Lab 4

Objectives:

The objective of this lab is to learn how to use JavaScript for different tasks, such as creating functions, validating user input, changing web page elements, handling events, and storing data. The Fibonacci series is generated using an arrow function to showcase concise function expressions, while recursion is implemented to calculate the factorial of a user-inputted number. A user registration form is designed with validation for name, email, and password to ensure proper data input. Additionally, a theme toggle feature is implemented using cookies to persist user preferences across sessions. A countdown timer is created to alert users after five minutes, demonstrating time-based event handling. jQuery is utilized for interactive UI enhancement by making div elements vanish on click, showcasing simplified DOM manipulation. Overall, this lab enhances understanding of JavaScript functions, event-driven programming, form validation, and state management in web applications.

Q.1. Write a JavaScript Program depicting the use of arrow function. Use this program to generate a Fibonacci series.

Theory:

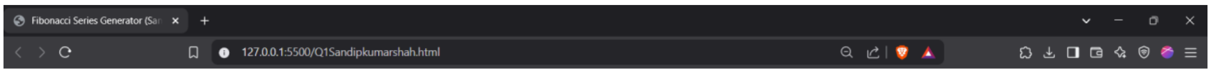
1. Tags:
   - <html>: Defines the root of an HTML document.
   - <head>: Contains metadata and links for the webpage.
   - <title>: Sets the title of the webpage.
   - <body>: Contains the visible content of the webpage.
   - <h1>: Displays the main heading.
   - <label>: Provides a label for the input field.
   - <input>: Accepts user input (number of Fibonacci terms).
   - <button>: Triggers the Fibonacci generation function when clicked.
   - <p>: Displays the generated Fibonacci series.
   - <script>: Embeds JavaScript code within the HTML document.
2. Attributes:
   - id="count": Identifies the input field for retrieving user input.
   - name="count": Specifies the name of the input field.
   - onclick="generateFibonacci()": Calls the JavaScript function when the button is clicked.
   - id="result": Identifies the paragraph element where the output is displayed.
3. JavaScript Functions:
   - fibonacci(num): An arrow function that generates the Fibonacci series up to the given number of terms.
   - generateFibonacci(): Retrieves user input, validates it, calls the fibonacci function, and displays the result.

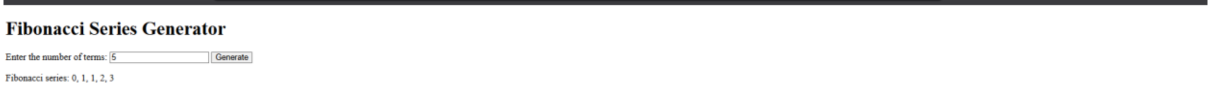
Source Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Fibonacci Series Generator(Sandipkumarshah)</title>

</head>

<body>

  <h1>Fibonacci Series Generator</h1>

  <label for="count">Enter the number of terms:</label>
```

```
    <input type="number" id="count" name="count">
    <button onclick="generateFibonacci()">Generate</button>
    <p id="result"></p>
    <script>
      const fibonacci = (num) => {
        let fibSeries = [0, 1];
        for (let i = 2; i < num; i++) {
          fibSeries.push(fibSeries[i - 1] + fibSeries[i - 2]);
        }
        return fibSeries.slice(0, num);
      };
      const generateFibonacci = () => {
        const count = parseInt(document.getElementById('count').value);
        if (isNaN(count) || count <= 0) {
          document.getElementById('result').innerText = 'Please enter a valid positive integer.';
          console.log("Invalid input: Please enter a valid positive integer.");
        } else {
          const result = fibonacci(count);
          document.getElementById('result').innerText = `Fibonacci series: ${result.join(', ')}`;
          console.log("Fibonacci series:", result);
        }
      };
    </script>
  </body>
  </html>
```

Output:

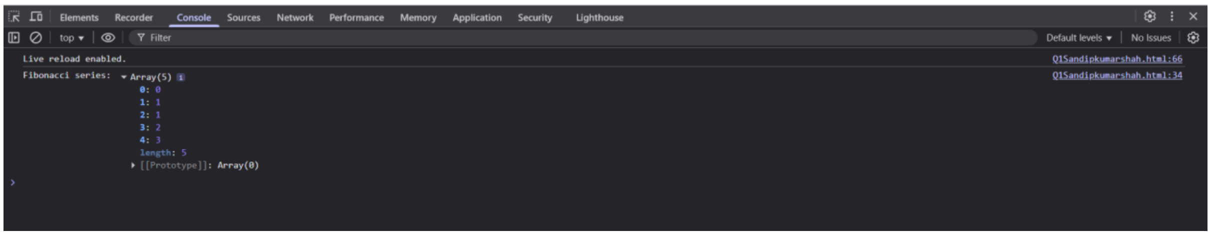

Fibonacci Series Generator (San
127.0.0.1:5500/Q1Sandipkumarshah.html



Fibonacci Series Generator

Enter the number of terms: 5 Generate

Fibonacci series: 0, 1, 1, 2, 3



