Assignment No.1

Aim

Design a Company Registration Form with CSS styles that includes all of the form tag's features. Show the relevant Form output in a compatible browser

Objective

1. To create a Company Registration Form with CSS styles that includes all of the form tag's features. Show the relevant Form output in a compatible browser.

Theory:

HTML (the Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the structure of the page, CSS the (visual and aural) layout, for a variety of devices. Along with graphics and scripting, HTML and CSS are the basis of building Web pages and Web Applications. Learn more below about:

Code

```
<!DOCTYPE html>
<html>
<head>
 <style>
 body {
 font-family: Arial, sans-serif;
 }

 .form-container {
 max-width: 400px;
 margin: 0 auto;
 padding: 20px;
 background-color: #f2f2f2;
 border-radius: 5px;
 }

 .form
</html>
```

Gui (ScreenShots)

Assignment No.2

Aim

Design a Company Registration Form with CSS styles that includes all of the form tag's features. Show the relevant Form output in a compatible browser

Objective

1. Design a Comprehensive Form Layout:
Include essential input fields like name, email, password, and contact information.
2. Apply CSS for Enhanced Aesthetics:
Improve the form's visual appeal through typography, colors, and layout .

Theory:

HTML Forms

HTML forms are essential for data collection and user interaction on web pages. They consist of various form elements like text inputs, radio buttons, checkboxes, dropdown menus, and more. These elements are wrapped within the <form> tag, which provides the structure for input fields and defines the action (URL to send the data) and method (HTTP method, like GET or POST) for form submission.

Form Elements

- Text Inputs: Used to capture textual data, such as names, addresses, and passwords.
- Email Inputs: Specialized input fields for collecting email addresses, offering validation for proper format.
- Password Inputs: Secure fields for collecting passwords, where characters are obscured.
- Radio Buttons and Checkboxes: Allow users to select one or multiple options.
- Select Menus: Dropdown lists from which users can choose one option.
- Buttons: For form submission and resetting the form fields.

CSS Styling

- Cascading Style Sheets (CSS) enhance the visual presentation of forms, making them user-friendly and aesthetically pleasing. Key aspects of CSS styling for forms include:
- Layout: Organizing form elements using CSS properties like display, flex, and grid to create a clean and intuitive layout.
- Typography: Styling text elements (labels, input placeholders) for readability using fonts, sizes, and colors.
- Spacing: Applying margin and padding to ensure proper spacing between form elements.
- Colors and Backgrounds: Using color schemes that align with the brand identity, improving user engagement.
- Borders and Shadows: Enhancing form fields with borders, rounded corners, and shadows for a modern look.

- Transitions and Animations: Implementing subtle transitions for interactive elements like buttons and inputs to enhance user experience.

Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ember & Vine</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <div class="image-container">
            <img src="restaurant-image.jpg" alt="Ember & Vine Restaurant"
class="background-image">
        </div>
        <div class="form-container">
            <h1>Welcome to Ember & Vine</h1>
            <div class="tabs">
                <button class="tab-button active"
onclick="showForm('loginForm')">Login</button>
                <button class="tab-button"
onclick="showForm('registerForm')">Register</button>
            </div>
            <div id="loginForm" class="form-content active">
                <form>
                    <label for="loginUsername">Username:</label>
                    <input type="text" id="loginUsername" name="username"
required>

                    <label for="loginPassword">Password:</label>
                        <input type="password" id="loginPassword"
name="password" required>

                    <button type="submit">Login</button>
```

```
                    <div id="loginMessage" class="form-message"></div>
                </form>
            </div>
            <div id="registerForm" class="form-content">
                <form>
                    <label for="name">Name:</label>
                    <input type="text" id="name" name="name" required>

                    <label for="email">Email:</label>
                    <input type="email" id="email" name="email" required>

                    <label for="registerUsername">Username:</label>
                    <input type="text" id="registerUsername"
name="username" required>

                    <label for="registerPassword">Password:</label>
                        <input type="password" id="registerPassword"
name="password" required>

                    <button type="submit">Register</button>
                    <div id="registerMessage" class="form-message"></div>
                </form>
            </div>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>

<style>

@import
url('https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&disp
lay=swap');

    body {
    font-family: 'Roboto', sans-serif;
```

```
    margin: 0;
    padding: 0;
    background-color: #2c3e50;
    color: #000000;
}

.container {
    display: flex;
    height: 100vh;
    align-items: center;
    justify-content: center;
    position: relative;
}

.image-container {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    overflow: hidden;
    z-index: -1;
}

.background-image {
    width: 100%;
    height: 100%;
    object-fit: cover;
    opacity: 0.5;
}

.form-container {
    background: #e9f1f9;
    border-radius: 8px;
    padding: 20px;
    max-width: 500px;
    width: 100%;
```

```
    position: relative;
}

h1 {
    text-align: center;
    color: #c6391a;
    margin-bottom: 20px;
}

.tabs {
    display: flex;
    justify-content: space-between;
    margin-bottom: 20px;
}

.tab-button {
    background: #c6391a;
    color: #ecf0f1;
    border: none;
    padding: 10px;
    width: 50%;
    cursor: pointer;
    border-radius: 8px;
    margin-right: 5px;
}

.tab-button.active {
    background: #891d05;
}

.form-content {
    display: none;
}

.form-content.active {
    display: block;
}
```

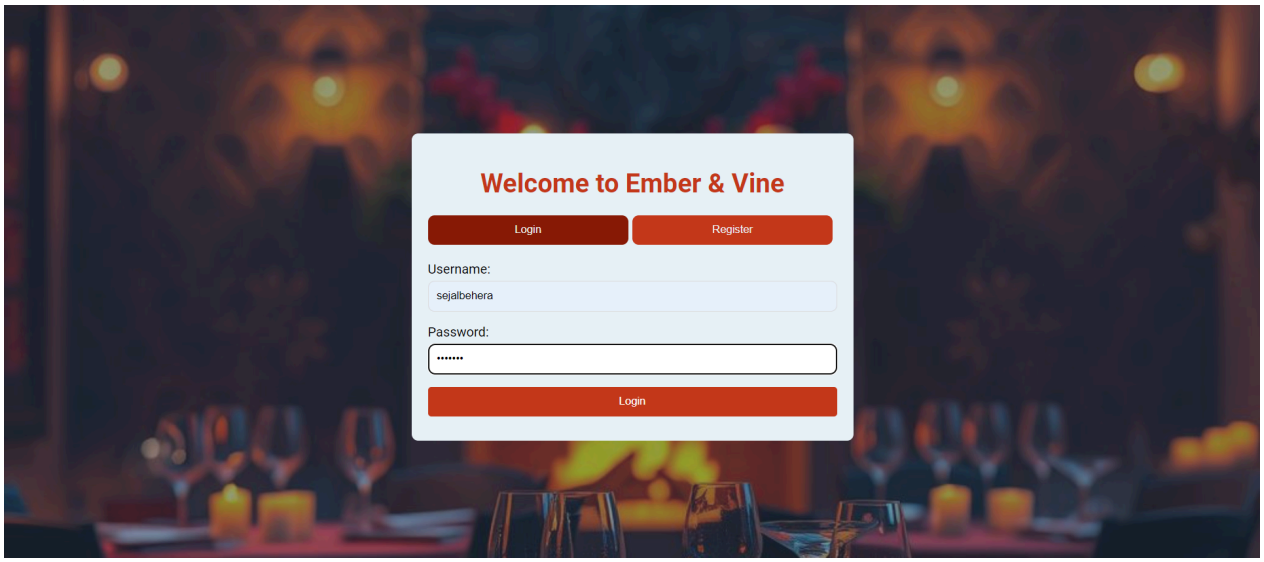
```
form {
    display: flex;
    flex-direction: column;
}

label {
    margin-bottom: 5px;
}

input {
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ddd;
    border-radius: 8px;
}

button {
    padding: 10px;
    background-color: #c6391a;
    border: none;
    color: #fff;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #952911;
}

.form-message {
    color: #e74c3c;
    text-align: center;
    margin-top: 10px;
}

</style>
```

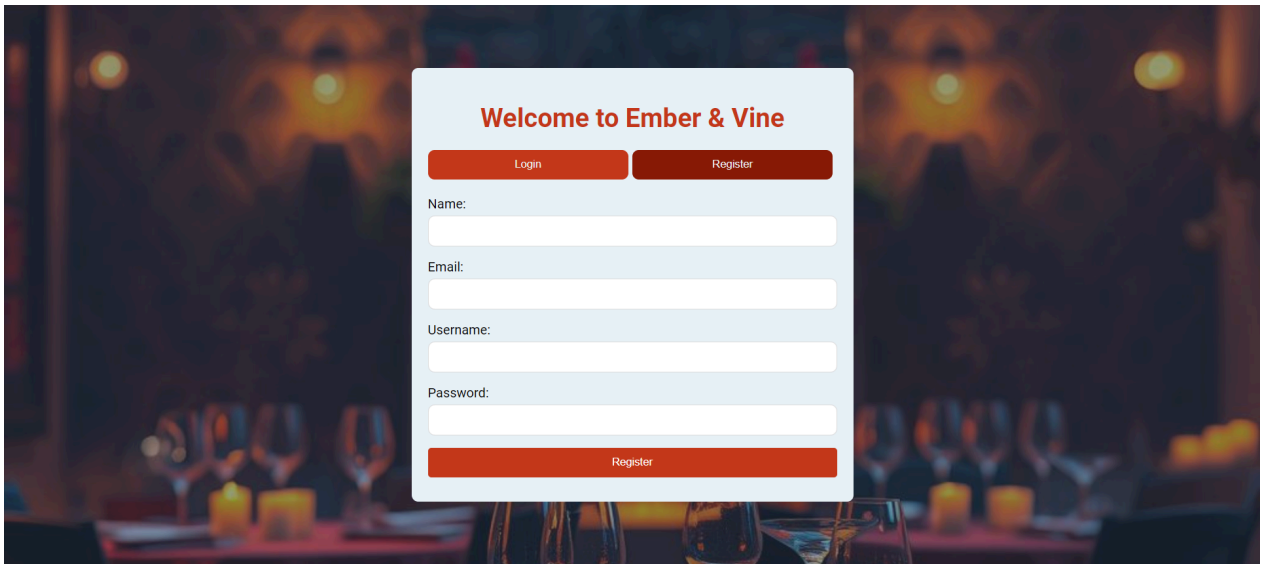
```
<script>
    function showForm(formId) {
    document.querySelectorAll('.form-content').forEach(function(form) {
        form.classList.remove('active');
    });
    document.getElementById(formId).classList.add('active');

    document.querySelectorAll('.tab-button').forEach(function(button) {
        button.classList.remove('active');
    });

document.querySelector(`.tab-button[onclick="showForm('${formId}')"]`).cla
ssList.add('active');
}

document.getElementById('loginForm').querySelector('form').addEventListene
r('submit', function(event) {
    event.preventDefault();

    const username = document.getElementById('loginUsername').value;
    const password = document.getElementById('loginPassword').value;
    const loginMessage = document.getElementById('loginMessage');

    if (username === '' || password === '') {
        loginMessage.textContent = 'Please fill in both fields.';
    } else {
        loginMessage.textContent = '';
        console.log('Login Username:', username);
        console.log('Login Password:', password);
        loginMessage.textContent = 'Login successful!';
    }
});

