WTL Extra Assignments

HTML + CSS

1. Registration Page for Email Website

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Email Sign-Up Page</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
        }

        .container {
            background-color: #fff;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.2);
            max-width: 400px;
            width: 100%;
        }

        h2 {
            text-align: center;
            color: #333;
        }

        .form-group {
            margin-bottom: 20px;
        }
```

```css
label {
    display: block;
    font-weight: bold;
}

input[type="text"],
input[type="email"],
input[type="password"] {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    transition: border-color 0.3s ease;
}

input[type="text"]:focus,
input[type="email"]:focus,
input[type="password"]:focus {
    border-color: #007BFF;
}

button[type="submit"] {
    background-color: #007BFF;
    color: #fff;
    border: none;
    border-radius: 5px;
    padding: 10px 20px;
    cursor: pointer;
    font-weight: bold;
    transition: background-color 0.3s ease;
    width: 100%;
}

button[type="submit"]:hover {
    background-color: #0056b3;
}

/* Add a subtle animation */
@keyframes fadeIn {
    from {
        opacity: 0;
    }
```

```
            to {
                opacity: 1;
            }
        }

        .container {
            animation: fadeIn 0.5s;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Email Sign-Up</h2>
        <form action="#" method="post">
            <div class="form-group">
                <label for="firstName">First Name</label>
                <input type="text" id="firstName" name="firstName" required>
            </div>
            <div class="form-group">
                <label for="lastName">Last Name</label>
                <input type="text" id="lastName" name="lastName" required>
            </div>
            <div class="form-group">
                <label for="email">Email Address</label>
                <input type="email" id="email" name="email" required>
            </div>
            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" id="password" name="password" required>
            </div>
            <div class="form-group">
                <label for="confirmPassword">Confirm Password</label>
                <input type="password" id="confirmPassword"
name="confirmPassword" required>
            </div>
            <button type="submit">Sign Up</button>
        </form>
    </div>
</body>
</html>
```

2. Registration Page for Social Networking Websites like facebook, Instagram, etc.

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Facebook Registration</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 0;
        }

        .container {
            background-color: #fff;
            max-width: 500px;
            margin: 0 auto;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }

        h2 {
            text-align: center;
            color: #1877f2; /* Facebook Blue Color */
        }

        .form-group {
            margin-bottom: 20px;
        }

        label {
            display: block;
            font-weight: bold;
        }

        input[type="text"],
        input[type="email"],
        input[type="password"] {
            width: 100%;
            padding: 10px;
            margin-top: 5px;
            margin-bottom: 10px;
            border: 1px solid #ddd;
            border-radius: 5px;
        }

        button[type="submit"] {
            background-color: #1877f2;
            color: #fff;
            border: none;
            border-radius: 5px;
            padding: 10px 20px;
            cursor: pointer;
            font-weight: bold;
        }

        button[type="submit"]:hover {
            background-color: #0e5a96;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Create a New Account</h2>
        <form action="#" method="post">
            <div class="form-group">
                <label for="firstName">First Name</label>
                <input type="text" id="firstName" name="firstName" required>
            </div>
            <div class="form-group">
                <label for="lastName">Last Name</label>
                <input type="text" id="lastName" name="lastName" required>
            </div>
            <div class="form-group">
                <label for="email">Email or Phone Number</label>
                <input type="text" id="email" name="email" required>
            </div>
            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" id="password" name="password" required>
            </div>
            <button type="submit">Sign Up</button>
        </form>
    </div>
</body>
</html>
```

3. Registration Page for Blood Bank website

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blood Bank Registration</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #ff6b6b;
            text-align: center;
        }
        .container {
            background-color: #fff;
            max-width: 600px;
            margin: 0 auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
        h1 {
            color: #d63031;
        }
        input[type="text"], input[type="password"], input[type="email"], select {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            background-color: #d63031;
            color: #fff;
            border: none;
            padding: 12px 20px;
```

