**EX.NO: 4**

**JAVASCRIPT**

**DATE: 12/01/2024**

**AIM:**

To create dynamic and interactive web pages using Javascript

**STEPS:**

Create a HTML document.

Link the document with Javascript using an external file or using script tag.

Include necessary functions in the Javascript code to create a dynamic webpage.

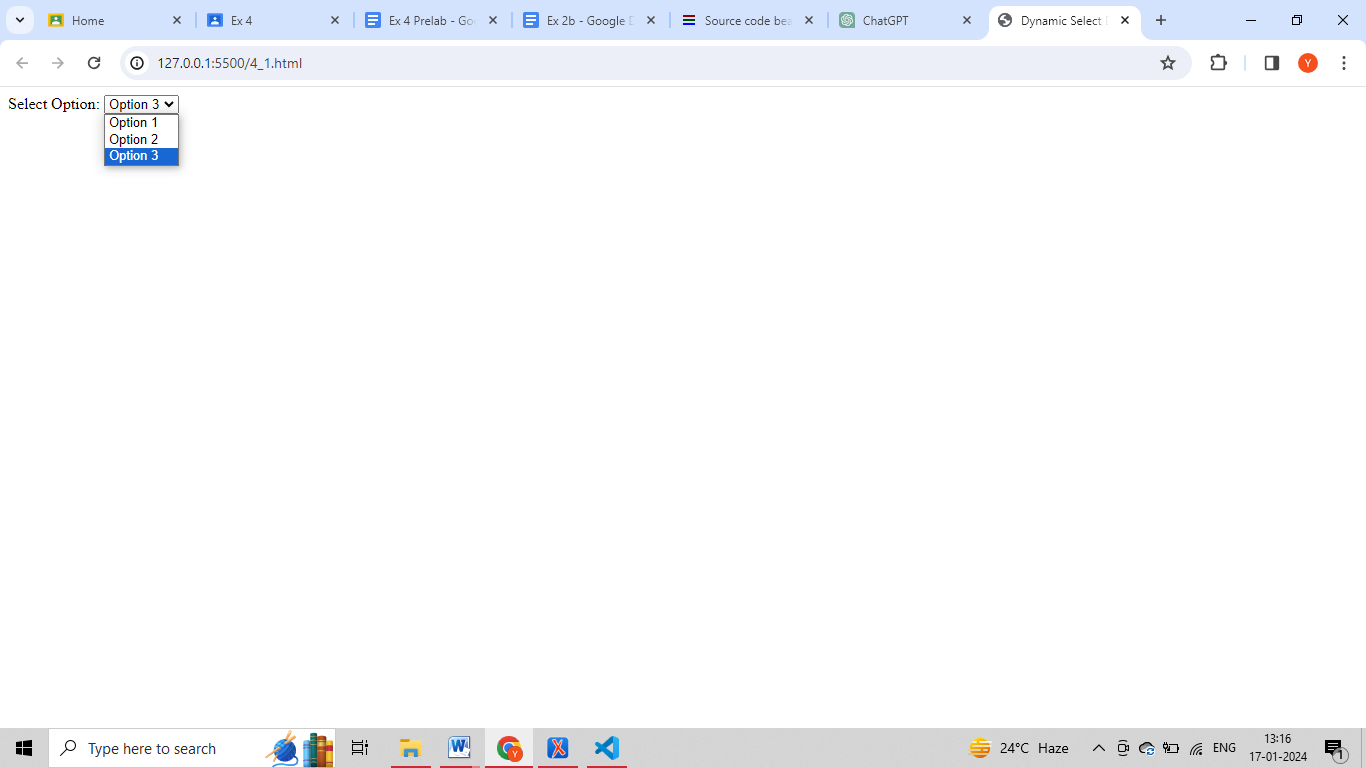
**TOOLS USED:** VS CODE,CHROME BROWSER

1) Create a Dynamic HTML Select Drop Down List Using JavaScript

**CODE: (HTML)**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Dynamic Select Dropdown</title>  </head>  <body>  <label for="dynamicSelect">Select Option:</label>  <select id="dynamicSelect"></select>  <script>  // JavaScript code to populate the dropdown dynamically  **var** dropdown = document.getElementById('dynamicSelect');  **var** options = ["Option 1", "Option 2", "Option 3"];  **for** (**var** i = **0**; i < options.length; i++) {  **var** option = document.createElement('option');  option.value = options[i];  option.text = options[i];  dropdown.add(option);  }  </script>  </body>  </html> |

**OUTPUT:**



2) Create, read and delete a cookie in your web browser using JavaScript

**CODE: (HTML)**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <title>Cookies Example</title>  </head>  <body>  <script>  // Create a cookie  document.cookie = "username=John Doe; expires=Thu, 18 Dec 2024 12:00:00 UTC; path=/";  // Read a cookie  **var** username = document.cookie.split(';')[**0**].split('=')[**1**];  console.log("Username:", username);  // Delete a cookie  document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";  </script>  </body>  </html> |

3) Write a JavaScript function to convert an amount to coins.