```
Live reload enabled.                                  Q1Sandipkumarshah.html:66
Fibonacci series:  ▾ Array(5)                         Q1Sandipkumarshah.html:34
                      0: 0
                      1: 1
                      2: 1
                      3: 2
                      4: 3
                      length: 5
                    ▸ [[Prototype]]: Array(0)
>
```

Q.2. Calculate the factorial of a number by using recursive function. Prompt the number from user input.

Theory:

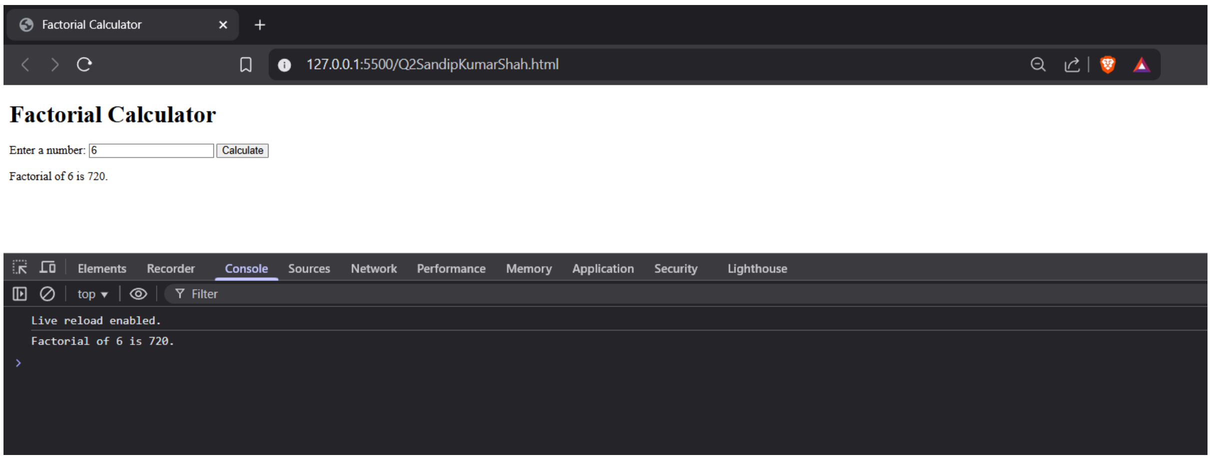
1. Tags:
   - <html>: Defines the root of the HTML document.
   - <head>: Contains metadata and links related to the document.
   - <title>: Sets the title of the webpage.
   - <body>: Contains the visible content of the webpage.
   - <h1>: Displays the main heading "Factorial Calculator".
   - <label>: Provides a label for the input field.
   - <input>: Accepts user input (number for factorial calculation).
   - <button>: Triggers the factorial calculation when clicked.
   - <p>: Displays the result of the factorial calculation.
   - <script>: Contains JavaScript code to perform factorial computation.
2. Attributes:
   - id="number": Identifies the input field to retrieve user input.
   - name="number": Assigns a name to the input field.
   - onclick="calculateFactorial()": Calls the JavaScript function when the button is clicked.
   - id="result": Identifies the paragraph element where the result is displayed.
3. JavaScript Functions:
   - factorial(n): A recursive function that calculates the factorial of a given number.
     - If n is 0 or 1, it returns 1 (base condition).
     - Otherwise, it recursively calls itself (n * factorial(n - 1)) until it reaches the base case.
   - calculateFactorial(): Retrieves the user input, validates it, and displays the result.
     - Gets the value from the <input> field.
     - Ensures the input is a non-negative integer.
     - Calls the factorial(n) function and updates the webpage dynamically using DOM manipulation.

Source Code:

```
<!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>
  <label for="number">Enter a number:</label>
  <input type="number" id="number" name="number">
  <button onclick="calculateFactorial()">Calculate</button>
  <p id="result"></p>
  <script>
    function factorial(n) {
      if (n === 0 || n === 1) {
        return 1;
      } else {
        return n * factorial(n - 1);
      }
    }
    function calculateFactorial() {
      const number = parseInt(document.getElementById('number').value);
      if (isNaN(number) || number < 0) {
        document.getElementById('result').innerText = 'Please enter a valid non-negative
integer.';
       console.log('Invalid input: Please enter a non-negative integer.');
      } else {
        const result = factorial(number);
```

```
            document.getElementById('result').innerText = `Factorial of ${number} is ${result}.`;
            console.log(`Factorial of ${number} is ${result}.`);

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

Output:



Factorial Calculator

Enter a number: 6 Calculate

Factorial of 6 is 720.

```
Live reload enabled.
Factorial of 6 is 720.
>
```

Q.3. Design the form for registration of a new user using HTML. The form must contain fields for Full Name of User, Email Address and Password. Use JavaScript to validate the following:

a. The Name of the user must be at least 3 characters long and should not contain any numbers.

b. Email Address validation

c. Password must be at least 8 characters long and must contain at least 1 number, 1 upper case character, 1 lower case character and 1 special character [@, $, #, etc.]