```html
            border-radius: 5px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Blood Bank Registration</h1>
        <form>
            <input type="text" placeholder="Full Name">
            <input type="email" placeholder="Email">
            <input type="password" placeholder="Password">
            <select>
                <option value="" disabled selected>Select Blood Group</option>
                <option value="A+">A+</option>
                <option value="B+">B+</option>
                <option value="O+">O+</option>
                <option value="AB+">AB+</option>
                <option value="A-">A-</option>
                <option value="B-">B-</option>
                <option value="O-">O-</option>
                <option value="AB-">AB-</option>
            </select>
            <button type="submit">Register</button>
        </form>
    </div>
</body>
</html>
```

JavaScript

1. Write a JavaScript program to display the current day and time in the following format. Sample Output : Today is : Tuesday. Current time is : 10 PM : 30 : 38

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Current Day and Time</title>
    <style>
        /* Styling for the page body */
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f4f4f4;
        }

        /* Styling for the output container */
        #output {
            background-color: #fff;
            max-width: 400px;
            margin: 0 auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <div id="output">
        <!-- Placeholder for the day of the week -->
        <h2>Today is: <span id="day"></span></h2>
        <!-- Placeholder for the current time -->
        <h2>Current time is: <span id="time"></span></h2>
    </div>

    <script>
        // Function to update the day and time
        function updateDayAndTime() {
            // Array of days of the week
            const daysOfWeek = ["Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"];

            // Create a new Date object to get the current date and time
            const now = new Date();

            // Get the day of the week (0 = Sunday, 1 = Monday, ...)
            const dayOfWeek = daysOfWeek[now.getDay()];

            // Get the current hour (in 24-hour format)
            let hours = now.getHours();
            let period = "AM";

            // Convert to 12-hour time format and determine AM/PM
            if (hours >= 12) {
                period = "PM";
                if (hours > 12) {
                    hours -= 12;
                }
            }

            // Get the minutes and seconds, and pad them with leading zeros if
needed
            const minutes = now.getMinutes().toString().padStart(2, "0");
            const seconds = now.getSeconds().toString().padStart(2, "0");

            // Display the day and time in the HTML elements
            document.getElementById("day").textContent = dayOfWeek;
            document.getElementById("time").textContent = `${hours} ${period} :
${minutes} : ${seconds}`;
        }

        // Call the function to display day and time immediately
        updateDayAndTime();

        // Update the time every second using setInterval
        setInterval(updateDayAndTime, 1000);
    </script>
</body>
</html>
```

2. Write a JavaScript program to find out if 1st January will be a Sunday between 2014 and 2050.

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>January 1st on Sunday</title>
    <style>
        /* Styling for the output */
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f4f4f4;
        }
        #output {
            background-color: #fff;
            max-width: 400px;
            margin: 0 auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <div id="output">
        <h2>Checking if January 1st will be a Sunday:</h2>
        <p>Between the years 2014 and 2050.</p>
        <p id="result"></p>
    </div>

    <script>
        // Function to check if January 1st will be a Sunday
        function checkJanuary1stSundays() {
            // Initialize variables
            let year;
            let found = false;

            // Loop through the years from 2014 to 2050
            for (year = 2014; year <= 2050; year++) {
                const date = new Date(year, 0, 1); // January is month 0 in
JavaScript

                // Check if January 1st is a Sunday (0 indicates Sunday)
                if (date.getDay() === 0) {
                    found = true;
                    break; // Exit the loop if found
                }
            }

            // Display the result
            if (found) {
                document.getElementById("result").textContent = `Yes, January 1st
will be a Sunday in ${year}.`;
            } else {
                document.getElementById("result").textContent = "No, January 1st
will not be a Sunday between 2014 and 2050.";
            }
        }