Sample function : amountTocoins(46, [25, 10, 5, 2, 1])

Here 46 is the amount. and 25, 10, 5, 2, 1 are coins.

Output : 25, 10, 10, 1

**CODE: (Javascript)**

|  |
| --- |
| **function** amountToCoins(amount, coins) {  **let** result = [];  **for** (**let** i = **0**; i < coins.length; i++) {  **while** (amount >= coins[i]) {  result.push(coins[i]);  amount -= coins[i];  }  }  **return** result;  }  console.log(amountToCoins(**46**, [**25**, **10**, **5**, **2**, **1**])); |

4) Check if the given string is a Palindrome using arrow Functions. Get the input

from the user by prompt() method

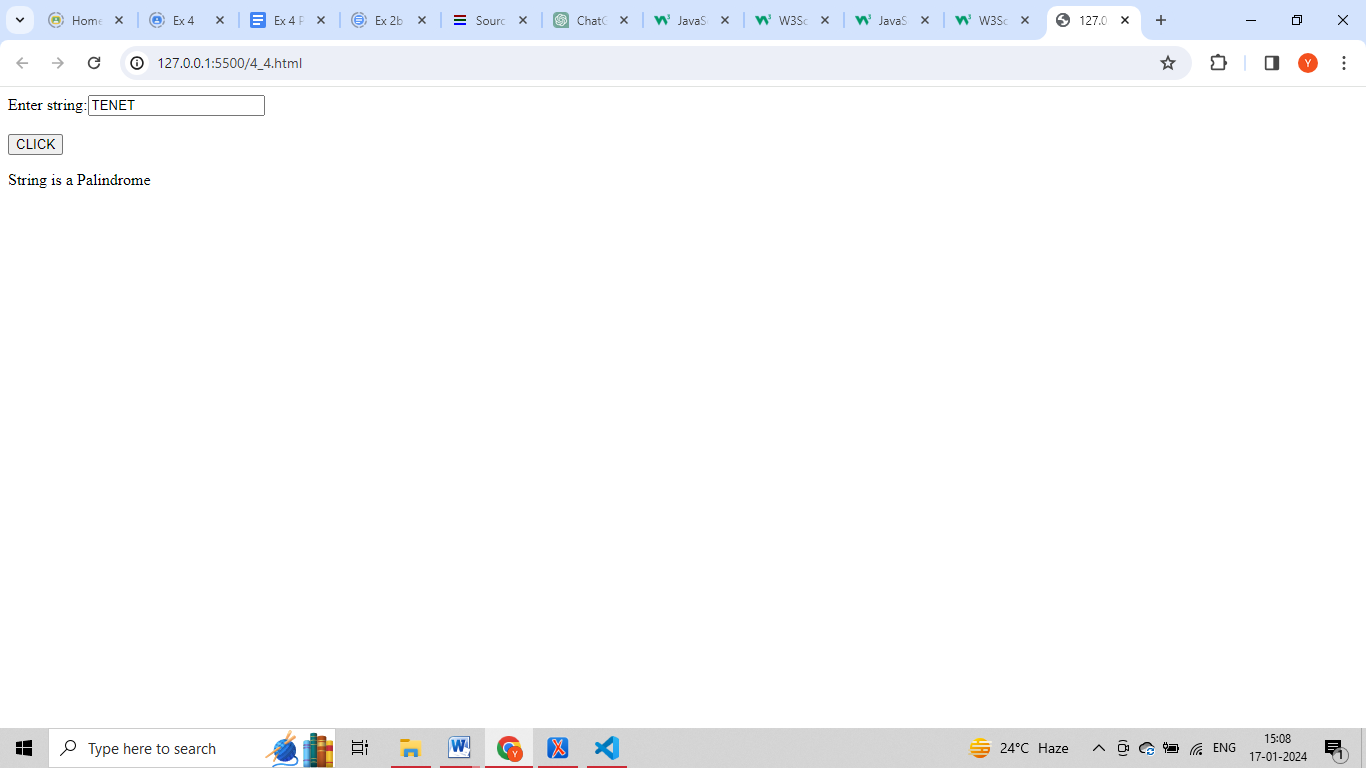
**CODE : (HTML)**

|  |
| --- |
| <html>  <head></head>  <body>  Enter string:<input type="text" id="id1"><br><br>  <input type="button" onclick="check()" value="CLICK">  <p id="id2"></p>  <script src="4\_4.js"></script>  </body>  </html> |

