  if (!mobilePattern.test(mobileNumber)) {  alert("Mobile Number should contain 10 numbers.");  return false;  }  return true; // Submit the form if all validations pass  }  </script>  </body>  </html>   * Develop and demonstrate the usage of inline, internal and external style sheet using CSS.   Inline  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Inline Style Example</title>  </head>  <body>  <h1 style="color: blue;">This is a heading with inline style</h1>  <p style="font-size: 18px; color: green;">This is a paragraph with inline style</p>  </body>  </html>  Internal  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Internal Style Example</title>  <style>  h1 {  color: red;  }  p {  font-size: 16px;  color: purple;  }  </style>  </head>  <body>  <h1>This is a heading with internal style</h1>  <p>This is a paragraph with internal style</p>  </body>  </html>  External  Style.css  h1 {  color: orange;  }  p {  font-size: 20px;  color: brown;  }  p.html  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>External Style Example</title>  <link rel="stylesheet" href="style.css">  </head>  <body>  <h1>This is a heading with external style</h1>  <p>This is a paragraph with external style</p>  </body>  </html> |
| 3. | Develop and demonstrate JavaScript with POP-UP boxes and functions for the following problems:   * Input: Click on Display Date button using onclick () function Output: Display date in the textbox   <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Display Date</title>  </head>  <body>  <h1>Display Date Example</h1>  <input type="text" id="dateInput" placeholder="Date will be displayed here">  <br>  <button onclick="displayDate()">Display Date</button>  <script>  function displayDate() {  // Get the current date  const currentDate = new Date();  // Get the day, month, and year from the current date  const day = currentDate.getDate();  const month = currentDate.getMonth() + 1; // Months are zero-indexed, so add 1  const year = currentDate.getFullYear();  // Format the date as "dd/mm/yyyy"  const formattedDate = `${day}/${month}/${year}`;  // Display the formatted date in the textbox  document.getElementById('dateInput').value = formattedDate;    // Show a popup box with the date  alert(`Current date: ${formattedDate}`);  }  </script>  </body>  </html>   * Input: A number n obtained using prompt Output: Factorial of n number using alert   <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Factorial Calculator</title>  </head>  <body>  <h1>Factorial Calculator</h1>  <button onclick="calculateFactorial()">Calculate Factorial</button>  <script>  function calculateFactorial() {  // Prompt the user for a number  const n = prompt('Enter a number to calculate its factorial:');  // Convert the input to a number  const num = parseInt(n);  // Check if the input is a valid number  if (isNaN(num)) {  alert('Invalid input. Please enter a valid number.');  return;  }  // Check if the input is a non-negative integer  if (num < 0) {  alert('Factorial is not defined for negative numbers.');  return;  }  // Calculate the factorial of the number  let factorial = 1;  for (let i = 1; i <= num; i++) {  factorial \*= i;  }  // Display the factorial using alert  alert(`The factorial of ${num} is ${factorial}`);  }  </script>  </body>  </html> |
| 4. | Create an HTML page with three frames <**frame>** elements side by side. Set the source of the first iframe to display a RIT WEBSITE and the source of the second frame to show a respective department and third one to load google map decribing 2 sentence on each frame.  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Frames Example</title>  </head>  <body>  <h1>Welcome to RIT</h1>  <div class="frame-container">  <iframe src="https://www.rit.edu" frameborder="0"></iframe>  <iframe src="https://www.rit.edu/science" frameborder="0"></iframe>  <iframe src="https://www.google.com/maps/ " frameborder="0"></iframe>  </div>  </body>  </html> |