        // Call the function to check
        checkJanuary1stSundays();
    </script>
</body>
</html>
```

This code checks each year between 2014 and 2050 to see if January 1st falls on a Sunday. It uses a loop to iterate through the years and the **`getDay()`** method of the JavaScript **`Date`** object to check the day of the week. The result is displayed on the webpage

3. Write a JavaScript program to replace each character in a given string with the next in the English alphabet. Note: 'a' will be replaced by 'b' or 'z' would be replaced by 'a'.

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Alphabet Character Replacement</title>
    <style>
        /* Styling for the output */
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f4f4f4;
        }
        #output {
            background-color: #fff;
            max-width: 400px;
            margin: 0 auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <div id="output">
        <h2>Alphabet Character Replacement</h2>
        <p>Original String: <span id="originalString"></span></p>
        <p>Modified String: <span id="modifiedString"></span></p>
    </div>

    <script>
        // Function to replace characters in the string
        function replaceCharacters(inputString) {
            // Initialize an empty string to store the modified result
            let resultString = "";

            // Loop through each character in the input string
            for (let i = 0; i < inputString.length; i++) {
                let char = inputString.charAt(i);

                // Check if the character is a lowercase letter (a-z)
                if (char >= 'a' && char <= 'z') {
                // Replace the character with the next character in the alphabet
                    // Special case: 'z' is replaced by 'a'
                    if (char === 'z') {
                        resultString += 'a';
                    } else {
              // Use charCodeAt and String.fromCharCode to get the next character
                        const nextCharCode = char.charCodeAt(0) + 1;
                        resultString += String.fromCharCode(nextCharCode);
                    }
                } else {
                 // If the character is not a lowercase letter, keep it unchanged
                    resultString += char;
                }
            }

            return resultString;
        }
        // Call the function to replace characters in a sample string
        const originalString = "hello world";
        const modifiedString = replaceCharacters(originalString);

        // Display the original and modified strings
        document.getElementById("originalString").textContent = originalString;
        document.getElementById("modifiedString").textContent = modifiedString;
    </script>
</body>
</html>
```

Explanation of the logic:

1. We define a function `replaceCharacters(inputString)` that takes an input string as its parameter.
2. Inside the function, we initialize an empty string `resultString` to store the modified result.
3. We loop through each character in the input string using a `for` loop.
4. For each character, we check if it is a lowercase letter (a-z) using the `char >= 'a' && char <= 'z'` condition.
5. If it's a lowercase letter, we handle the replacement logic. If the character is 'z', we replace it with 'a'. Otherwise, we use the `charCodeAt` method to get the ASCII code of the character, add 1 to it to get the code of the next character in the alphabet, and then use `String.fromCharCode` to convert it back to a character.
6. If the character is not a lowercase letter, we keep it unchanged.
7. The modified characters are concatenated to `resultString`.
8. Finally, we return `resultString`, which contains the modified string.
9. We call this function with a sample string, display the original and modified strings on the webpage, and it will replace each character in the string with the next character in the English alphabet while handling the special case of 'z'.

4. Write a JavaScript program to find the types of a given angle. Types of angles:Acute angle: An angle between 0 and 90 degrees. Right angle: An 90 degree angle. Obtuse angle: An angle between 90 and 180 degrees. Straight angle: A 180 degree angle.

Ans: -

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Angle Type Finder</title>
    <style>
        /* Styling for the output */
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f4f4f4;
        }
        #output {
            background-color: #fff;
            max-width: 400px;
            margin: 0 auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <div id="output">
        <h2>Angle Type Finder</h2>
        <label for="angleInput">Enter an angle (in degrees): </label>
        <input type="number" id="angleInput">
        <button onclick="findAngleType()">Find Angle Type</button>
        <p id="result"></p>
    </div>

    <script>
        // Function to find and display the type of angle
        function findAngleType() {
            // Get the input angle from the user
            const angle =
parseFloat(document.getElementById("angleInput").value);

            // Check if the input is a valid number
            if (!isNaN(angle)) {
                let angleType;

                // Determine the type of angle based on its value
                if (angle > 0 && angle < 90) {
                    angleType = "Acute angle";
                } else if (angle === 90) {
                    angleType = "Right angle";
                } else if (angle > 90 && angle < 180) {
                    angleType = "Obtuse angle";
                } else if (angle === 180) {
                    angleType = "Straight angle";
                } else {
                    angleType = "Invalid angle (not in the range 0-180 degrees)";
                }