Theory:

1. Tags and Attributes
    - <html>: Defines the HTML document.
    - <head>: Contains metadata and links to styles/scripts.
    - <title>: Sets the title of the webpage.
    - <body>: Contains the main content of the page.
    - <div class="form-container">: Creates a centered container for the form.
    - <h1>: Displays the title "Register".
    - <form id="registrationForm">: Defines the form and assigns an ID for JavaScript validation.
    - <label for="name">: Provides a label for the "Full Name" input field.
    - <input type="text" id="name" name="name" required>: Text input field for the full name, marked as required.
    - <input type="email" id="email" name="email" required>: Email input field with built-in validation.
    - <input type="password" id="password" name="password" required>: Password input field with required validation.
    - <button type="submit">: Button to submit the form.
    - <script>: Contains JavaScript for validation.
2. CSS Styling
    - body: Centers content and sets a light background color.
    - .form-container: Styles the form container with padding, a shadow, and rounded corners for a modern look.

- h1: Styles the heading with a dark color.
- label: Positions labels above input fields and adds spacing.
- input: Adds padding, a border, and a standard font size for better readability.
- button: Styles the submit button with a green color, rounded corners, and hover effects.

3. JavaScript Validation and Functions

- document.getElementById("registrationForm").addEventListener("submit", function(event) {...}): Listens for the form submission event.
- const name = document.getElementById("name");: Retrieves the full name input field.
- const email = document.getElementById("email");: Retrieves the email input field.
- const password = document.getElementById("password");: Retrieves the password input field.
- if (!/^[A-Za-z]{3,}$/.test(name.value)): Checks if the name is at least 3 characters long and contains only letters.
- if (!/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@!$#])[A-Za-z\d@!$#]{8,}$/.test(password.value)): Validates password complexity.
-
- if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email.value)): Ensures that the email follows the correct format.
- event.preventDefault();: Stops the form submission if validation fails.
- setCustomValidity("Error Message"): Displays custom error messages using built-in browser validation pop-ups.
- reportValidity(): Triggers the browser's native validation error messages.

Source Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Registration Form</title>
```

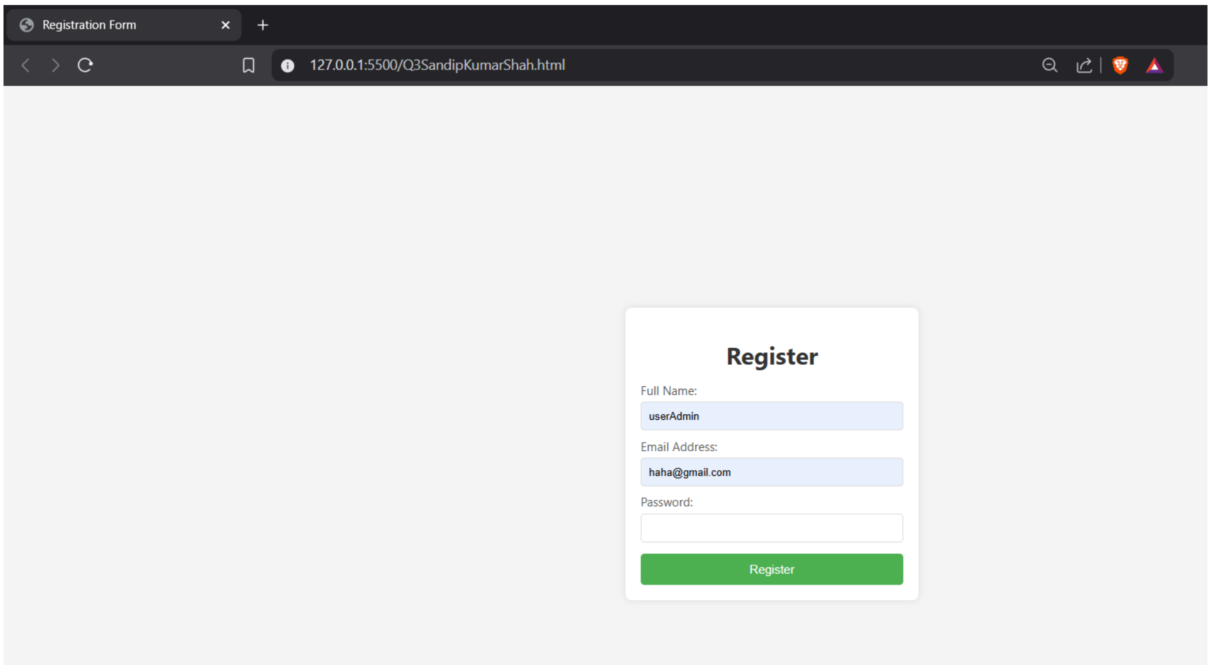
```
<style>
    body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        display: flex;
        justify-content: center;
        align-items: center;
        min-height: 100vh;
        background-color: #f4f4f4;
        margin: 0;
    }
    .form-container {
        text-align: center;
        width: 350px;
        background-color: #fff;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
        color: #333;
        margin-bottom: 15px;
    }
    label {
        display: block;
        margin-top: 10px;
        text-align: left;
        color: #555;
    }
```