| 5. | Create a base class called `Vehicle` with properties `make` and `model`. Create a derived class called `Car` that inherits from `Vehicle` and has an additional property called `year`. Create an instance of `Car` and display its make, model, and year using JavaScript and HTML.  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Car Information</title>  </head>  <body>  <h1>Car Information</h1>  <div id="carInfo"></div>  <script>  // Base class Vehicle  class Vehicle {  constructor(make, model) {  this.make = make;  this.model = model;  }  }  // Derived class Car inheriting from Vehicle  class Car extends Vehicle {  constructor(make, model, year) {  super(make, model);  this.year = year;  }  }  // Create an instance of Car  const myCar = new Car("Toyota", "Camry", 2022);  // Display the car information in HTML  const carInfoDiv = document.getElementById("carInfo");  carInfoDiv.innerHTML = `  <p>Make: ${myCar.make}</p>  <p>Model: ${myCar.model}</p>  <p>Year: ${myCar.year}</p>  `;  </script>  </body>  </html> |
| 6. | Demonstrate BOX MODEL with having following properties   * **Content** - The content of the box, where text and images appear * **Padding** - Clears an 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  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Box Model Demo</title>  <link rel="stylesheet" href="styles.css">  </head>  <body>  <div class="box">  <p>This is the content of the box.</p>  </div>  </body>  </html>  Css  /\* Styles for the box \*/  .box {  width: 200px;  height: 100px;  background-color: lightblue;  /\* Padding \*/  padding: 20px;  /\* Border \*/  border: 2px solid darkblue;  /\* Margin \*/  margin: 30px;  } |
| 7. | * Create an ordered list (numbered list) and unordered list in HTML that displays the names of five fruits. Apply a CSS style to change the numbering format to uppercase Roman numerals. Your HTML code should produce the following output:   I. Apple  II. Banana  III. Cherry  IV. Durian  V. Elderberry  Write the HTML code to achieve the desired output.  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Fruit List</title>  <link rel="stylesheet" href="styles.css">  </head>  <body>  <h1>List of Fruits</h1>  <div class="fruits">  <ol class="uppercase-roman">  <li>Apple</li>  <li>Banana</li>  <li>Cherry</li>  <li>Durian</li>  <li>Elderberry</li>  </ol>  </div>  <ul>  <li>Apple</li>  <li>Banana</li>  <li>Cherry</li>  <li>Durian</li>  <li>Elderberry</li>  </ul>  </body>  </html>   * Write an HTML page that contains a selection box with a list of 5 countries. When the user selects a country, its capital should be printed next in the list. Add CSS to customize the properties of the font of the capital (color, bold and font size).   html  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Country Capitals</title>  <link rel="stylesheet" href="styles.css">  </head>  <body>  <h1>Select a Country and See Its Capital</h1>  <div>  <select id="countryList" onchange="showCapital()">  <option value="">Select a country</option>  <option value="USA">United States</option>  <option value="UK">United Kingdom</option>  <option value="France">France</option>  <option value="Germany">Germany</option>  <option value="Japan">Japan</option>  </select>  <p id="capital"></p>  </div>  </body>  </html>  Css  /\* Styles for the capital text \*/  #capital {  color: blue;  font-weight: bold;  font-size: 18px;  }  Javascript  <script>  function showCapital() {  const countryList = document.getElementById("countryList");  const capital = document.getElementById("capital");  const selectedCountry = countryList.value;  switch (selectedCountry) {  case "USA":  capital.textContent = "Washington, D.C.";  break;  case "UK":  capital.textContent = "London";  break;  case "France":  capital.textContent = "Paris";  break;  case "Germany":  capital.textContent = "Berlin";  break;  case "Japan":  capital.textContent = "Tokyo";  break;  default:  capital.textContent = "";  break;  }  }  </script> |