                // Display the result
              document.getElementById("result").textContent = `The angle is a
${angleType}.`;
            } else {
                // Display an error message if the input is not a valid number
                document.getElementById("result").textContent = "Please enter a
valid angle (numeric value).";
            }
        }
    </script>
</body>
</html>
```

Explanation of the logic:

1. We create an HTML form that includes an input field for the angle, a button to trigger the calculation, and a paragraph to display the result.
2. We define a JavaScript function `findAngleType()` that is called when the button is clicked.
3. Inside the function, we get the value of the input angle using `document.getElementById("angleInput").value` and parse it as a floating-point number using `parseFloat()`.
4. We check if the input is a valid number using `isNaN()`. If it's not a valid number, we display an error message.
5. If the input is a valid number, we determine the type of angle based on its value:
   - If the angle is between 0 and 90 degrees, it's an "Acute angle."
   - If the angle is exactly 90 degrees, it's a "Right angle."
   - If the angle is between 90 and 180 degrees, it's an "Obtuse angle."
   - If the angle is exactly 180 degrees, it's a "Straight angle."
   - If the angle is outside the valid range, we consider it an "Invalid angle."
6. We display the result in the HTML paragraph with the id "result."

This program allows users to input an angle and then calculates and displays its type based on the defined criteria.

5. Write a JavaScript program to check whether it is possible to replace $ in a given expression x $ y = z with one of the four signs +, -, * or / to obtain a correct expression. For example x = 10, y = 30 and z = 300, we can replace $ with a multiple operator (*) to obtain x * y = z

Ans: -

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Expression Checker</title>
    <style>
        /* Styling for the output */
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f4f4f4;
        }
        #output {
            background-color: #fff;
            max-width: 400px;
            margin: 0 auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <div id="output">
        <h2>Expression Checker</h2>
        <label for="xInput">Enter the value of x: </label>
        <input type="number" id="xInput">
        <label for="yInput">Enter the value of y: </label>
        <input type="number" id="yInput">
        <label for="zInput">Enter the value of z: </label>
        <input type="number" id="zInput">
        <button onclick="checkExpression()">Check Expression</button>
        <p id="result"></p>
    </div>
    <script>
        // Function to check if the expression is correct and find the operator
        function checkExpression() {
            // Get the values of x, y, and z from the user
            const x = parseFloat(document.getElementById("xInput").value);
            const y = parseFloat(document.getElementById("yInput").value);
            const z = parseFloat(document.getElementById("zInput").value);

            // Check if any of the inputs are not valid numbers
            if (isNaN(x) || isNaN(y) || isNaN(z)) {
                document.getElementById("result").textContent = "Please enter
valid numeric values for x, y, and z.";
                return;
            }

            // Initialize an empty string to store the operator
            let operator = "";

            // Check if replacing $ with any of the operators (+, -, *, /) makes
the expression true
            if (x + y === z) {
                operator = "+";
            } else if (x - y === z) {
                operator = "-";
            } else if (x * y === z) {
                operator = "*";
            } else if (x / y === z) {
                operator = "/";
            }

            // Display the result
            if (operator !== "") {
                document.getElementById("result").textContent = `It is possible
to replace $ with "${operator}" to make the expression true.`;
            } else {
                document.getElementById("result").textContent = "It is not
possible to replace $ with any operator to make the expression true.";
            }
        }
    </script>
</body>
</html>
```

Explanation of the logic:

1. We create an HTML form that includes input fields for the values of `x`, `y`, and `z`, a button to trigger the calculation, and a paragraph to display the result.
2. We define a JavaScript function `checkExpression()` that is called when the button is clicked.
3. Inside the function, we get the values of `x`, `y`, and `z` from the user using `parseFloat()`.
4. We check if any of the inputs are not valid numbers using `isNaN()`. If any of them are invalid, we display an error message.
5. If all inputs are valid, we check if replacing `$` with any of the operators (+, -, *, /) makes the expression true by testing all possible combinations.
6. We display the result in the HTML paragraph with the id "result."

This program allows users to input values for `x`, `y`, and `z` and checks whether it's possible to replace `$` with one of the four operators to make the expression true.

6. Write a Java Script to display sum of integers for entered numbers on browsers screen (e. g. if users input 10 then result will be 0+1+2+3+4+5+6+7+8+9+10=55.

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sum of Integers</title>
    <style>
        /* Styling for the output */
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f4f4f4;
        }
        #output {
            background-color: #fff;
            max-width: 400px;
            margin: 0 auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <div id="output">
        <h2>Sum of Integers</h2>
        <label for="numberInput">Enter a number: </label>
        <input type="number" id="numberInput">
        <button onclick="calculateSum()">Calculate Sum</button>
        <p id="result"></p>
    </div>

    <script>
        // Function to calculate and display the sum of integers
        function calculateSum() {
            // Get the input number from the user
            const number =
parseInt(document.getElementById("numberInput").value);