document.getElementById('registerForm').querySelector('form').addEventList
ener('submit', function(event) {
    event.preventDefault();
```

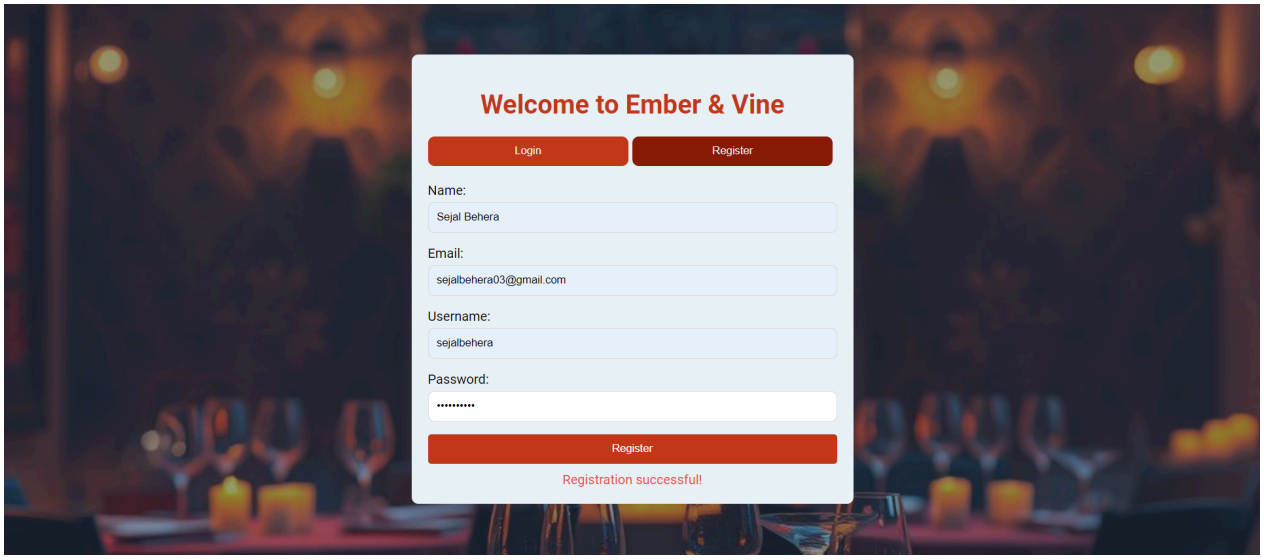
```
    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const username = document.getElementById('registerUsername').value;
    const password = document.getElementById('registerPassword').value;
    const registerMessage = document.getElementById('registerMessage');

    if (name === '' || email === '' || username === '' || password === '')
{
        registerMessage.textContent = 'Please fill in all fields.';
    } else {
        registerMessage.textContent = '';
        console.log('Name:', name);
        console.log('Email:', email);
        console.log('Register Username:', username);
        console.log('Register Password:', password);
        registerMessage.textContent = 'Registration successful!';
    }
});

function togglePassword(inputId) {
    const input = document.getElementById(inputId);
    const button = input.nextElementSibling;
    if (input.type === 'password') {
        input.type = 'text';
        button.innerHTML = '<img src="eye-open-icon.png" alt="Hide
Password" width="20">';
    } else {
        input.type = 'password';
        button.innerHTML = '<img src="eye-icon.png" alt="Show Password"
width="20">';
    }
}

document.addEventListener('DOMContentLoaded', function() {
    showForm('loginForm');
});
```

```
</script>
```

Gui (ScreenShots)







Conclusion

In conclusion, a company registration form is a fundamental part of any web application that aims to onboard users effectively. For Ember and Vine, the registration form is more than just a data collection tool; it's a reflection of the brand's commitment to providing a seamless and engaging user experience. By incorporating all the features of the `<form>` tag and applying thoughtful CSS styling, the form becomes a powerful instrument in the company's digital presence. Ensuring compatibility across different browsers further emphasizes the company's dedication to accessibility and customer satisfaction.

Assignment No.3

Aim

Write a JavaScript code to take inputs from user and display that inputs in following pattern "Hello ..... Welcome To World of JavaScript". Write a JavaScript code to perform arithmetic operation by taking values from users (Add, sub, Mul,Div)(1. 4 Button, 2.Dropdown)

Objective

1. Personalized Greeting:
Implement a JavaScript function to capture user input (e.g., name) and display a customized greeting message.
2. Arithmetic Operations Interface:
Create an interface with four buttons to perform addition, subtraction, multiplication, and division. Implement a dropdown menu for selecting the operation and corresponding fields to input numbers.
3. User Input Handling:
Ensure that the JavaScript code correctly captures and processes user inputs. Display results dynamically based on user interaction.

Theory:

1. **HTML Form**: The HTML form is used to collect input from the user. It includes a text input field where the user can enter their name and a button to submit the input.
    - **Dropdown Form**: Provides a dropdown menu to select an arithmetic operation (Add, Subtract, Multiply, Divide) and input fields for two numbers.
    - **Button Form**: Contains four buttons, each corresponding to a different arithmetic operation. Users enter numbers and click a button to perform the selected operation.
2. **JavaScript Function**:
    - **Accessing User Input**: JavaScript retrieves the value entered by the user through the `document.getElementById('userInput').value` method.
    - **String Manipulation**: To ensure the message fits the desired pattern, JavaScript calculates the number of dots needed to align the input text. This is achieved using the `String.prototype.repeat()` method, which creates a string consisting of a specified number of repetitions of a given string (in this case, dots).
    - **Formatting Output**: The message is then formatted to include the user input and the required number of dots, resulting in the final output.
    - **Dropdown Form**:
        - **Retrieving Values**: JavaScript retrieves the values of the two numbers and the selected operation.

        - **Performing Operations**: Based on the selected operation, a `switch` statement is used to execute the corresponding arithmetic operation (addition, subtraction, multiplication, or division). Special handling is included for division to avoid division by zero.
        - **Displaying Results**: The result of the calculation is displayed on the web page.
    - **Button Form**:
        - **Handling Button Clicks**: Each button triggers the `performArithmetic` function with a specific operation as an argument.
        - **Performing Operations**: The function performs the specified arithmetic operation and displays the result.

Code

1.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>JavaScript Greeting</title>
</head>
<body>
    <div class="container">
        <h1>Enter Your Name</h1>
        <input type="text" id="userName" placeholder="Enter your
name">
        <button onclick="displayGreeting()">Submit</button>
        <div id="greetingMessage"></div>
    </div>
```

```
    <script src="script.js"></script>
</body>
</html>

<script>
    function displayGreeting() {
    const userName = document.getElementById('userName').value;
    if (userName) {
        const greetingMessage = `Hello ${userName}, Welcome To World of JavaScript.`;
        document.getElementById('greetingMessage').textContent = greetingMessage;
    } else {
        document.getElementById('greetingMessage').textContent = 'Please enter your name.';
    }
}
</script>

<style>
    body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #1a1a2e;
    color: #e0e0e0;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
```

```
    height: 100vh;
}

.container {
    background-color: #16213e;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    text-align: center;
    width: 320px;
}

h1 {
    margin-bottom: 25px;
    color: #e0e0e0;
    font-size: 1.8em;
}

input[type="text"] {
    padding: 12px;
    margin: 10px 0;
    border: 1px solid #314e52;
    border-radius: 5px;
    width: calc(100% - 24px);
    background-color: #e0e0e0;
    color: #16213e;
    font-size: 1em;
}
```

```
button {
    padding: 12px 25px;
    margin-top: 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    background-color: #00a8cc;
    color: #fff;
    font-size: 1em;
    transition: background-color 0.3s, transform 0.3s;
}

button:hover {
    background-color: #005377;
    transform: translateY(-2px);
}

#greetingMessage {
    margin-top: 25px;
    font-size: 1.2em;
    font-weight: bold;
    color: #00a8cc;
}

</style>
```

2. Part i

```
<!-- Buttons -->
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Calculator - Buttons</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Calculator</h1>
        <div class="input-container">
            <input type="number" id="value1" placeholder="Enter
first value">
            <input type="number" id="value2" placeholder="Enter
second value">
        </div>
        <div class="button-container">
            <button onclick="performOperation('add')">Add</button>
            <button
onclick="performOperation('sub')">Subtract</button>
            <button
onclick="performOperation('mul')">Multiply</button>
            <button
```

```
onclick="performOperation('div')">Divide</button>
            </div>
            <div id="result-container">
                <h2>Result</h2>
                <div id="result"></div>
            </div>
        </div>
        <script src="script-buttons.js"></script>
    </body>
    </html>


    <style>
        body {
        font-family: Arial, sans-serif;
        background-color: #2c3e50;
        color: #ecf0f1;
        margin: 0;
        padding: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
    }

    .container {
        background-color: #34495e;
        padding: 30px;
```

```
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        text-align: center;
        width: 300px;
    }

    h1 {
        margin-bottom: 20px;
        color: #ecf0f1;
    }

    .input-container {
        margin-bottom: 20px;
    }

    .input-container input {
        padding: 10px;
        margin: 5px 0;
        border: 1px solid #ccc;
        border-radius: 4px;
        width: calc(100% - 22px);
        background-color: #ecf0f1;
    }

    .button-container {
        margin-bottom: 20px;
    }

```

```
.button-container button {
    padding: 10px 20px;
    margin: 0 5px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    background-color: #3498db;
    color: #fff;
}

.button-container button:last-child {
    margin-top: 10px;
}

.button-container button:hover {
    background-color: #2980b9;
}

.dropdown-container {
    margin-bottom: 20px;
}

.dropdown-container select {
    padding: 10px;
    margin-right: 10px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
```

```
        background-color: #ecf0f1;
        color: #2c3e50;
    }

    #result-container {
        margin-top: 20px;
    }

    #result-container h2 {
        margin-bottom: 10px;
        color: #ecf0f1;
    }

    #result {
        font-size: 1.2em;
        font-weight: bold;
        color: #ecf0f1;
    }



    </style>

    <script>
        function performOperation(operation) {
        const value1 =
parseFloat(document.getElementById('value1').value);
        const value2 =
parseFloat(document.getElementById('value2').value);
        let result;
```

```
226.
227.         if (isNaN(value1) || isNaN(value2)) {
228.             result = 'Please enter valid numbers';
229.         } else {
230.             switch (operation) {
231.                 case 'add':
232.                     result = value1 + value2;
233.                     break;
234.                 case 'sub':
235.                     result = value1 - value2;
236.                     break;
237.                 case 'mul':
238.                     result = value1 * value2;
239.                     break;
240.                 case 'div':
241.                     if (value2 === 0) {
242.                         result = 'Cannot divide by zero';
243.                     } else {
244.                         result = value1 / value2;
245.                     }
246.                     break;
247.             }
248.         }
249.         document.getElementById('result').textContent = `Result:
    ${result}`;
250.     }
251.
252.     </script>
```

Part ii

```
<!-- Dropdown -->



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Calculator - Dropdown</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="container">

        <h1>Calculator</h1>

        <div class="input-container">

            <input type="number" id="value1" placeholder="Enter first
value">

            <input type="number" id="value2" placeholder="Enter second
value">

        </div>

        <div class="dropdown-container">
```

```
            <select id="operationSelect">
                <option value="add">Add</option>
                <option value="sub">Subtract</option>
                <option value="mul">Multiply</option>
                <option value="div">Divide</option>
            </select>
            <button onclick="performOperation()">Calculate</button>
        </div>
        <div id="result-container">
            <h2>Result</h2>
            <div id="result"></div>
        </div>
    </div>
    <script src="script-dropdown.js"></script>
</body>
</html>




<style>
```

```
    body {
    font-family: Arial, sans-serif;
    background-color: #2c3e50;
    color: #ecf0f1;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}


.container {
    background-color: #34495e;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
    width: 300px;
}
```

```css
h1 {
    margin-bottom: 20px;
    color: #ecf0f1;
}

.input-container {
    margin-bottom: 20px;
}

.input-container input {
    padding: 10px;
    margin: 5px 0;
    border: 1px solid #ccc;
    border-radius: 4px;
    width: calc(100% - 22px);
    background-color: #ecf0f1;
}

.button-container, .dropdown-container {
```

```
    margin-bottom: 20px;
}

.button-container button, .dropdown-container button {
    padding: 10px 20px;
    margin: 0 5px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    background-color: #3498db;
    color: #fff;
}

.button-container button:hover, .dropdown-container button:hover {
    background-color: #2980b9;
}

.dropdown-container select {
    padding: 10px;
    margin-right: 10px;
```