| 8. | Demonstrate the following how Elements are then positioned using following properties   * static * relative * fixed * absolute * sticky   <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Positioning Elements</title>  <link rel="stylesheet" href="styles.css">  </head>  <body>  <div class="container">  <div class="static">Static Positioning</div>  <div class="relative">Relative Positioning</div>  <div class="fixed">Fixed Positioning</div>  <div class="absolute">Absolute Positioning</div>  <div class="sticky">Sticky Positioning</div>  </div>  </body>  </html>  Css  /\* Common styles for all elements \*/  .container {  display: flex;  flex-wrap: wrap;  }  div {  width: 150px;  height: 100px;  margin: 10px;  padding: 10px;  border: 2px solid black;  text-align: center;  line-height: 100px;  font-size: 16px;  font-weight: bold;  }  /\* Static Positioning \*/  .static {  position: static;  }  /\* Relative Positioning \*/  .relative {  position: relative;  top: 50px;  left: 50px;  }  /\* Fixed Positioning \*/  .fixed {  position: fixed;  top: 20px;  right: 20px;  }  /\* Absolute Positioning \*/  .absolute {  position: absolute;  bottom: 0;  left: 0;  }  /\* Sticky Positioning \*/  .sticky {  position: sticky;  top: 150px;  } |
| 9. | 1. Given an array of numbers, write a function to filter out all the even numbers and return a new array containing only the odd numbers using java script.   function filterOddNumbers(numbersArray) {  // Use the filter method to create a new array containing only the odd numbers  const oddNumbersArray = numbersArray.filter(number => number % 2 !== 0);  return oddNumbersArray;  }  // Example usage:  const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  const oddNumbers = filterOddNumbers(numbers);  console.log(oddNumbers); // Output: [1, 3, 5, 7, 9]   1. Given an array of numbers, write a function to sort the numbers in ascending order using java script.   function sortAscending(numbersArray) {  // Use the sort method to sort the numbers in ascending order  const sortedArray = numbersArray.sort((a, b) => a - b);  return sortedArray;  }  // Example usage:  const numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5];  const sortedNumbers = sortAscending(numbers);  console.log(sortedNumbers); // Output: [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9] |
| 10. | 1. Given an array of strings, write a function to remove all duplicate strings and return a new array with unique strings only using java script.   function removeDuplicates(stringsArray) {  // Use Set to remove duplicates and convert it back to an array  const uniqueStringsArray = [...new Set(stringsArray)];  return uniqueStringsArray;  }  // Example usage:  const strings = ["apple", "orange", "banana", "apple", "kiwi", "orange"];  const uniqueStrings = removeDuplicates(strings);  console.log(uniqueStrings); // Output: ["apple", "orange", "banana", "kiwi"]   1. Given an array of strings, write a function to create a new array with the first letter of each string capitalized.   function capitalizeFirstLetter(stringsArray) {  const capitalizedArray = stringsArray.map(str => str.charAt(0).toUpperCase() + str.slice(1));  return capitalizedArray;  }  // Example usage:  const strings = ["apple", "banana", "orange", "kiwi"];  const capitalizedStrings = capitalizeFirstLetter(strings);  console.log(capitalizedStrings); // Output: ["Apple", "Banana", "Orange", "Kiwi"] |
| 11. | Write a function called **counter** that returns a closure. The closure should keep track of a count variable and provide two methods: **increment** and **decrement**. The **increment** method should increase the count by 1, and the **decrement** method should decrease the count by 1. The closure should also provide a method **getCount** to return the current count.  function counter() {  let count = 0;  function increment() {  count += 1;  }  function decrement() {  count -= 1;  }  function getCount() {  return count;  }  return {  increment: increment,  decrement: decrement,  getCount: getCount  };  }  // Example usage:  const myCounter = counter();  myCounter.increment();  myCounter.increment();  console.log(myCounter.getCount()); // Output: 2  myCounter.decrement();  console.log(myCounter.getCount()); // Output: 1 |
| 12. | Create a function called **sumArray** that takes an array of numbers as a parameter and returns the sum of all the numbers. Make sure to define the necessary variables within the function scope.  function sumArray(numbersArray) {  let sum = 0;  for (let i = 0; i < numbersArray.length; i++) {  sum += numbersArray[i];  }  return sum;  }  // Example usage:  const numbers = [1, 2, 3, 4, 5];  const result = sumArray(numbers);  console.log(result); // Output: 15 |