            // Check if the input is a valid positive integer
            if (!isNaN(number) && number >= 0) {
                // Initialize a variable to store the sum
                let sum = 0;
                let steps = "";

                // Loop through integers from 0 to the input number and add them
to the sum
                for (let i = 0; i <= number; i++) {
                    sum += i;
                    // Build the step-by-step addition string
                    steps += i + (i < number ? " + " : "");
                }

                // Display the result with step-by-step addition
                document.getElementById("result").textContent = `The sum of
integers from 0 to ${number} is ${steps} = ${sum}.`;
            } else {
                // Display an error message if the input is not a valid positive
integer
```

```
                document.getElementById("result").textContent = "Please enter a valid positive integer.";
            }
        }
    </script>
</body>
</html>
```

Explanation of the logic:

1. We create an HTML form that includes an input field for the user to enter a number, a button to trigger the calculation, and a paragraph to display the result.
2. We define a JavaScript function `calculateSum()` that is called when the button is clicked.
3. Inside the function, we get the value of the input number using `parseInt()`.
4. We check if the input is a valid positive integer using `isNaN()` and `number >= 0`.
5. If the input is a valid positive integer, we initialize a variable `sum` to store the sum of integers.
6. We use a `for` loop to iterate from 0 to the input number and add each integer to the `sum`.
7. We display the result in the HTML paragraph with the id "result."

This program allows users to input a number, and it calculates and displays the sum of integers from 0 to that number.

7. Write a JavaScript function to add rows to a table.

Ans: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Adding and Editing Rows in a Table</title>
    <style>
        /* Styling for the table and button */
        table {
            border-collapse: collapse;
            width: 80%;
            margin: 20px auto;
        }
        th, td {
            border: 1px solid #dddddd;
            text-align: left;
            padding: 8px;
        }
        th {
            background-color: #f2f2f2;
        }
        #addRowBtn {
            display: block;
            margin: 10px auto;
            padding: 10px 20px;
        }
        .editButton {
            background-color: #4CAF50;
            color: white;
            border: none;
            padding: 5px 10px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <h2>Table with Rows</h2>
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Age</th>
                <th>Country</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody id="tableBody">
            <tr>
                <td>John</td>
                <td>30</td>
                <td>USA</td>
                <td><button class="editButton"
onclick="editRow(this)">Edit</button></td>
            </tr>
            <tr>
                <td>Mary</td>
                <td>25</td>
                <td>Canada</td>
                <td><button class="editButton"
onclick="editRow(this)">Edit</button></td>
            </tr>
        </tbody>
    </table>
    <button id="addRowBtn" onclick="addRow()">Add Row</button>

    <script>
        // Function to add a new row to the table
        function addRow() {
            // Get a reference to the table body
            const tableBody = document.getElementById("tableBody");

            // Create a new table row element
            const newRow = document.createElement("tr");

            // Create table data (cell) elements with input fields
            const nameCell = document.createElement("td");
            const nameInput = document.createElement("input");
            nameInput.type = "text";
            nameInput.placeholder = "Name";
            nameCell.appendChild(nameInput);

            const ageCell = document.createElement("td");
            const ageInput = document.createElement("input");
            ageInput.type = "number";
            ageInput.placeholder = "Age";
            ageCell.appendChild(ageInput);

            const countryCell = document.createElement("td");
            const countryInput = document.createElement("input");
            countryInput.type = "text";
            countryInput.placeholder = "Country";
            countryCell.appendChild(countryInput);

            const actionCell = document.createElement("td");
            const editButton = document.createElement("button");
            editButton.className = "editButton";
            editButton.textContent = "Edit";
            editButton.onclick = function() {
                editRow(this);
            };
            actionCell.appendChild(editButton);
            // Append the cells with input fields to the new row
            newRow.appendChild(nameCell);
            newRow.appendChild(ageCell);
            newRow.appendChild(countryCell);
            newRow.appendChild(actionCell);

            // Append the new row to the table body
            tableBody.appendChild(newRow);
        }

        // Function to edit a specific row
        function editRow(button) {
            const row = button.parentElement.parentElement;
            const cells = row.getElementsByTagName("td");

            // Toggle the contenteditable attribute for each cell to make them
editable
            for (let i = 0; i < cells.length - 1; i++) {
                const cell = cells[i];
                const inputField = cell.querySelector("input");
                if (inputField) {
                    inputField.removeAttribute("readonly");
                }
            }

            // Change the button text to "Save" and update its click function
            button.textContent = "Save";
            button.onclick = function() {
                saveRow(this);
            };
        }

        // Function to save the changes made to a specific row
        function saveRow(button) {
            const row = button.parentElement.parentElement;
            const cells = row.getElementsByTagName("td");

            // Toggle the contenteditable attribute to make cells read-only again
            for (let i = 0; i < cells.length - 1; i++) {
                const cell = cells[i];
                const inputField = cell.querySelector("input");
                if (inputField) {
                    inputField.setAttribute("readonly", true);
                }
            }
            // Change the button text back to "Edit" and update its click
function
            button.textContent = "Edit";
            button.onclick = function() {
                editRow(this);
            };
        }
    </script>
</body>
</html>
```