```
        input {
            width: 100%;
            padding: 10px;
            margin-top: 5px;
            border: 1px solid #ddd;
            border-radius: 5px;
            font-size: 14px;
            box-sizing: border-box;
        }
        button {
            width: 100%;
            padding: 12px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
            margin-top: 15px;
            transition: background-color 0.3s ease;
        }
    </style>
</head>
<body>
    <div class="form-container">
        <h1>Register</h1>
        <form id="registrationForm">
            <label for="name">Full Name:</label>
            <input type="text" id="name" name="name" required>
```

```
          <label for="email">Email Address:</label>
          <input type="email" id="email" name="email" required>
          <label for="password">Password:</label>
          <input type="password" id="password" name="password" required>
          <button type="submit">Register</button>
       </form>
   </div>
   <script>
                          document.getElementById("registrationForm").addEventListener("submit",
function(event) {
          const name = document.getElementById("name");
          const email = document.getElementById("email");
          const password = document.getElementById("password");
          let isValid = true;
          // Validate Name
          if (!/^[A-Za-z ]{3,}$/.test(name.value)) {
               name.setCustomValidity("Name must be at least 3 characters long and should not
contain numbers.");
              console.log("Error: Name must be at least 3 characters long and should not contain
numbers.");
             isValid = false;
          } else {
             name.setCustomValidity("");
             console.log("Valid Name:", name.value);
          }
          // Validate Email
          if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email.value)) {
             email.setCustomValidity("Please enter a valid email address.");
             console.log("Error: Invalid Email Address.");
             isValid = false;
```