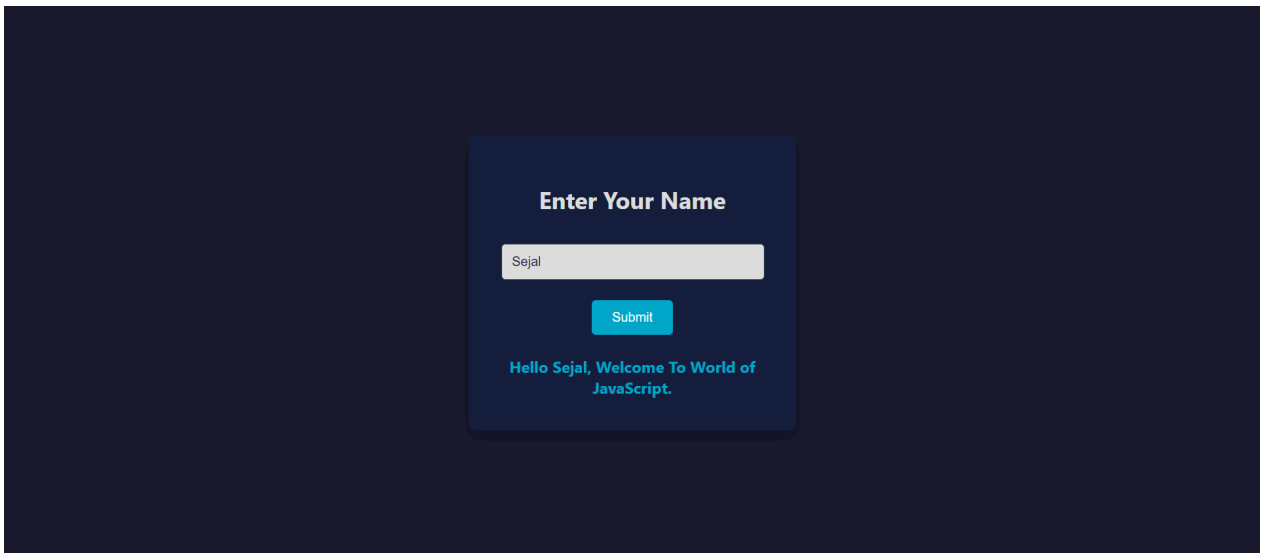
```
    border: none;
    border-radius: 4px;
    cursor: pointer;
    background-color: #ecf0f1;
    color: #2c3e50;
}

#result-container {
    margin-top: 20px;
}

#result-container h2 {
    margin-bottom: 10px;
    color: #ecf0f1;
}

#result {
    font-size: 1.2em;
    font-weight: bold;
    color: #ecf0f1;
```

```
}

</style>



<script>

    function performOperation() {

    const value1 = parseFloat(document.getElementById('value1').value);

    const value2 = parseFloat(document.getElementById('value2').value);

    const operation = document.getElementById('operationSelect').value;

    let result;



    if (isNaN(value1) || isNaN(value2)) {

        result = 'Please enter valid numbers';

    } else {

        switch (operation) {

            case 'add':

                result = value1 + value2;

                break;

            case 'sub':

                result = value1 - value2;

                break;
```

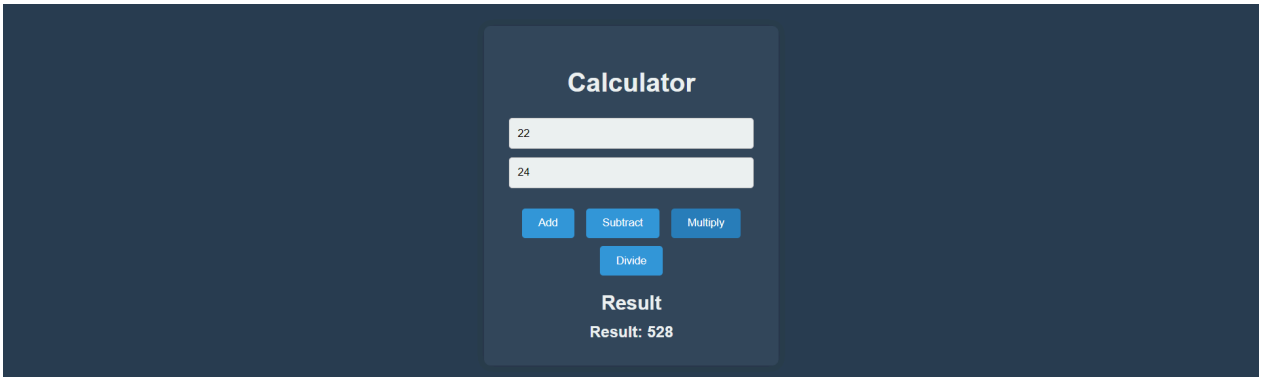
```
            case 'mul':
                result = value1 * value2;
                break;
            case 'div':
                if (value2 === 0) {
                    result = 'Cannot divide by zero';
                } else {
                    result = value1 / value2;
                }
                break;
        }
    }
    document.getElementById('result').textContent = `Result: ${result}`;
}



</script>
```

Gui (ScreenShots)

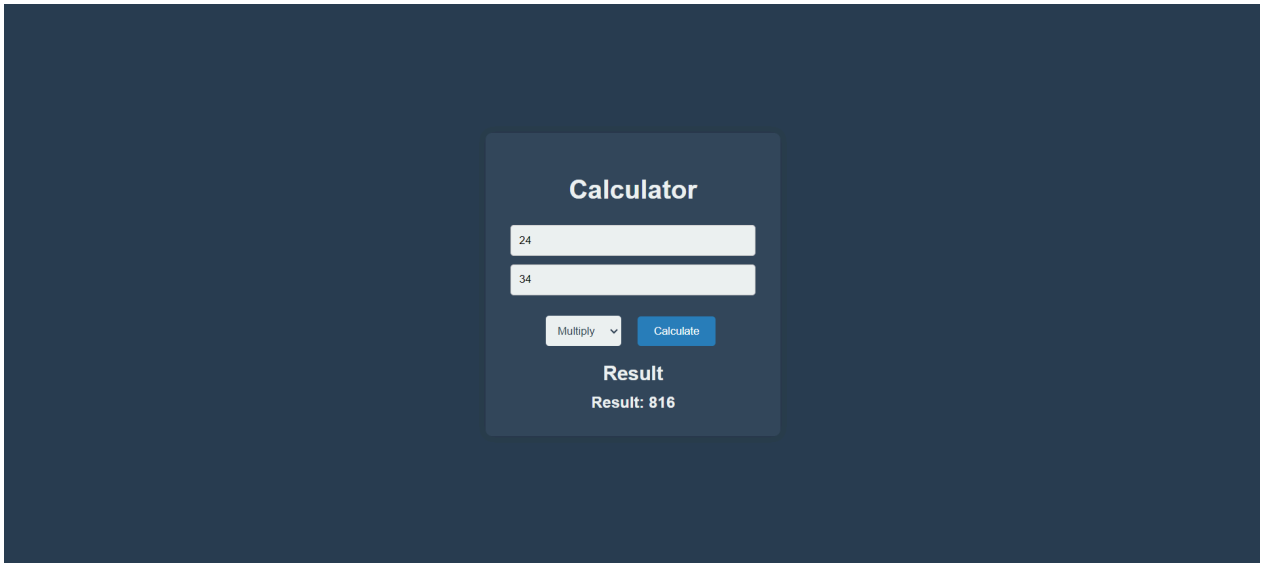
1.



2. Part i



Part ii



Conclusion

In conclusion, the development of JavaScript code for personalized greetings and arithmetic operations enhances user interaction on a webpage. The personalized greeting feature allows for dynamic and engaging user experiences by customizing messages based on input. Meanwhile, the arithmetic operations functionality offers users a simple interface to perform and view results for various calculations. By implementing these features, the webpage not only becomes more interactive but also demonstrates the practical applications of JavaScript in handling user inputs and performing real-time operations.

Assignment No.4

Aim

To explore arrays and its functionality in JavaScript.

Objective

a] Create one button on every click of button different colors should be applied to the Background. Create one button on every click of button different images should be applied.
b] Declare a Javascript String array of colors say colors = [“Red”, “Green”, “Blue”] Accept a value from the user and add it to the array if the value is not present in the array.
c] Write a JavaScript code to apply font color and background color to Heading from dropdown, if font color and background color is same no changes should be reflected.
d] Create an array using JavaScript and display the occurrences of a specific character [For example; arr = [‘a, ‘b’, ‘a’, ‘c’, ‘z’] Output should be occurrences of a is 2]

Theory:

JavaScript and web development concepts, focusing on dynamic manipulation of the DOM (Document Object Model), user interactions, and basic array operations:

1. Event Handling

Event handling allows developers to listen for and respond to user actions, such as clicking buttons. In JavaScript, the `addEventListener()` function is used to attach event listeners to DOM elements. For example, buttons in the examples trigger functions on a 'click' event to change the background color, image, or perform other actions.

2. CSS for Layout and Styling

CSS is crucial for styling elements and arranging them on the page. In these examples, CSS is used to center elements using properties like `display: flex`, `justify-content: center`, and `align-items: center`. This combination ensures that buttons or other elements are horizontally and vertically aligned, regardless of screen size. The `background-size: cover` and `background-position: center` properties help manage how background images are displayed.

3. Arrays and Array Manipulation

Arrays are a fundamental data structure in JavaScript used to store lists of items, such as colors or images. Array operations like `push()` are used to add elements, and the `includes()` method checks whether an item already exists. Cycling through arrays is managed using basic logic, such as incrementing an index and using the modulo operator (`%`) to loop back to the start of the array

when the end is reached.

4. DOM Manipulation

DOM manipulation allows developers to dynamically change elements on a webpage. JavaScript is used to modify styles or properties of elements by accessing them through `document.getElementById()` and then adjusting attributes such as `style.backgroundColor` or `style.backgroundImage`. This approach allows real-time updates to the page based on user input or actions, like changing colors or backgrounds when a button is clicked.

5. Conditional Logic

Conditional logic is used to control behavior based on certain conditions. For example, checking if a value exists in an array before adding it prevents duplicates. In another case, the font color and background color of an element are only applied if they are not the same. These conditions help ensure that actions are only performed under the right circumstances.

6. CSS Transitions and Animations

For smoother visual updates, CSS transitions (like `transition: background-image 0.5s ease-in-out`) are used to animate changes. This makes interactions feel more fluid and provides a better user experience.

7. Input Handling

User input is handled through form elements, such as text input fields, where the value entered by the user is captured and processed. This is a basic yet important feature in interactive applications, where user inputs can trigger dynamic changes, such as adding new colors to an array.