Explanation of the code:

1. We start with the basic HTML structure, including a table with three columns (Name, Age, Country), two initial rows, and a button to add new rows.
2. The `addRow` function is defined. This function will be called when the "Add Row" button is clicked.
3. Inside the `addRow` function:
    - We get a reference to the `<tbody>` element with the id "tableBody" where we want to add rows.
    - We create a new table row (`<tr>`) element using `document.createElement("tr")`.
4. For each cell in the new row:
    - We create a new table data (`<td>`) element using `document.createElement("td")`.
    - We set the text content of each cell to the default values ("New Name," "New Age," "New Country").
5. We append the newly created table data elements to the new row.
6. Finally, we append the new row to the table body, effectively adding a new row to the table.

When you click the "Add Row" button, it will execute the `addRow` function, which adds a new row with default values to the table.

8. Write a Java Script to display Multiplication table of entered values of rows and columns as follows:

Ans: -

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Multiplication Table</title>
    <style>
        /* Styling for the output table */
        table {
            border-collapse: collapse;
            margin: 20px auto;
        }
        th, td {
            border: 1px solid #dddddd;
            text-align: center;
            padding: 8px;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <h2>Multiplication Table</h2>
    <label for="rows">Enter the number of rows:</label>
    <input type="number" id="rows" min="1" value="5">
    <label for="columns">Enter the number of columns:</label>
    <input type="number" id="columns" min="1" value="5">
    <button onclick="generateTable()">Generate Table</button>
    <div id="tableContainer"></div>

    <script>
        // Function to generate the multiplication table
        function generateTable() {
            // Get the number of rows and columns from the user input
            const numRows = parseInt(document.getElementById("rows").value);
            const numColumns =
parseInt(document.getElementById("columns").value);

            // Create a table element
            const table = document.createElement("table");

            // Create table header row
            const headerRow = document.createElement("tr");
            for (let j = 1; j <= numColumns; j++) {
                const th = document.createElement("th");
                th.textContent = j;
                headerRow.appendChild(th);
            }
            table.appendChild(headerRow);

            // Create the multiplication table
            for (let i = 1; i <= numRows; i++) {
                const row = document.createElement("tr");
                for (let j = 1; j <= numColumns; j++) {
                    const cell = document.createElement("td");
                    cell.textContent = i * j;
                    row.appendChild(cell);
                }
                table.appendChild(row);
            }

            // Clear previous table (if any) and display the new one
            const tableContainer = document.getElementById("tableContainer");
            tableContainer.innerHTML = "";
            tableContainer.appendChild(table);
        }
    </script>
</body>
</html>
```

Explanation of the code:

1. Users are prompted to enter the number of rows and columns they want for the multiplication table.
2. When the "Generate Table" button is clicked, the `generateTable` function is called.
3. Inside the `generateTable` function:
    - The number of rows and columns are retrieved from user input using `document.getElementById`.
    - A new `table` element is created.
4. A table header row is created, and column headers (numbers 1 to `numColumns`) are added to it.
5. A nested loop generates the multiplication table:
    - Rows are created in an outer loop (numbers 1 to `numRows`).

- For each row, cells are created in an inner loop (numbers 1 to `numColumns`), and the product of row number and column number is placed in each cell.

6. The previous table (if any) is cleared, and the newly generated table is appended to the `tableContainer` div for display.

This code dynamically generates a multiplication table based on user input for the number of rows and columns and updates the table when the user clicks the "Generate Table" button.