```
            } else {
                email.setCustomValidity("");
                console.log("Valid Email:", email.value);
            }
            // Validate Password
            if (!/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@!$#])[A-Za-z\d@!$#]{8,}$/.test(password.value)) {
                password.setCustomValidity("Password must be at least 8 characters long, with at least 1 number, 1 uppercase letter, 1 lowercase letter, and 1 special character (@,!,$,#, etc.).");
                console.log("Error: Invalid Password.");
                isValid = false;
            } else {
                password.setCustomValidity("");
                console.log("Valid Password:", password.value);
            }
            if (!isValid) {
                event.preventDefault();
                name.reportValidity();
                email.reportValidity();
                password.reportValidity();
            } else {
                console.log("Form submitted successfully!");
            }
        });
    </script>
</body>
</html>
```

Output:



Registration Form

127.0.0.1:5500/Q3SandipKumarShah.html

Register

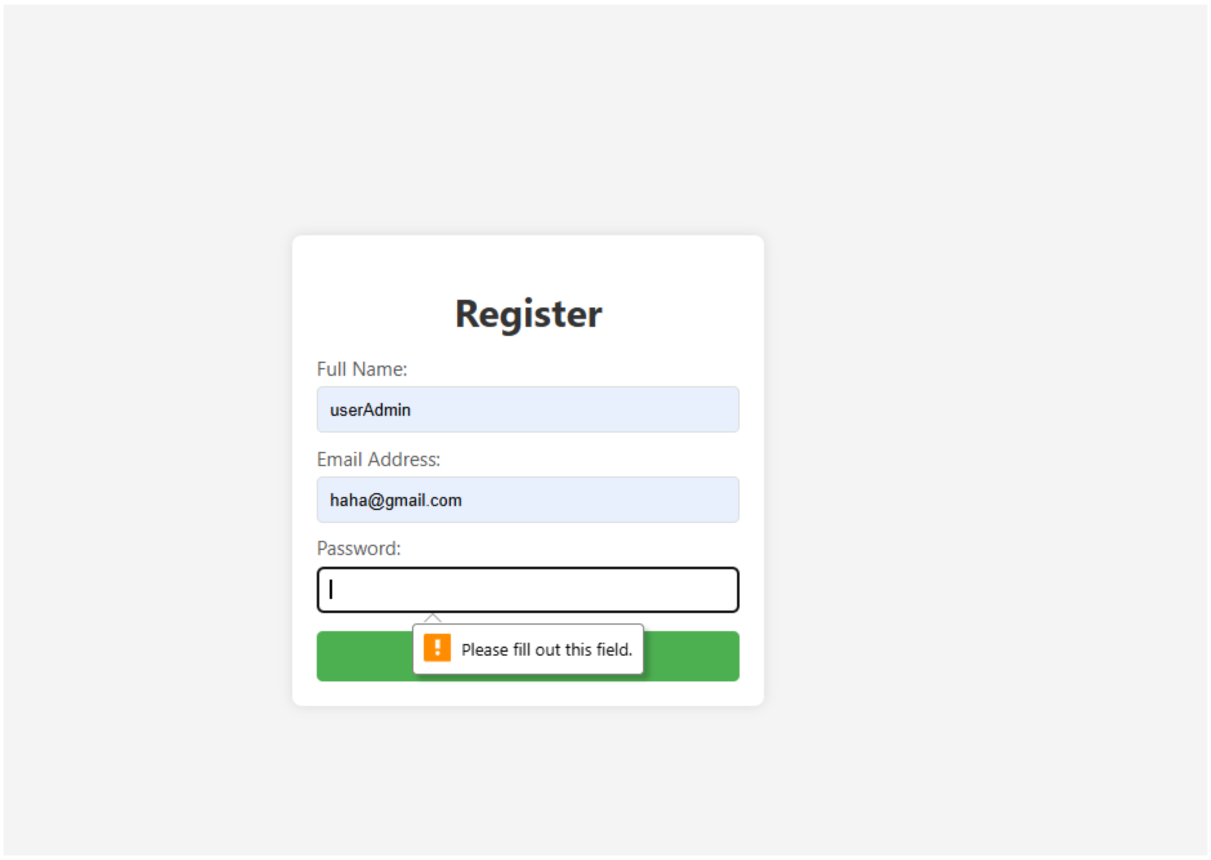
Full Name:

userAdmin

Email Address:

haha@gmail.com

Password:



Register

Full Name:

userAdmin

Email Address:

haha@gmail.com

Password:

Please fill out this field.

Q. 4. Write a JavaScript Program to toggle the back ground color from dark to light and vice versa. However, the mode should persist on reloading as well.

Hint: Make use of cookies.

Theory:

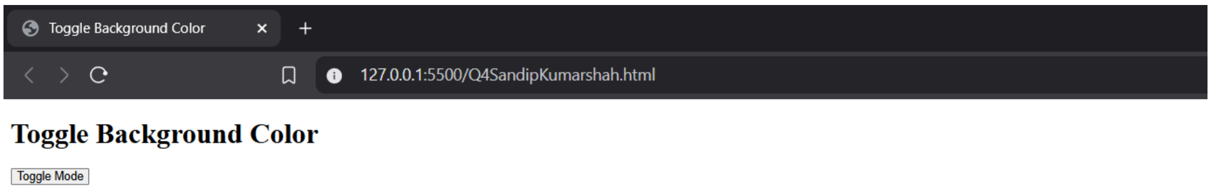
4. Tags and Attributes
    - <!DOCTYPE html> declares the document as an HTML5 file.
    - <html lang="en"> defines the root element of the document with English as the language.
    - <head> contains metadata, styles, and the page title.
    - <title> specifies the title displayed in the browser tab.
    - <style> contains CSS to style the webpage.
    - <body> holds the main content visible on the page.
    - <h1> displays a heading with the text "Toggle Background Color".
    - <button> creates a clickable button for toggling the background color.
    - onclick="toggleMode()" calls the toggleMode() function when the button is clicked.
    - <script> embeds JavaScript to handle theme switching and cookie storage.
    - onclick="toggleMode()" triggers the JavaScript function toggleMode() when the button is clicked.
5. CSS Styling:
    - .dark-mode sets the background color to dark gray and text color to white.
    - .light-mode sets the background color to white and text color to black.
6. JavaScript Functions:
    - setCookie(name, value, days) stores a cookie with a given name, value, and expiration period.
    - getCookie(name) retrieves a stored cookie value based on the given name.
    - toggleMode() switches between light and dark modes and saves the preference in a cookie.
    - loadMode() loads the stored mode from cookies and applies it when the page is opened.
    - window.onload = loadMode; ensures the last selected theme is applied when the page loads.

Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Toggle Background Color</title>
    <style>
        body.dark-mode {
            background-color: #333;
            color: #fff;
        }
        body.light-mode {
            background-color: #fff;
            color: #000;
        }
    </style>
</head>
<body>
    <h1>Toggle Background Color</h1>
    <button onclick="toggleMode()">Toggle Mode</button>
    <script>
        const setCookie = (name, value, days) => {
            const d = new Date();
            d.setTime(d.getTime() + (days * 24 * 60 * 60 * 1000));
            const expires = "expires=" + d.toUTCString();
            document.cookie = name + "=" + value + ";" + expires + ";path=/";
        };
        const getCookie = (name) => {
```

```
        const cname = name + "=";
        const decodedCookie = decodeURIComponent(document.cookie);
        const ca = decodedCookie.split(';');
        for (let i = 0; i < ca.length; i++) {
          let c = ca[i];
          while (c.charAt(0) === ' ') {
            c = c.substring(1);
          }
          if (c.indexOf(cname) === 0) {
            return c.substring(cname.length, c.length);
          }
        }
        return "";
      };
      const toggleMode = () => {
        const currentMode = document.body.classList.contains('dark-mode') ? 'dark' : 'light';
        const newMode = currentMode === 'dark' ? 'light' : 'dark';
        document.body.classList.remove(currentMode + '-mode');
        document.body.classList.add(newMode + '-mode');
        setCookie('mode', newMode, 7);
      };
      const loadMode = () => {
        const mode = getCookie('mode') || 'light';
        document.body.classList.add(mode + '-mode');
      };
      window.onload = loadMode;
    </script>
  </body>
</html
```