Code

A] i. Background color change

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Background Color</title>
    <style>
        body {
            margin: 0;
```

```
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            text-align: center;
            font-family: Arial, sans-serif;
        }

        button {
            padding: 10px 20px;
            font-size: 16px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <button id="colorButton">Change Background Color</button>

    <script>
        const colors = ["Red", "Green", "Blue", "Yellow", "Pink",
"Purple"];
        let colorIndex = 0;

        document.getElementById("colorButton").addEventListener("click",
function () {
            document.body.style.backgroundColor = colors[colorIndex];
            colorIndex = (colorIndex + 1) % colors.length;
        });
    </script>

</body>
</html>
```

A] ii.

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Background Image</title>
    <style>
        body {
            margin: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            text-align: center;
            font-family: Arial, sans-serif;
            background-size: cover;
            background-position: center;
            transition: background-image 0.5s ease-in-out;
        }

        button {
            padding: 10px 20px;
            font-size: 16px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
        }

        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>

    <button id="imageButton">Change Background Image</button>
```

```
    <script>
        const images = [

"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTUFLz6HHxmtX7fnhxjlXqIcYnLy1x84kq44w&s",

"https://buffer.com/cdn-cgi/image/w=1000,fit=contain,q=90,f=auto/library/content/images/size/w1200/2023/10/free-images.jpg",

"https://letsenhance.io/static/8f5e523ee6b2479e26ecc91b9c25261e/1015f/MainAfter.jpg"
        ];
        let imageIndex = 0;

        document.getElementById("imageButton").addEventListener("click", function () {
            document.body.style.backgroundImage = `url('${images[imageIndex]}')`;
            imageIndex = (imageIndex + 1) % images.length;
        });
    </script>

</body>
</html>
```

B]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Add to Array</title>
</head>
<body>
```

```
    <input type="text" id="colorInput" placeholder="Enter a color">
    <button id="addColorButton">Add Color</button>

    <script>
        let colors = ["Red", "Green", "Blue"];


document.getElementById("addColorButton").addEventListener("click",
function () {
            const color =
document.getElementById("colorInput").value.trim();
            if (color && !colors.includes(color)) {
                colors.push(color);
                alert(`Color ${color} added to the array.`);
            } else {
                alert(`Color ${color} is already in the array.`);
            }
            console.log(colors);
        });
    </script>

</body>
</html>
```

C|

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Font and Background Color</title>
</head>
<body>

    <h1 id="heading">This is a Heading</h1>
```

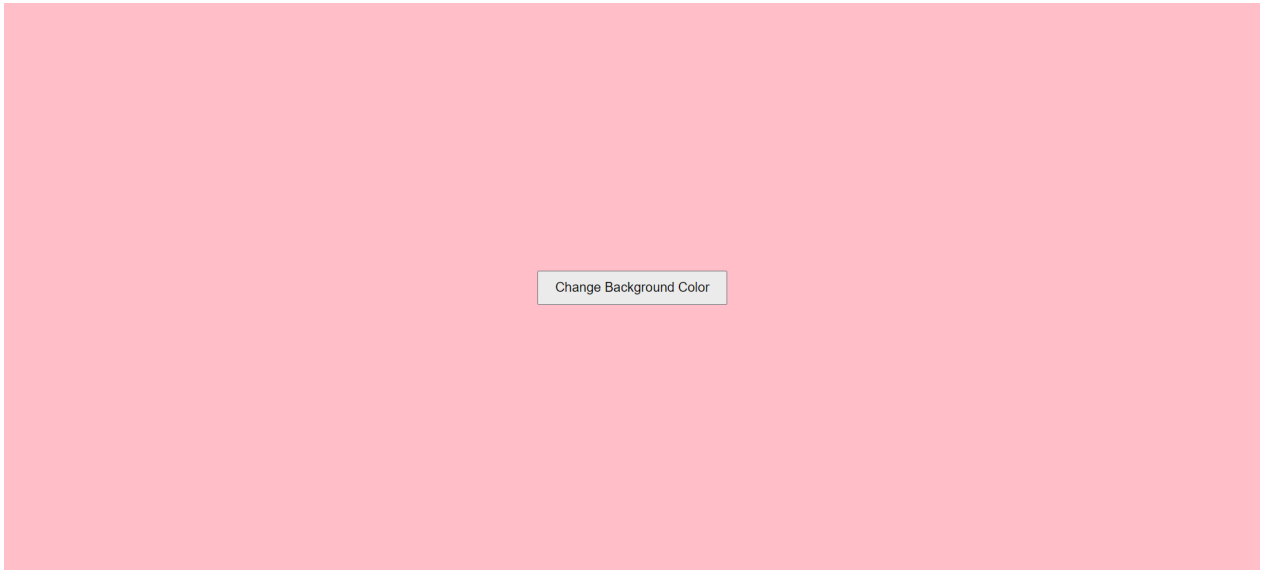
```
    <label for="fontColor">Font Color:</label>
    <select id="fontColor">
        <option value="Red">Red</option>
        <option value="Green">Green</option>
        <option value="Blue">Blue</option>
    </select>

    <label for="bgColor">Background Color:</label>
    <select id="bgColor">
        <option value="Red">Red</option>
        <option value="Green">Green</option>
        <option value="Blue">Blue</option>
    </select>

    <button id="applyButton">Apply Colors</button>

    <script>
        document.getElementById("applyButton").addEventListener("click",
function () {
            const fontColor = document.getElementById("fontColor").value;
            const bgColor = document.getElementById("bgColor").value;

            if (fontColor !== bgColor) {
                document.getElementById("heading").style.color =
fontColor;
                document.getElementById("heading").style.backgroundColor =
bgColor;
            } else {
                alert("Font color and background color should not be the
same.");
            }
        });
    </script>

</body>
</html>
```

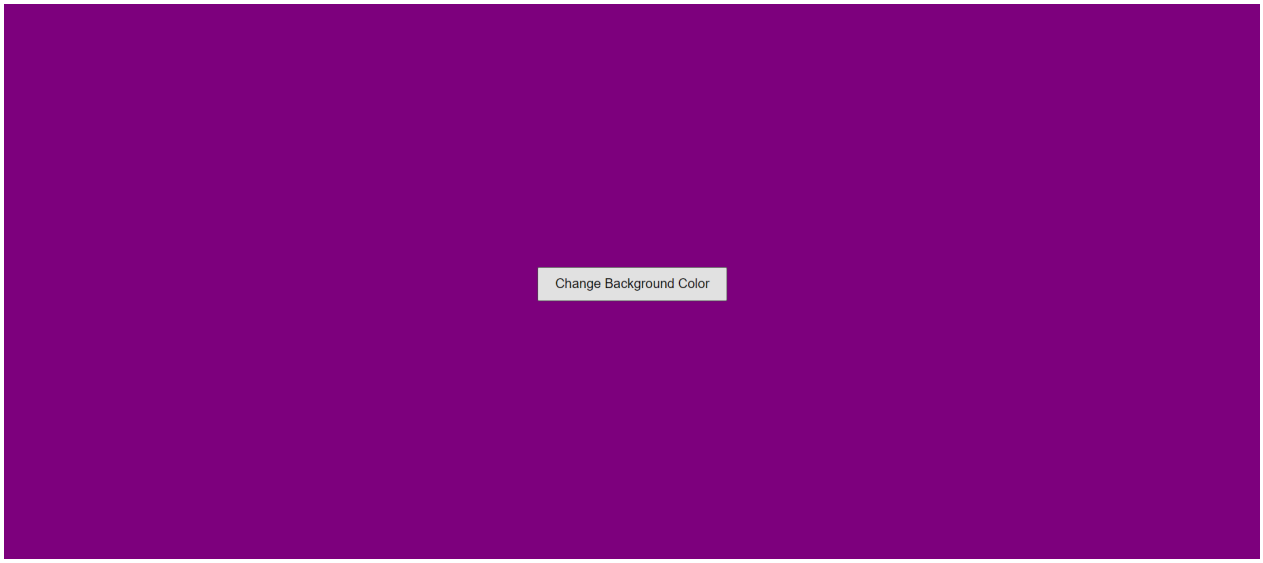
D]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Character Occurrences</title>
</head>
<body>

    <input type="text" id="charInput" placeholder="Enter a character">
    <button id="countButton">Count Occurrences</button>
    <p id="result"></p>

    <script>
        const arr = ['a', 'b', 'a', 'c', 'z'];

        document.getElementById("countButton").addEventListener("click",
function () {
            const char =
document.getElementById("charInput").value.trim();
            if (char.length === 1) {
                const count = arr.filter(c => c === char).length;
                document.getElementById("result").textContent =
`Occurrences of '${char}' is ${count}`;
            } else {
                alert("Please enter a single character.");
            }
        });
    </script>