**CODE: (Javascript)**

|  |
| --- |
| **function** check()  {  **let** str=document.getElementById("id1").value ;  **let** rev=str.split('').reverse().join('');  // document.write(rev);  **if**(str==rev)  {  document.getElementById("id2").innerHTML="String is a Palindrome";  }  **else**  {  document.getElementById("id2").innerHTML="String is not a Palindrome";  }  } |

**OUTPUT:**

****

5) a) Create a registration form (with text box, text area, password, email, ph no,

check boxes, radio buttons, prompt and confirm boxes), do validation for all the

fields (provide custom messages)

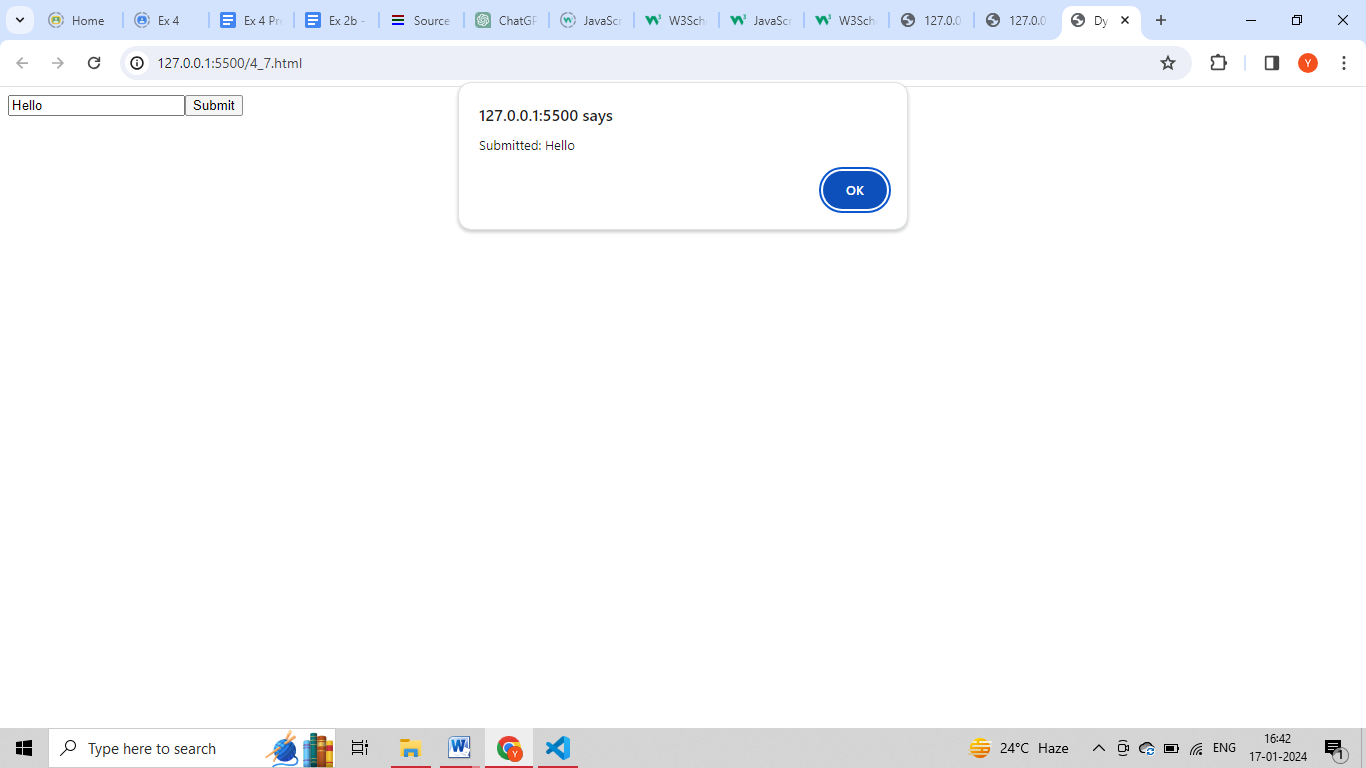
b) How to check script is enabled or not - demonstrate