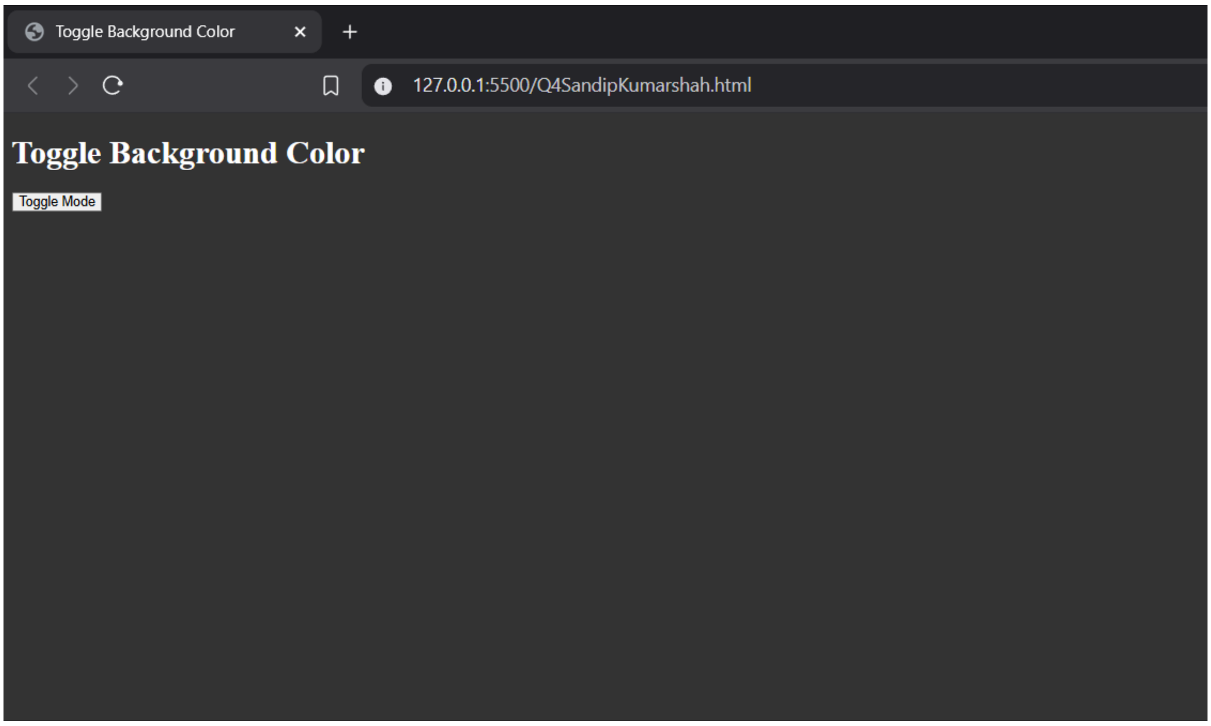
Output:



Toggle Background Color

127.0.0.1:5500/Q4SandipKumarshah.html

Toggle Background Color



Toggle Background Color

127.0.0.1:5500/Q4SandipKumarshah.html

Toggle Background Color

Q.5. Design a countdown Timer using JavaScript. The timer must count till 5 minutes and alert the user of time up.