</body>
</html>
```

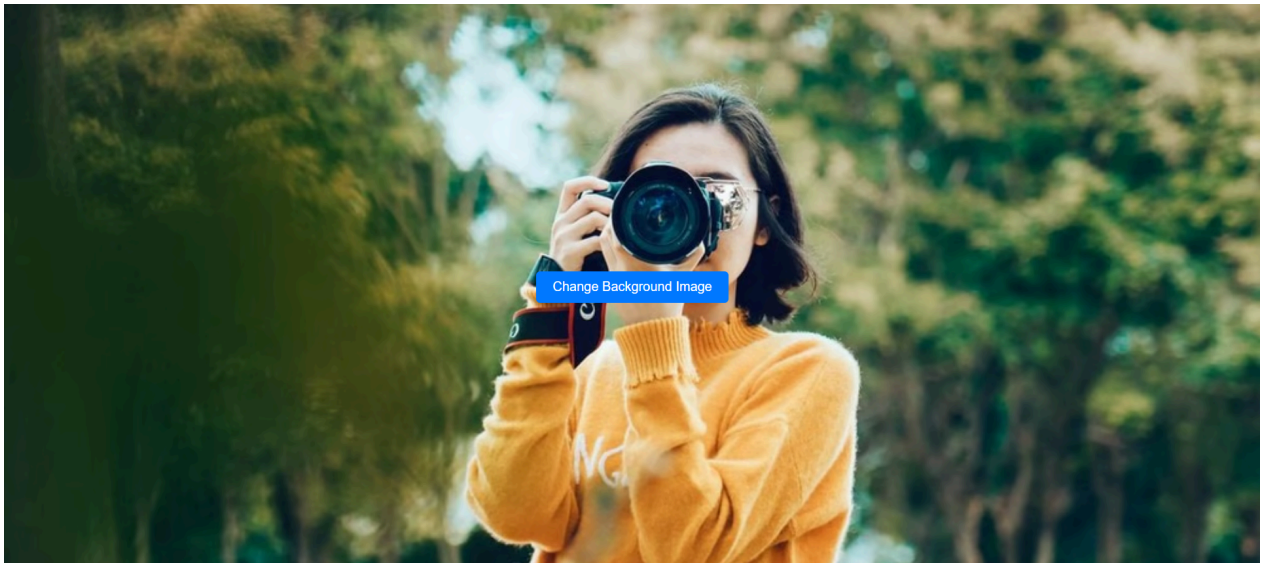
Gui (ScreenShots)

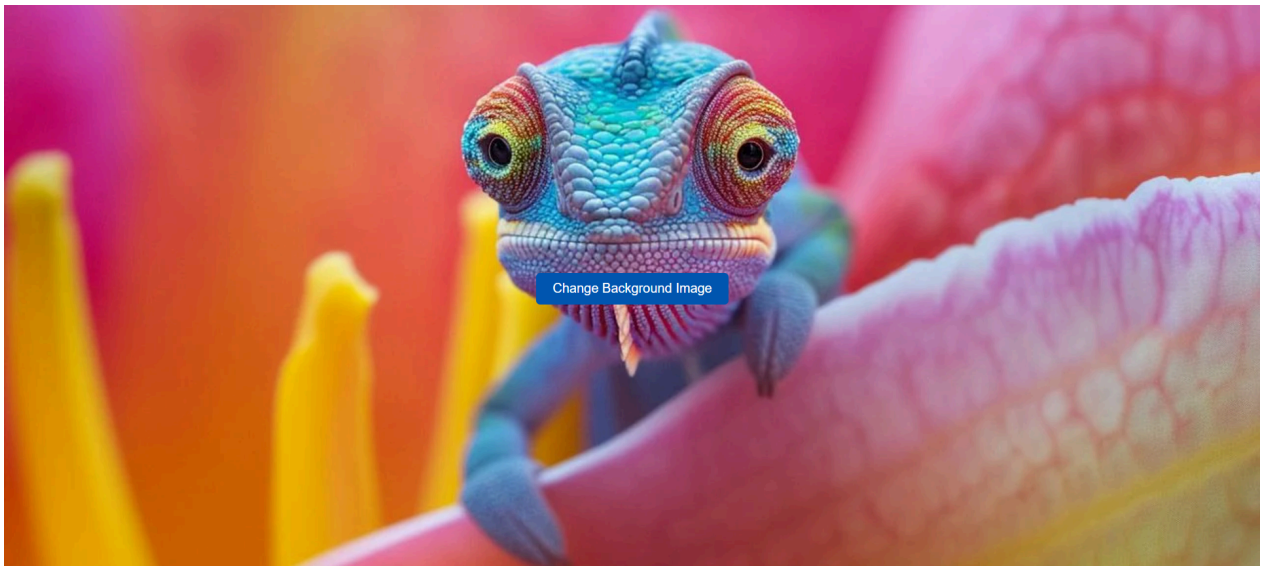
A] i.



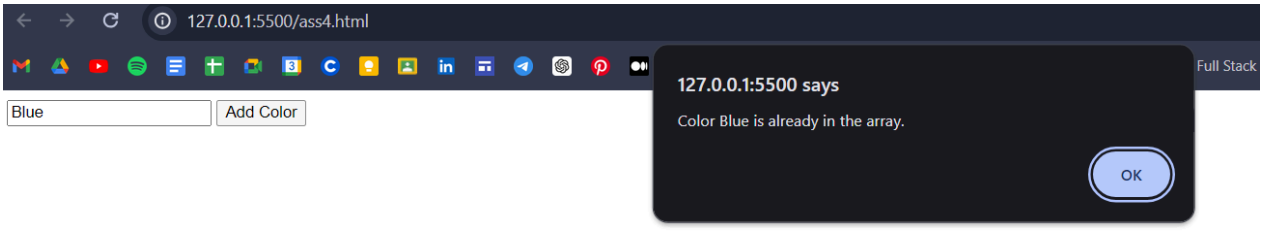


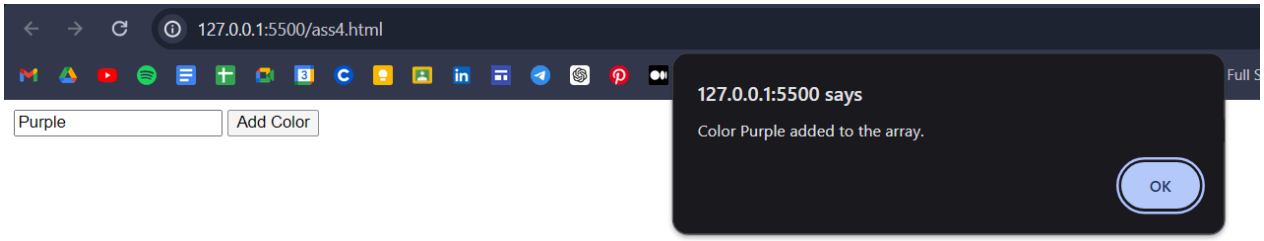
A] ii.


Change Background Image

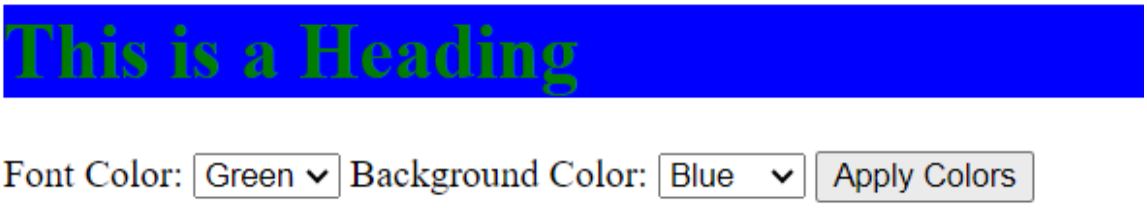

Change Background Image

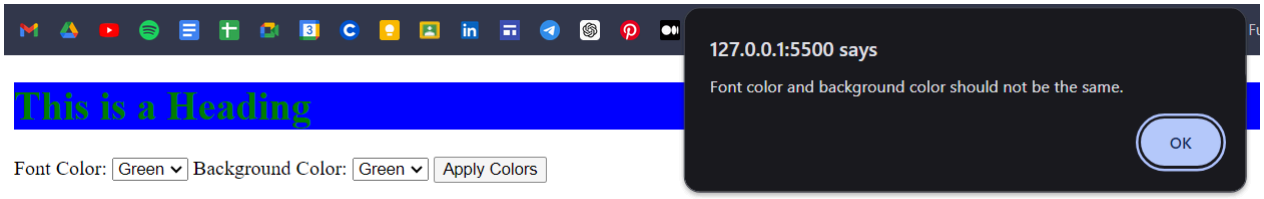
B]


127.0.0.1:5500/ass4.html
Full Stack
Blue
Add Color
127.0.0.1:5500 says
Color Blue is already in the array.
OK


127.0.0.1:5500/ass4.html
Purple
Add Color
127.0.0.1:5500 says
Color Purple added to the array.
OK

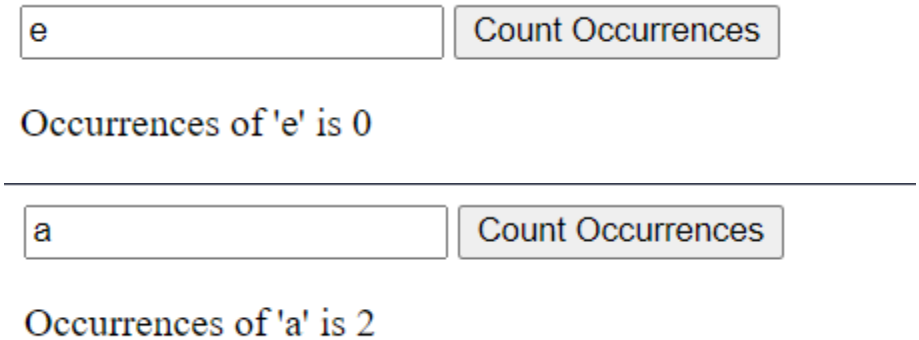
C]


This is a Heading
Font Color: Green
Background Color: Blue
Apply Colors


127.0.0.1:5500 says
Font color and background color should not be the same.
OK
This is a Heading
Font Color: Green
Background Color: Green
Apply Colors


This is a Heading
Font Color: Red
Background Color: Green
Apply Colors

D]


e
Count Occurrences
Occurrences of 'e' is 0
a
Count Occurrences
Occurrences of 'a' is 2

Conclusion

In conclusion, a company registration form is a fundamental part of any web application that aims to onboard users effectively. For Ember and Vine, the registration form is more than just a data collection tool; it's a reflection of the brand's commitment to providing a seamless and engaging user experience. By incorporating all the features of the `<form>` tag and applying thoughtful CSS styling, the form becomes a powerful instrument in the company's digital presence. Ensuring compatibility across different browsers further emphasizes the company's dedication to accessibility and customer satisfaction.

Assignment No.5

Aim

To implement interactive JavaScript functionalities that enhance user experience, including appending rows to a table, dynamically creating anchor tags, changing button colors based on user actions, and modifying the page background color through list-based array interactions.

Objective

A] Write a Javascript code to append static row to the table on click of button
B] Write a Javascript code to append anchor tag that redirects to the new page
C] Write a Javascript code to change the color of button with following events
    1.Onmouseover
    2.Onmouseout
    3.Onclick
D] Write a Javascript code to change the background color of page onclick of array elements displayed as a list.

Theory:

HTML (the Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the structure of the page, CSS the (visual and aural) layout, for a variety of devices. Along with graphics and scripting, HTML and CSS are the basis of building Web pages and Web Applications. Learn more below about:

Code

A]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Append Static Row</title>
    <style>
        table {
            width: 50%;
            margin-top: 20px;
            border-collapse: collapse;
        }
        table, th, td {
```

```
            border: 1px solid black;
            padding: 8px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>

    <h1>Append Static Row to Table</h1>
    <button onclick="appendRow()">Add Row</button>

    <table id="myTable">
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Country</th>
        </tr>
    </table>

    <script>
        function appendRow() {
            var table = document.getElementById("myTable");
            var row = table.insertRow();
            var cell1 = row.insertCell(0);
            var cell2 = row.insertCell(1);
            var cell3 = row.insertCell(2);
            cell1.innerHTML = "John Doe";
            cell2.innerHTML = "30";
            cell3.innerHTML = "USA";
        }
    </script>

</body>
</html>
```

B]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Append Anchor Tag</title>
</head>
<body>

    <h1>Append Anchor Tag to the Page</h1>
    <button onclick="appendAnchor()">Add Link</button>

    <div id="linkContainer"></div>

    <script>
        function appendAnchor() {
            var div = document.getElementById("linkContainer");
            var anchor = document.createElement("a");
            anchor.href = "https://www.example.com";
            anchor.textContent = "Go to Example.com";
            anchor.target = "_blank";
            div.appendChild(anchor);
        }
    </script>

</body>
</html>
```

C]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Button Color on Events</title>
```

```
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f4f4f4;
            padding: 20px;
        }

        .button-container {
            margin-top: 20px;
        }

        button {
            padding: 15px 30px;
            font-size: 16px;
            margin: 10px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }

        .hover-button {
            background-color: #3498db;
            color: white;
        }

        .mouseout-button {
            background-color: #e74c3c;
            color: white;
        }

        .click-button {
            background-color: #2ecc71;
            color: white;
        }

        button:hover {
```

```
            opacity: 0.9;
        }
    </style>
</head>
<body>

    <h1>Change Button Color on Events</h1>

    <div class="button-container">
        <button class="hover-button"
                onmouseover="changeColorOnHover(this)">Hover over
me</button>

            <button class="mouseout-button"
                onmouseout="changeColorOnMouseOut(this)">Mouse out from
me</button>

        <button class="click-button"
                onclick="changeColorOnClick(this)">Click me</button>
    </div>

    <script>
        function changeColorOnHover(button) {
            button.style.backgroundColor = "#f39c12";
        }

        function changeColorOnMouseOut(button) {
            button.style.backgroundColor = "#8e44ad";
        }

        function changeColorOnClick(button) {
            button.style.backgroundColor = "#e67e22";
        }
    </script>

</body>
</html>
```

D]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Background Color on Click</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            transition: background-color 0.5s ease;
            padding: 20px;
        }

        h1 {
            margin-bottom: 20px;
        }

        ul {
            list-style-type: none;
            padding: 0;
        }

        li {
            padding: 10px;
            margin: 5px 0;
            background-color: #f2f2f2;
            cursor: pointer;
            border-radius: 5px;
            text-align: center;
            width: 200px;
            transition: background-color 0.3s ease;
        }

        li:hover {
            background-color: #ddd;
        }
```

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

    <h1>Click on a Color to Change Background</h1>
    <ul id="colorList"></ul>

    <script>
        const colors = ['blue', 'green', 'coral', 'pink', 'yellow',
'gray'];

        const colorList = document.getElementById('colorList');

        colors.forEach(function(color) {
            let listItem = document.createElement('li');
            listItem.textContent = color;
            listItem.style.backgroundColor = color;
            listItem.onclick = function() {
                document.body.style.backgroundColor = color;
            }
            colorList.appendChild(listItem);
        });
    </script>

</body>
</html>
```

Gui (ScreenShots)

A]