c) Demonstrate Auto-filling one field same as other

**CODE: (HTML)**

|  |
| --- |
| <HTML>  <form id="registrationForm">  <label for="name">Name:</label>  <input type="text" id="name" required>  <!-- Add other form fields as needed -->  <button type="button" onclick="validateForm()">Submit</button>  </form>    <script>  **function** validateForm() {  **const** name = document.getElementById('name').value;  // Perform validation for other fields    **if** (name === "") {  alert("Name cannot be empty!");  **return**;  }    // Add other validation checks    alert("Form submitted successfully!");  }  document.write("Script is enabled!");  </script>  </HTML> |

**OUTPUT:**



6) Explore various functions like:

Javascript | Arrow functions, JavaScript | escape(), JavaScript | unescape()

JavaScript | Window print(), Javascript | Window Blur() and Window Focus()

Method, JavaScript | console.log(), JavaScript | Replace() Method, JavaScript |

Map.get( ), JavaScript | Map.entries( ) and JavaScript | Map.has( )

**CODE: (Javascript)**

|  |
| --- |
| **const** add = (a, b) => a + b;  console.log(add(**2**, **3**)); // Output: 5  **const** encoded = escape("Hello, World!");  console.log(encoded);  **const** decoded = unescape(encoded);  console.log(decoded);  window.print();  window.blur();  window.focus();  console.log("Hello, console!");  **const** myMap = **new** Map();  myMap.set(**1**, "One");  myMap.set(**2**, "Two");  console.log(myMap.get(**1**)); // Output: One  console.log([...myMap.entries()]); // Output: [[1, 'One'], [2, 'Two']]  console.log(myMap.has(**3**)); // Output: false |

7) Demonstrate the use of Regular expressions with an example.

**CODE: (Javascript)**

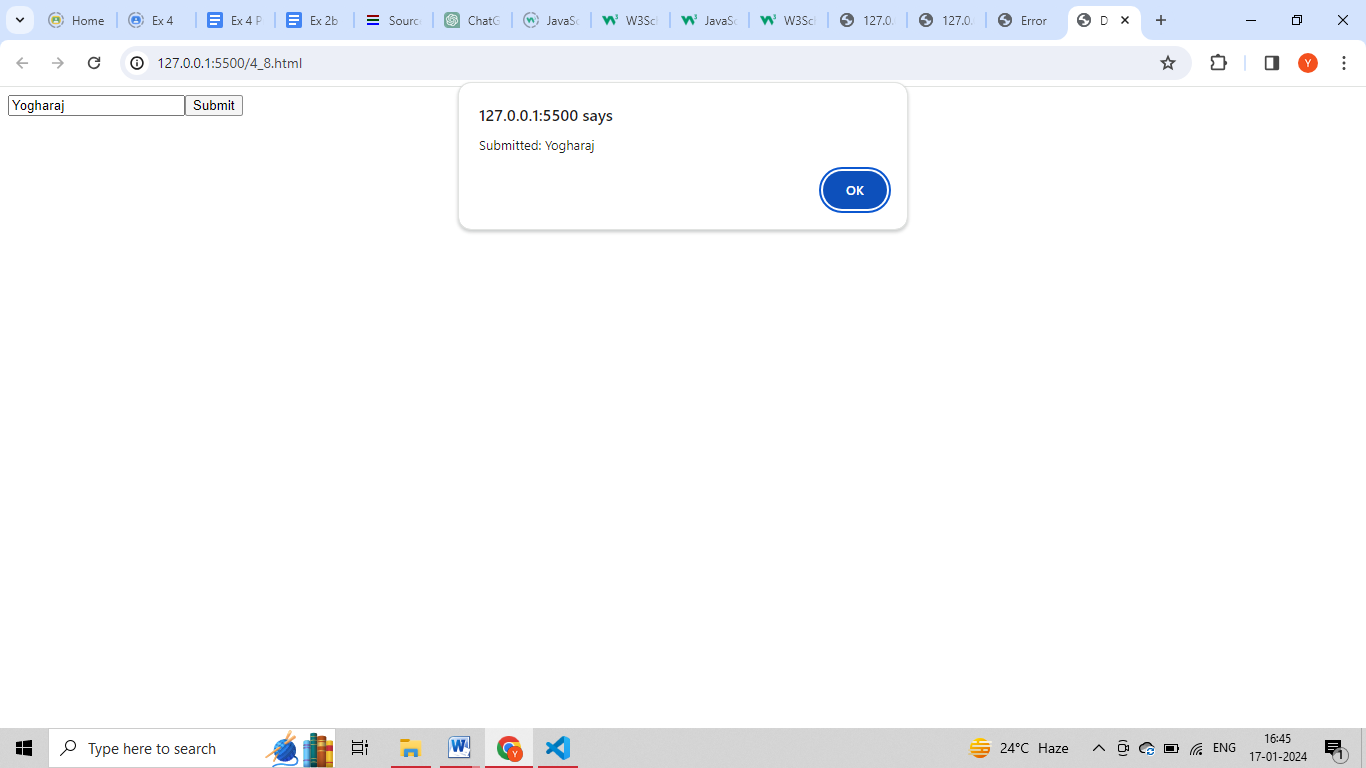
|  |
| --- |
| **const** regex = /^[a-zA-Z0-9]+$/;  **const** userInput = prompt("Enter a string:");  **if** (regex.test(userInput)) {  console.log("Valid input!");  } **else** {  console.log("Invalid input! Only alphanumeric characters are allowed.");  } |