Theory:

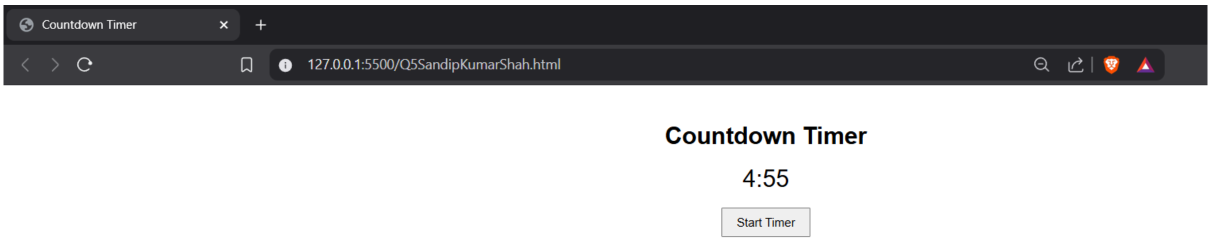
7. HTML Tags and Attributes:
    - <!DOCTYPE html> declares the document as an HTML5 file.
    - <html lang="en"> defines the root element with English as the language.
    - <head> contains metadata, styles, and the title of the page.
    - <title> sets the page title displayed in the browser tab.
    - <style> contains CSS to style the page.
    - <body> holds all the visible content on the webpage.
    - <h1> displays the main heading "Countdown Timer".
    - <div id="timer"> acts as a container to display the countdown timer.
    - <button> creates a clickable button to start the timer.
    - onclick="startTimer()" triggers the JavaScript function when the button is clicked.
    - <script> embeds JavaScript to handle the countdown logic.
    - lang="en" specifies that the document is in English.
    - charset="UTF-8" defines character encoding for text display.
    - name="viewport" helps ensure responsive design.
    - content="width=device-width, initial-scale=1.0" makes the layout adaptable to different screen sizes.
    - id="timer" uniquely identifies the div element displaying the countdown.
    - onclick="startTimer()" runs the startTimer() function when the button is clicked.
8. CSS (Cascading Style Sheets):
    - .body: sets a font style, centers text, and adds a margin to the top.
    - #timer: applies a larger font size and spacing to the countdown display.
    - .button: styles the button with padding, font size, and a pointer cursor for better interaction.
9. JavaScript Functions and Their Purpose:
    - let timeLeft = 300; initializes the countdown time to 5 minutes (300 seconds).
    - let timerInterval; declares a variable to store the timer interval.
    - function startTimer() starts the countdown timer when the button is clicked.
    - if (timerInterval) return; prevents multiple timers from running simultaneously.
    - setInterval(() => { ... }, 1000); updates the timer every second.
    - let minutes = Math.floor(timeLeft / 60); calculates the remaining minutes.
    - let seconds = timeLeft % 60; calculates the remaining seconds.
    - seconds = seconds < 10 ? '0' + seconds : seconds; ensures the seconds display in two digits.
    - document.getElementById("timer").innerText = \${minutes}:${seconds}`;` updates the timer display.

- if (timeLeft === 0) { clearInterval(timerInterval); alert("Time's up!"); } stops the timer and alerts when time runs out.
- timeLeft--; decreases the countdown time by one second.

Source Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Countdown Timer</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      text-align: center;

      margin-top: 50px;

    }

    #timer {

      font-size: 2em;

      margin: 20px;

    }

    button {

      padding: 10px 20px;

      font-size: 1em;

      cursor: pointer;