Append Static Row to Table

Add Row

Name	Age	Country
John Doe	30	USA
John Doe	30	USA
John Doe	30	USA
John Doe	30	USA

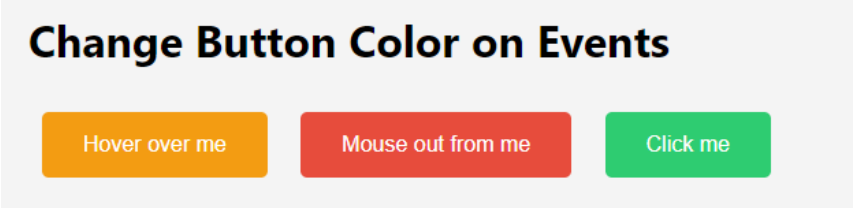
B]

Append Anchor Tag to the Page

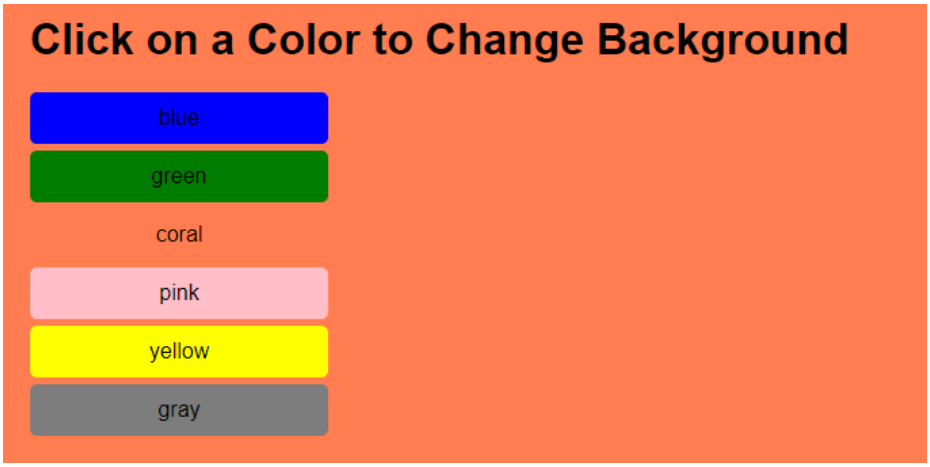
Add Link

Go to Example.com

C]



D]



Conclusion

In this collection of JavaScript exercises, we explored various interactive features for enhancing user interfaces. For (A), we learned how to dynamically append static rows to a table with a button click, enabling easier content updates. In (B), we created a script that appends an anchor tag, offering users quick navigation to a new page. (C) focused on changing button colors based on different user actions such as mouseover, mouseout, and click events, adding interactivity and visual feedback. Lastly, (D) demonstrated how to manipulate the page's background color by clicking on list items, which were dynamically generated from an array. Together, these examples highlight how JavaScript can be used to build responsive and engaging web experiences.

Assignment No.6

Aim

Write a Javascript to create shopping applications which adds the books in cart, updates the existing books, delete the book and display the same. Create proper UI for the same.

Objective:

1. To implement JavaScript functionalities for adding, updating, deleting, and displaying books in a shopping cart.
2. To design a dynamic and responsive user interface for managing the cart operations.
3. To demonstrate the use of DOM manipulation and event handling in JavaScript.
4. To provide users with a seamless experience for managing a shopping cart in a web application.

Theory:

A shopping cart is a crucial feature in e-commerce applications, enabling users to manage their purchases interactively. JavaScript, as a powerful client-side scripting language, can be used to create, manage, and update a cart dynamically.

Key Concepts:

1. **DOM Manipulation**:
   JavaScript provides methods like `getElementById`, `querySelector`, and `createElement` to interact with and modify the web page's elements dynamically. These methods are used to add, update, and remove book entries in the cart.
2. **Event Handling**:
   Events such as clicks, input changes, and form submissions are handled using event listeners to execute the cart's functionalities.
3. **Data Structures**:
   Arrays or objects are utilized to store the book details (title, author, price, and quantity). Operations such as adding, updating, and deleting involve manipulating these data structures.
4. **UI Integration**:
   HTML forms, tables, and buttons are integrated with JavaScript to create a visually appealing interface for interacting with the cart.

Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Shopping Cart - Bookstore</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
      background-color: #f9f9f9;
    }
    h1 {
      text-align: center;
      color: #333;
    }
    .form-container {
      max-width: 600px;
      margin: 20px auto;
      padding: 20px;
      background: #fff;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }
    .form-container input, .form-container button {
      display: block;
      width: 100%;
      margin: 10px 0;
      padding: 10px;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    th, td {
```

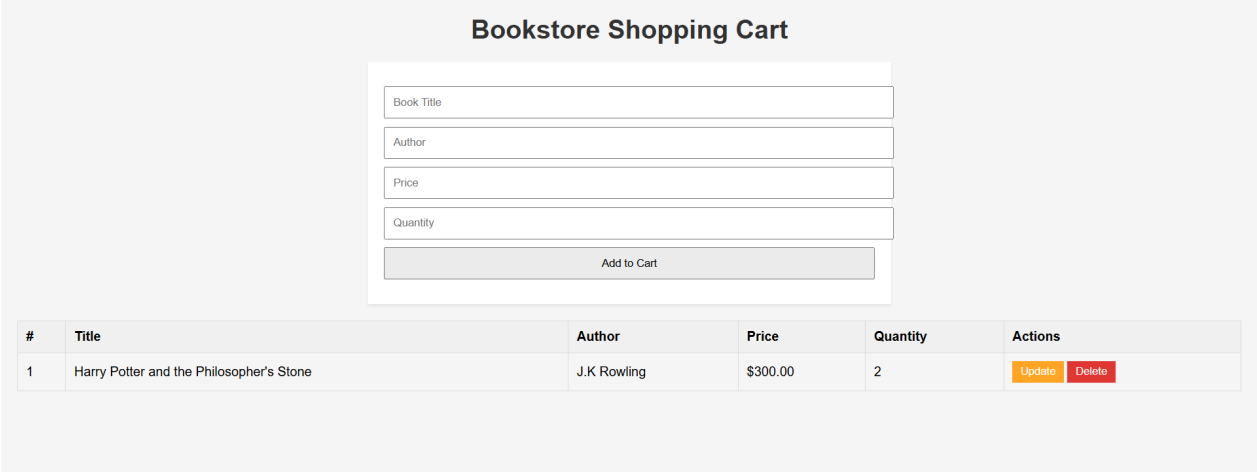
```
      padding: 10px;
      text-align: left;
      border: 1px solid #ddd;
    }
    th {
      background-color: #f4f4f4;
    }
    .btn {
      cursor: pointer;
      color: white;
      border: none;
      padding: 5px 10px;
    }
    .btn-update {
      background-color: #ffa726;
    }
    .btn-delete {
      background-color: #e53935;
    }
  </style>
</head>
<body>

  <h1>Bookstore Shopping Cart</h1>
  <div class="form-container">
    <input type="text" id="bookTitle" placeholder="Book Title">
    <input type="text" id="bookAuthor" placeholder="Author">
    <input type="number" id="bookPrice" placeholder="Price" step="0.01">
    <input type="number" id="bookQuantity" placeholder="Quantity" min="1">
    <button id="addBook">Add to Cart</button>
  </div>

  <table>
    <thead>
      <tr>
        <th>#</th>
        <th>Title</th>
```

```
        <th>Author</th>
        <th>Price</th>
        <th>Quantity</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody id="cartItems"></tbody>
  </table>

  <script src="app.js"></script>

</body>
</html>

<script>
    let cart = [];

function renderCart() {
  const cartItems = document.getElementById('cartItems');
  cartItems.innerHTML = '';

  cart.forEach((item, index) => {
    const row = `
      <tr>
        <td>${index + 1}</td>
        <td>${item.title}</td>
        <td>${item.author}</td>
        <td>$${item.price.toFixed(2)}</td>
        <td>${item.quantity}</td>
        <td>
          <button class="btn btn-update"
onclick="updateBook(${index})">Update</button>
          <button class="btn btn-delete"
onclick="deleteBook(${index})">Delete</button>
        </td>
      </tr>
    `;
```

```
    cartItems.innerHTML += row;
  });
}

function addBook() {
  const title = document.getElementById('bookTitle').value;
  const author = document.getElementById('bookAuthor').value;
  const price = parseFloat(document.getElementById('bookPrice').value);
  const quantity =
parseInt(document.getElementById('bookQuantity').value);

  if (title && author && !isNaN(price) && !isNaN(quantity)) {
    cart.push({ title, author, price, quantity });
    renderCart();
    clearInputs();
  } else {
    alert('Please fill in all fields correctly.');
  }
}

function updateBook(index) {
  const newTitle = prompt('Enter new title:', cart[index].title);
  const newAuthor = prompt('Enter new author:', cart[index].author);
  const newPrice = parseFloat(prompt('Enter new price:',
cart[index].price));
  const newQuantity = parseInt(prompt('Enter new quantity:',
cart[index].quantity));

  if (newTitle && newAuthor && !isNaN(newPrice) && !isNaN(newQuantity)) {
    cart[index] = { title: newTitle, author: newAuthor, price: newPrice,
quantity: newQuantity };
    renderCart();
  } else {
    alert('Invalid input. No changes made.');
  }
}
```

```
function deleteBook(index) {
  if (confirm('Are you sure you want to delete this book?')) {
    cart.splice(index, 1);
    renderCart();
  }
}

function clearInputs() {
  document.getElementById('bookTitle').value = '';
  document.getElementById('bookAuthor

</script>
```

Gui (ScreenShots)



Conclusion

The shopping application successfully demonstrates the use of JavaScript for managing a dynamic shopping cart system. Users can add books to the cart, update their details, delete them, and view the contents of the cart in real time. This assignment highlights the importance of JavaScript in building interactive web applications and reinforces skills in DOM manipulation, event handling, and UI integration.

Assignment No. 7

Aim

To develop a PHP script that accepts a numeric input from the user

Objective

A] Write a PHP script to take a number from the user and print the table of that number.
B] Write a PHP script to display numeric, associative and multidimensional arrays for each loop.

Theory:

PHP Basics: A Brief Theory

1. Introduction to PHP

PHP (Hypertext Preprocessor) is a widely-used, open-source scripting language designed for web development. It is embedded within HTML to create dynamic web pages and can be executed on the server, making it suitable for backend development. PHP is known for its simplicity, flexibility, and compatibility with various databases, including MySQL, PostgreSQL, and SQLite.

2. PHP Syntax and Embedding in HTML

PHP code is typically embedded within HTML using <?php ... ?> tags. When the server processes a PHP file, it executes the PHP code and outputs the result as HTML, which is then displayed in the browser. Here's a basic structure:

```
<!DOCTYPE html>
<html>
<head>
    <title>PHP Basics</title>
</head>
<body>
    <?php
        echo "Hello, World!"; // Prints text to the browser
    ?>
</body>
</html>
```

3. Variables and Data Types

Variables in PHP are defined using the $ symbol and can store different data types, such as strings, integers, floats, arrays, and booleans. PHP is loosely typed, meaning variable types are automatically determined.

```php
<?php
    $name = "Alice";        // String
    $age = 25;              // Integer
    $height = 5.8;          // Float
    $isStudent = true;      // Boolean
?>
```

4. Operators

PHP supports a wide range of operators for arithmetic (+, -, *, /), comparison (==, !=, >, <), logical (&&, ||, !), and assignment (=, +=, -=) operations.

5. Control Structures

PHP includes control structures such as conditionals (if, else, elseif) and loops (for, while, foreach) to manage the flow of execution.

```php
<?php
    $number = 5;
    if ($number > 0) {
        echo "Positive number";
    } else {
        echo "Negative number";
    }
    for ($i = 1; $i <= 5; $i++) {
        echo $i . " ";
    }
?>
```

6. Functions

Functions in PHP help modularize code, making it reusable and organized. PHP has many built-in functions, and custom functions can also be created using the function keyword.