8) Create a Form Dynamically with the JavaScript

**CODE: (HTML)**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Dynamic Form</title>  </head>  <body>  <script>  // Create form dynamically  **const** form = document.createElement('form');  form.id = 'dynamicForm';  // Create input element  **const** input = document.createElement('input');  input.type = 'text';  input.name = 'dynamicInput';  input.placeholder = 'Enter something...';  // Create submit button  **const** submitButton = document.createElement('button');  submitButton.type = 'submit';  submitButton.innerText = 'Submit';  // Append input and button to the form  form.appendChild(input);  form.appendChild(submitButton);  // Append the form to the body  document.body.appendChild(form);  // Add form submission event  form.addEventListener('submit', **function** (event) {  event.preventDefault();  **const** inputValue = input.value;  alert('Submitted: ' + inputValue);  });  </script>  </body>  </html> |

**OUTPUT:**

****

9) Implement the concept of memoization in javascript

**CODE: (Javascript)**

|  |
| --- |
| **function** memoize(func) {  **const** cache = {};  **return** **function** (...args) {  **const** key = JSON.stringify(args);    **if** (cache[key]) {  console.log("Result fetched from cache!");  **return** cache[key];  } **else** {  **const** result = func(...args);  cache[key] = result;  **return** result;  }  };  }  // Example function to be memoized  **const** add = (a, b) => {  console.log("Performing expensive calculation...");  **return** a + b;  };  // Memoizing the 'add' function  **const** memoizedAdd = memoize(add);  // Testing the memoized function  console.log(memoizedAdd(**2**, **3**)); // Output: Performing expensive calculation... 5  console.log(memoizedAdd(**2**, **3**)); // Output: Result fetched from cache! 5 |

10) Implement first class functions and higher order functions in javascript

**CODE: (Javascript)**

|  |
| --- |
| **const** greet = **function** (name) {  **return** "Hello, " + name + "!";  };    // Passing functions as arguments  **const** capitalize = **function** (func, name) {  **return** func(name).toUpperCase();  };    // Returning functions from other functions  **const** getGreeter = **function** () {  **return** **function** (name) {  **return** "Hi, " + name + "!";  };  };    // Using functions as values in objects  **const** person = {  name: "John",  greet: **function** () {  **return** "Hola, " + **this**.name + "!";  },  };    console.log(greet("Alice")); // Output: Hello, Alice!  console.log(capitalize(greet, "Bob")); // Output: HELLO, BOB!  console.log(getGreeter()("Charlie")); // Output: Hi, Charlie!  console.log(person.greet()); // Output: Hola, John!  // Higher-order function that takes a function as an argument  **const** applyOperation = **function** (operation, a, b) {  **return** operation(a, b);  };    // Example operations  **const** add = (a, b) => a + b;  **const** multiply = (a, b) => a \* b;    console.log(applyOperation(add, **2**, **3**)); // Output: 5  console.log(applyOperation(multiply, **2**, **3**)); // Output: 6 |

**RESULT:**

Thus dynamic web pages are created successfully using Javascript and HTML.