    }

  </style>

</head>

<body>

  <h1>Countdown Timer</h1>

  <div id="timer">05:00</div>
  <button onclick="startTimer()">Start Timer</button>
  <script>
    let timeLeft = 300;
    let timerInterval;
    function startTimer() {
      if (timerInterval) return;
      timerInterval = setInterval(() => {
        let minutes = Math.floor(timeLeft / 60);
        let seconds = timeLeft % 60;
        seconds = seconds < 10 ? '0' + seconds : seconds;
        document.getElementById("timer").innerText = `${minutes}:${seconds}`;
        if (timeLeft === 0) {
          clearInterval(timerInterval);
          alert("Time's up!");
        }
        timeLeft--;
      }, 1000);
    }
  </script>
</body>
</html>
```

Output:



Countdown Timer

4:55

Q.6. Design a HTML file with 4 divs. Each div should vanish on clicking, Use jQuery.

Theory:

1. Tags and Attributes:
    - <!DOCTYPE html>: Declares the document as an HTML5 file.
    - <html lang="en">: Defines the root HTML element and sets the language to English.
    - <head>: Contains metadata, styles, and script references.
    - <title>: Sets the page title displayed in the browser tab.
    - <script src="https://code.jquery.com/jquery-3.6.0.min.js">: Links to an external jQuery library for JavaScript functionality.
    - <style>: Contains CSS for styling the page.
    - <body> : Holds all visible content on the webpage.
    - <h1>: Displays the main heading.
    - <div class="vanish-div">: Creates a div with a class for styling and interaction.
    - <script>: Contains JavaScript for handling events.
2. CSS (Cascading Style Sheets):
    - body: Sets the font to Arial, aligns text to the center, and adds a top margin.
    - . vanish-div: Styles the divs with width, height, color, and alignment.
    - Width, height: Defines the size of each div.
    - background-color: Sets a light blue background.
    - margin: Centers the divs and spaces them apart.
    - line-height text-align: Aligns the text inside the div both horizontally and vertically.
    - cursor: pointer: Changes the cursor to indicate interactivity.
3. JavaScript (jQuery) Functions:
    - $(document).ready(function(){ ... }); : Ensures the script runs after the page has loaded.
    - $(".vanish-div").click(function(){ ... }); : Adds a click event to all divs with class .vanish-div.

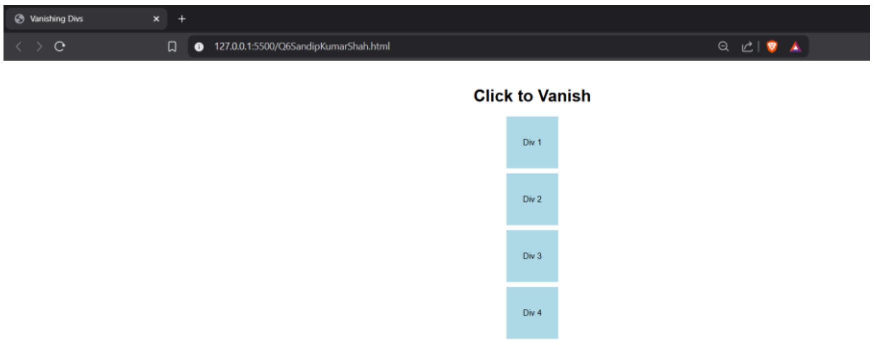
- $(this).fadeOut(); : Makes the clicked div gradually disappear using the fade-out effect.

Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vanishing Divs</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
    }
    .vanish-div {
      width: 100px;
      height: 100px;
      background-color: lightblue;
      margin: 10px auto;
      line-height: 100px;
      text-align: center;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <h1>Click to Vanish</h1>
  <div class="vanish-div">Div 1</div>
```

```
    <div class="vanish-div">Div 2</div>
    <div class="vanish-div">Div 3</div>
    <div class="vanish-div">Div 4</div>
    <script>
        $(document).ready(function(){
            $(".vanish-div").click(function(){
                $(this).fadeOut();
            });
        });
    </script>
</body>
</html>
```

Output:



Vanishing Divs

127.0.0.1:5500/Q6SandipKumarShah.html

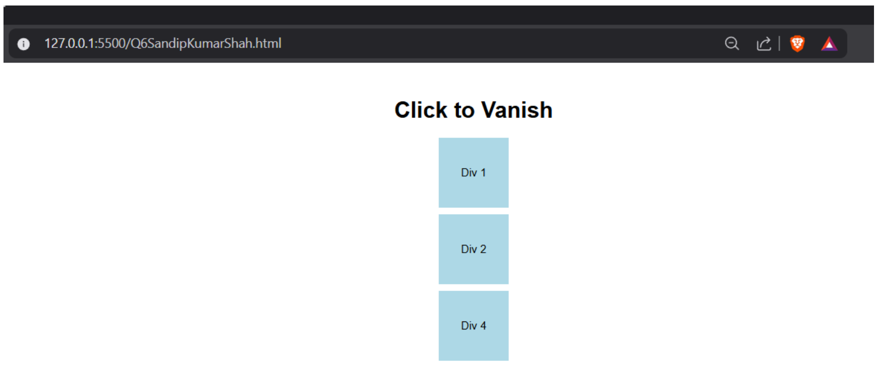
Click to Vanish

Div 1

Div 2

Div 3

Div 4



127.0.0.1:5500/Q6SandipKumarShah.html

Click to Vanish

Div 1

Div 2

Div 4

Conclusion:

In conclusion, this lab helped in understanding the practical use of JavaScript for various web development tasks. By implementing functions, recursion, form validation, event handling, and state management, we learned how JavaScript enhances interactivity and user experience on websites. The use of cookies for theme persistence and jQuery for simple animations further demonstrated the power of JavaScript in modern web applications. Overall, this lab provided hands-on experience in writing efficient and interactive scripts for web development.