```php
<?php
    function greet($name) {
        return "Hello, " . $name;
    }
    echo greet("Alice");
?>
```

7. Arrays

Arrays in PHP store multiple values in a single variable and come in three types:

- **Numeric Arrays**: Indexed by numbers.
- **Associative Arrays**: Indexed by named keys.
- **Multidimensional Arrays**: Contain arrays within arrays, allowing complex data structures.

```php
<?php
    $fruits = ["apple", "banana", "cherry"]; // Numeric array
    $person = ["name" => "Alice", "age" => 25]; // Associative array
?>
```

8. Form Handling

PHP handles HTML form data using global arrays $_GET and $_POST, allowing data submission through GET (URL parameters) or POST (form data in request body). This is essential for user interaction on web applications.

```php
<form method="post" action="welcome.php">
    Name: <input type="text" name="name">
    <input type="submit">
</form>
```

9. Database Connectivity

PHP integrates seamlessly with databases, particularly MySQL, allowing for storage, retrieval, and manipulation of data. Database connections are managed using MySQLi or PDO (PHP Data Objects) libraries.

10. Conclusion

PHP remains a core technology for building dynamic, data-driven web applications. Its ease of use, extensive library of built-in functions, and wide adoption make it a strong choice for server-side programming.

Code

Part A]

```php
<!DOCTYPE html>
<html>
<head>
    <title>Multiplication Table</title>
</head>
<body>
    <form method="post">
        Enter a number: <input type="number" name="number" required>
        <input type="submit" value="Generate Table">
```

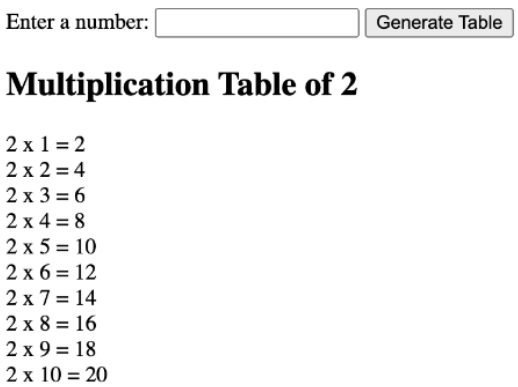
```
    </form>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $number = $_POST["number"];
        echo "<h2>Multiplication Table of $number</h2>";
        for ($i = 1; $i <= 10; $i++) {
            echo "$number x $i = " . ($number * $i) . "<br>";
        }
    }
    ?>
</body>
</html>
```

Part B]

```
<!DOCTYPE html>
<html>
<head>
    <title>Array Types</title>
</head>
<body>
    <?php
    // Numeric array
    $numericArray = [1, 2, 3, 4, 5];
    echo "<h3>Numeric Array</h3>";
    foreach ($numericArray as $value) {
        echo $value . "<br>";
    }

    // Associative array
    $associativeArray = [
        "name" => "Alice",
        "age" => 25,
        "city" => "New York"
    ];
```

```
    echo "<h3>Associative Array</h3>";
    foreach ($associativeArray as $key => $value) {
        echo "$key: $value<br>";
    }

    // Multidimensional array
    $multiArray = [
        ["name" => "Bob", "age" => 30, "city" => "Chicago"],
        ["name" => "John", "age" => 35, "city" => "Los Angeles"],
        ["name" => "Sara", "age" => 28, "city" => "Miami"]
    ];
    echo "<h3>Multidimensional Array</h3>";
    foreach ($multiArray as $person) {
        foreach ($person as $key => $value) {
            echo "$key: $value<br>";
        }
        echo "<br>";
    }
    ?>
</body>
</html>
```

Gui (ScreenShots)



A.

Numeric Array

1
2
3
4
5

Associative Array

name: Alice
age: 25
city: New York

Multidimensional Array

name: Bob
age: 30
city: Chicago

name: John
age: 35
city: Los Angeles

name: Sara
age: 28
city: Miami

B.

Assignment No. 8

Aim :

To design and implement a basic shopping application form in PHP that allows users to add items by providing itemID, itemName, and itemQuantity. The aim is to display a list of added items dynamically using PHP sessions, providing a foundational understanding of session handling, form processing, and data display with HTML and PHP.

Objective

Design a shopping application form with following fields [itemID, itemName, itemQuantity] Write a PHP script to add and display the items.

THEORY:

1. Introduction to PHP Sessions

In PHP, sessions allow data to be stored across multiple pages or form submissions. This is especially useful for applications like a shopping cart, where we need to retain information about added items as the user interacts with the form. Sessions in PHP are initiated using the session_start() function, which creates a unique session for each user. Data can then be stored in a $_SESSION superglobal array.

Key Points:

- Sessions are stored on the server, unlike cookies, which are stored on the client's browser.
- Each user session is unique, allowing data to be preserved across different pages until the session expires or is destroyed.

2. Form Handling in PHP

HTML forms enable data collection from users. In this shopping app, the form collects the item's ID, name, and quantity and then sends it to the server using the POST method.

Form Fields in the Code:

- itemID (text input): The unique identifier for each item.
- itemName (text input): The name of the item being added.
- itemQuantity (number input): The quantity of the item to be purchased.

The POST Method:

- When the form is submitted, it sends data to the server.
- PHP retrieves this data from the $_POST array using the names of each input field ($_POST['itemID'], $_POST['itemName'], and $_POST['itemQuantity']).

3. Storing Items Using Session Arrays

PHP arrays provide a structured way to store multiple values. In this application, an array of items is stored in the $_SESSION variable. Each item is itself an associative array, with keys for itemID, itemName, and itemQuantity. This structure makes it easy to store and retrieve item details efficiently.

Code Example:

```php
$item = [
    "itemID" => $itemID,
    "itemName" => $itemName,
    "itemQuantity" => $itemQuantity
];
$_SESSION['items'][] = $item;
```

Here:

- Each item is added to $_SESSION['items'], which is an array containing all items added to the cart.
- This array remains in memory across page loads due to the session.

4. Displaying Items with an HTML Table

After items are added, the PHP script checks if there are any items in $_SESSION['items']. If items exist, it displays them in an HTML table format.

Table Structure:

- The table consists of rows and columns. Each item is represented by a row, with separate columns for itemID, itemName, and itemQuantity.
- The foreach loop iterates over each item in $_SESSION['items'] to display it in the table.

Code Example:

```php
foreach ($_SESSION['items'] as $item) {
    echo "<tr>";
    echo "<td>" . htmlspecialchars($item['itemID']) . "</td>";
    echo "<td>" . htmlspecialchars($item['itemName']) . "</td>";
    echo "<td>" . htmlspecialchars($item['itemQuantity']) . "</td>";
    echo "</tr>";
}
```

5. HTML and PHP Integration

This code integrates HTML and PHP by embedding PHP code within the HTML file. When the server processes the file, it executes the PHP code and outputs the resulting HTML to the user's browser. This combination enables dynamic content updates based on user interactions, like adding items to a shopping cart and displaying them immediately.

6. Data Validation and Security (optional enhancement)

While this script is basic, it uses the htmlspecialchars() function to prevent XSS (Cross-Site Scripting) attacks by encoding special characters. For production environments, further validation and sanitization would be needed to prevent SQL injection and ensure valid data input.

Summary

This shopping application form demonstrates:

- The use of PHP sessions to retain data across page loads.
- Handling form submissions and storing data in associative arrays.
- Displaying dynamically added content (shopping cart items) in an HTML table format.
- Basic web application functionality where a user can add items, which are then stored temporarily in a session, simulating a simple shopping cart.

This approach is foundational for developing more complex applications with persistent data storage (e.g., using a database).

Code

```
<!DOCTYPE html>

<html>

<head>

   <title>Shopping Application</title>

</head>

<body>

   <h1>Shopping Application</h1>


   <form method="post">

       <label for="itemID">Item ID:</label>

       <input type="text" name="itemID" required><br><br>


       <label for="itemName">Item Name:</label>

       <input type="text" name="itemName" required><br><br>


       <label for="itemQuantity">Item Quantity:</label>
```

```
    <input type="number" name="itemQuantity" required><br><br>

    <input type="submit" value="Add Item">
</form>

<?php
session_start();

if (!isset($_SESSION['items'])) {
    $_SESSION['items'] = [];
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $itemID = $_POST['itemID'];
    $itemName = $_POST['itemName'];
    $itemQuantity = $_POST['itemQuantity'];

    $item = [
        "itemID" => $itemID,
        "itemName" => $itemName,
        "itemQuantity" => $itemQuantity
    ];

    $_SESSION['items'][] = $item;
```

```
    }


    if (!empty($_SESSION['items'])) {
        echo "<h2>Items in Shopping Cart</h2>";
        echo "<table border='1'>";
        echo "<tr><th>Item ID</th><th>Item Name</th><th>Item
Quantity</th></tr>";

        foreach ($_SESSION['items'] as $item) {
            echo "<tr>";
            echo "<td>" . htmlspecialchars($item['itemID']) . "</td>";
            echo "<td>" . htmlspecialchars($item['itemName']) . "</td>";
            echo "<td>" . htmlspecialchars($item['itemQuantity']) .
"</td>";
            echo "</tr>";
        }

        echo "</table>";
    } else {
        echo "<p>No items in the cart.</p>";
    }
    ?>
</body>
</html>
```

Output:

Shopping Application

Item ID:

Item Name:

Item Quantity:

Add Item

Items in Shopping Cart

Item ID	Item Name	Item Quantity
123	okah	2

Assignment No. 9

Aim

To develop a web-based student result display system using PHP and MySQL. This system allows users to filter and view student records based on passing year and class grade (First Class, Second Class, Pass, or All). The aim is to demonstrate dynamic data retrieval from a MySQL database and conditional display based on user-selected criteria, implementing secure database queries and creating a user-friendly interface for data interaction.

Objective

- Create the database table with the entries mentioned:
- [rollNo, studName, studDept, passingYear, classGrades] Grades should be either {First
- Class, Second Class, Pass}
- Design the UI as given below and show the appropriate result through PHP script

Theory:

This project aims to create a student result display system using PHP and MySQL. The system allows users to select a year and grade category to view student records based on specific criteria. Here is an overview of the key concepts and components involved:

1. Introduction to PHP and MySQL

PHP is a widely-used server-side scripting language designed for web development, and it is particularly well-suited for interacting with databases. MySQL is a popular relational database management system that allows data to be stored, managed, and retrieved efficiently. Together, PHP and MySQL provide a powerful framework for developing dynamic, data-driven web applications.

In this project, PHP is used to process form inputs, query the MySQL database, and dynamically generate HTML output based on the query results.

2. Database Design

In this system, a MySQL database (student_db) is created with a table (students) to store student records. The table has the following fields:

- rollNo: A unique identifier for each student.
- studName: The student's name.
- studDept: The department or course of the student.
- passingYear: The year the student graduated.
- classGrades: The grade classification for the student, which is stored as an ENUM type with values 'First Class', 'Second Class', and 'Pass'.

This structure enables easy retrieval of student records based on specific criteria, such as passing year and

grade.

3. HTML Form Design

The user interface (UI) includes an HTML form with:

- A dropdown for selecting the passingYear.
- Radio buttons for selecting the classGrades filter (First Class, Second Class, Pass, or All).
- A submit button labeled “Display” to initiate the query.

When the user submits the form, the selected year and classGrades values are sent to the PHP script for processing.

4. Form Handling and Data Retrieval in PHP

When the form is submitted, the PHP script captures the form inputs using the $_POST superglobal array. The passingYear and classGrades values are then used to filter records in the database.

Steps for Data Processing:

- Base Query: The PHP script starts with a base SQL query to retrieve records matching the selected passingYear.
- Conditional Query Modification: If a specific classGrades is selected (e.g., ‘First Class’ or ‘Second Class’), an additional condition is added to the SQL query to filter by classGrades. If “All” is selected, only the passingYear filter is applied.

5. Using Prepared Statements for Security

Prepared statements in PHP are used to prevent SQL injection attacks, a common security vulnerability in web applications. SQL injection occurs when attackers manipulate query parameters to execute unauthorized SQL commands. By using placeholders (?) in SQL queries and binding parameters with bind_param(), prepared statements ensure that user inputs are treated as data, not as executable SQL code.

For example, the query in this project is structured like this:

```
$stmt = $conn->prepare("SELECT rollNo, studName, studDept, passingYear,
classGrades FROM students WHERE passingYear = ?");
```

If a specific grade is selected, the query is modified as:

```
$stmt = $conn->prepare("SELECT rollNo, studName, studDept, passingYear,
classGrades FROM students WHERE passingYear = ? AND classGrades = ?");
```

6. Displaying Query Results in HTML

Once the query is executed, the results are fetched and displayed in an HTML table. Each row of the table corresponds to a student record with columns for rollNo, studName, studDept, passingYear, and classGrades. This structure makes the information easy to read and accessible for the user.

The table generation uses PHP loops to iterate over the fetched results and dynamically create table rows with the data.

7. Data Security and Display

To prevent XSS (Cross-Site Scripting) attacks, the htmlspecialchars() function is used to sanitize the output. This function converts special characters to HTML entities, preventing malicious code from being injected and executed in the browser. For instance:

```
  echo "<td>" . htmlspecialchars($row['studName']) . "</td>";
```

This ensures that data is displayed securely in the browser without the risk of unintended JavaScript execution.

8. Summary of Workflow

1. The user selects a year and grade from the form.
2. Upon form submission, PHP captures the inputs and constructs an SQL query based on the selected filters.
3. A prepared statement executes the query, securely fetching records that match the criteria.
4. The results are displayed in an HTML table format, with appropriate sanitization applied to prevent security risks.

9. Benefits and Applications

This project showcases a simple, secure method for developing dynamic, data-driven web applications. The concepts used here—such as session handling, form processing, prepared statements, and data sanitization—are foundational skills for building secure PHP applications. The project can be expanded to add features like pagination, more detailed filtering, or integration with other databases or frameworks.

This system can serve as the basis for a more comprehensive student management system, enabling educational institutions to manage and display student data securely and efficiently.

Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Student Result Display</title>
</head>
<body>

<h1>Student Result Display</h1>

<form method="post" action="student_display.php">
   <label for="year">Years:</label>
   <select name="year" id="year">
       <option value="2014">2014</option>
       <option value="2015">2015</option>
       <option value="2016">2016</option>
       <option value="2017">2017</option>
       <option value="2018">2018</option>
       <option value="2019">2019</option>
       <option value="2020">2020</option>
```

```
    </select><br><br>

    <input type="radio" name="classGrade" value="First Class"
id="firstClass">
    <label for="firstClass">First Class</label><br>

    <input type="radio" name="classGrade" value="Second Class"
id="secondClass">
    <label for="secondClass">Second Class</label><br>

    <input type="radio" name="classGrade" value="Pass" id="pass">
    <label for="pass">Pass</label><br>

    <input type="radio" name="classGrade" value="All" id="all" checked>
    <label for="all">All</label><br><br>

    <input type="submit" name="display" value="Display">
</form>

<?php
$conn = new mysqli("localhost", "root", "", "student_db");

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['display'])) {
    $year = $_POST['year'];
    $classGrade = $_POST['classGrade'];

    $sql = "SELECT rollNo, studName, studDept, passingYear, classGrades
FROM students WHERE passingYear = ?";

    if ($classGrade != "All") {
        $sql .= " AND classGrades = ?";
    }
```

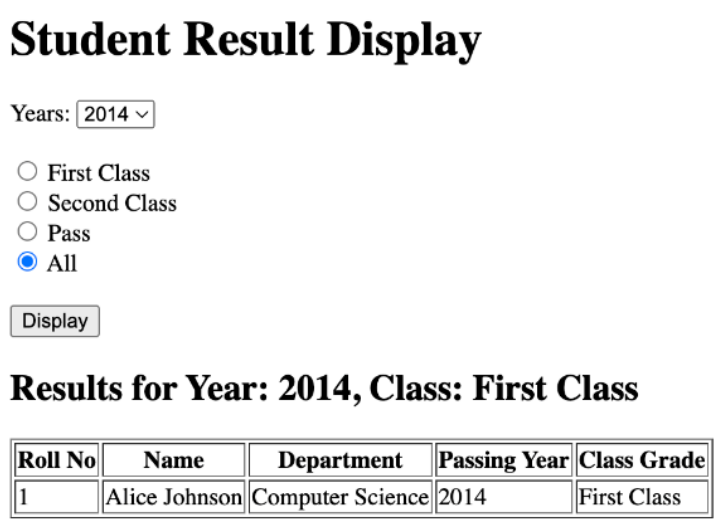
```
    $stmt = $conn->prepare($sql);
    if ($classGrade == "All") {
        $stmt->bind_param("i", $year);
    } else {
        $stmt->bind_param("is", $year, $classGrade);
    }

    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        echo "<h2>Results for Year: $year, Class: $classGrade</h2>";
        echo "<table border='1'>";
        echo "<tr><th>Roll
No</th><th>Name</th><th>Department</th><th>Passing Year</th><th>Class
Grade</th></tr>";

        while ($row = $result->fetch_assoc()) {
            echo "<tr>";
            echo "<td>" . htmlspecialchars($row['rollNo']) . "</td>";
            echo "<td>" . htmlspecialchars($row['studName']) . "</td>";
            echo "<td>" . htmlspecialchars($row['studDept']) . "</td>";
            echo "<td>" . htmlspecialchars($row['passingYear']) . "</td>";
            echo "<td>" . htmlspecialchars($row['classGrades']) . "</td>";
            echo "</tr>";
        }
        echo "</table>";
    } else {
        echo "<p>No results found for the selected criteria.</p>";
    }

    $stmt->close();
}

$conn->close();
?>
```

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

SQL QUERY:

```
CREATE TABLE students (
    rollNo INT PRIMARY KEY,
    studName VARCHAR(50),
    studDept VARCHAR(50),
    passingYear YEAR,
    classGrades ENUM('First Class', 'Second Class', 'Pass')
);
INSERT INTO students (rollNo, studName, studDept, passingYear, classGrades) VALUES
(1, 'Alice Johnson', 'Computer Science', 2014, 'First Class'),
(2, 'Bob Smith', 'Mechanical Engineering', 2014, 'Second Class'),
(3, 'Carol Williams', 'Electrical Engineering', 2015, 'Pass'),
(4, 'David Brown', 'Computer Science', 2015, 'First Class');
```

Output:



Assignment No. 10

Aim:
To create a simple web application that demonstrates the use of jQuery for adding animations to HTML elements. The aim is to animate a element by changing its size and opacity upon the click of a button, showcasing jQuery's ability to manipulate CSS properties dynamically and enhance user interactivity on a web page. This project introduces the basics of jQuery animations and event handling, allowing for smooth transitions and visual effects.

Objective

Write a JQuery code to give animation to div on click of button.

Theory:

jQuery is a popular JavaScript library that simplifies HTML document traversal, manipulation, and event handling. One of its powerful features is the ability to create smooth animations with minimal code. This project demonstrates how to use jQuery to animate a element when a button is clicked, providing a practical example of event-driven programming and animation control on a webpage.

1. Introduction to jQuery

jQuery was designed to simplify JavaScript coding, providing a more concise and flexible syntax for handling various tasks like event handling, DOM manipulation, and AJAX requests. Its syntax is simple, allowing developers to "write less, do more." By including the jQuery library, we gain access to numerous functions that make it easier to handle complex JavaScript tasks, such as animations and asynchronous operations.
Key Features of jQuery:

- Simplified syntax for common JavaScript operations
- Cross-browser compatibility
- Built-in functions for animations and effects
- Robust event handling

2. Event Handling in jQuery

An event is an action or occurrence detected by the browser, such as a user clicking a button or submitting a form. jQuery simplifies event handling by allowing developers to attach event listeners to elements easily. In this project, we use the click() event to detect when the user clicks the "Animate Div" button, triggering the animation of the element.
In jQuery, event handlers can be added to elements using syntax like:

```
$("#elementID").event(function() {
   // Code to execute when the event occurs
});
```

In our example, $("#animateButton").click(function() { ... }); adds a click event listener to the button with ID animateButton. When this button is clicked, the specified animation for the element is executed.

3. Using jQuery animate() Function

The animate() function in jQuery is a versatile method for creating animations by changing CSS properties over time. It allows you to define which CSS properties to animate, how long the animation should last, and an optional callback function to execute once the animation completes.

Syntax of animate():

```
$(selector).animate({ properties }, duration, callback);
```

- Properties: An object that defines the CSS properties to animate and their target values. For example, { width: "300px", height: "300px", opacity: 0.5 } specifies that the width and height should change to 300px and the opacity should be reduced to 0.5.
- Duration: Specifies how long the animation should take, in milliseconds. For example, 1000 represents 1 second.
- Callback: An optional function that executes once the animation completes. This can be used for further animations or actions after the initial animation.

In this project:

- When the button is clicked, the animate() function increases the 's width, height, and decreases its opacity, creating a "growing and fading" effect.
- The callback function then reverses these properties to return the to its original size and opacity, creating a looping animation effect.

4. CSS Properties and jQuery Animations

jQuery allows certain CSS properties to be animated. Commonly used properties include:

- width and height: Changes the dimensions of the element.
- opacity: Controls the transparency of the element (0 is fully transparent, 1 is fully opaque).
- margin, padding, and position properties can also be animated.

In our example, we animate the width, height, and opacity properties of the , creating a smooth transition that expands and fades the element.

5. Callback Functions

The callback function in jQuery is a function that executes after an animation completes. This is useful for creating sequential animations or performing additional actions once the animation is done. In this project, the callback function reverses the animation by returning the to its original size and opacity.

This step-by-step animation effect demonstrates a form of control flow in animations, where actions can occur in a specific sequence rather than simultaneously.

6. Benefits and Applications of jQuery Animations

Using jQuery for animations provides the following advantages:

- Smooth Transitions: jQuery animations create visually appealing transitions that improve user experience.
- Minimal Code: jQuery's concise syntax reduces the amount of code needed to achieve complex effects.
- Cross-Browser Compatibility: jQuery animations work consistently across different browsers, simplifying compatibility issues.
- Event-Driven Interactivity: jQuery animations can be triggered by various events, enhancing interactivity and user engagement on web pages.

Applications:

- Creating responsive, interactive elements on webpages, such as expanding menus, image sliders, or modal pop-ups.
- Animating elements to improve the user experience, drawing attention to important sections or buttons.

Summary

This project introduces the basics of jQuery animations by creating an interactive effect on a element. The button click event triggers the animation, changing the size and opacity of the element. By exploring the animate() function and event handling in jQuery, we can understand how to create dynamic and responsive animations with minimal code. This knowledge provides a foundation for building more complex animations and interactive features on modern web applications.

Code

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>jQuery Animation Example</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
cript>
   <style>
       #animateDiv {
           width: 100px;
           height: 100px;
           background-color: lightblue;
           margin-top: 20px;
       }
```

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


    <button id="animateButton">Animate Div</button>
    <div id="animateDiv"></div>


    <script src="script.js"></script>



</body>
</html>

$(document).ready(function(){
    $("#animateButton").click(function(){
        $("#animateDiv").animate({
            width: "300px",
            height: "300px",
            opacity: 0.5,
        }, 1000, function(){
            $(this).animate({
                width: "100px",
                height: "100px",
                opacity: 1.0,
            }, 1000);
        });
    });
});
```

Gui (ScreenShots)