**Experiment 9**

**Creating Input Forms with Email and Phone Validation Using HTML, CSS, and JavaScript**

**Objective:**

The goal of this experiment is to develop an interactive, responsive web form that checks the validity of user inputs—specifically email addresses and phone numbers—by leveraging HTML for layout, CSS for styling, and JavaScript with the Validator.js library for input validation.

**Technologies Employed:**

* **HTML5** – Used to create the structure of the form
* **CSS3** – Utilized for styling and positioning
* **JavaScript (ES6)** – Used to implement validation logic
* **Validator.js** – A JavaScript library for validating string-based inputs

**Steps:**

**1. HTML Layout:**  
A form is built using HTML with fields for email and phone number. Labels and span elements are used to show real-time error messages.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Form Validation</title>

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

</head>

<body>

<div class="container">

<form id="myForm">

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<span id="emailError" class="error"></span>

<label for="phone">Phone Number:</label>

<input type="text" id="phone" name="phone" required>

<span id="phoneError" class="error"></span>

<button type="submit">Submit</button>

</form>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/validator/13.6.0/validator.min.js"></script>

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

</body>

</html>

**2. CSS Styling:**  
CSS is applied to create a modern, centered, and clean-looking form that enhances user experience.

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.container {

background-color: #fff;

padding: 20px;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

form {

display: flex;

flex-direction: column;

}

label {

margin-bottom: 5px;

}

input {

margin-bottom: 10px;

padding: 10px;

border: 1px solid #ccc;

border-radius: 3px;

}

button {

padding: 10px;

background-color: #28a745;

color: white;

border: none;

border-radius: 3px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

.error {

color: red;

font-size: 0.875em;

}

**3. JavaScript for Validation:**  
JavaScript is used to prevent default form submission and validate the email and phone number using Validator.js.

document.getElementById('myForm').addEventListener('submit', function (e) {

e.preventDefault();

const email = document.getElementById('email').value;

const phone = document.getElementById('phone').value;

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

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

emailError.textContent = '';

phoneError.textContent = '';

if (!validator.isEmail(email)) {

emailError.textContent = 'Please enter a valid email address.';

}

if (!validator.isMobilePhone(phone, 'any')) {

phoneError.textContent = 'Please enter a valid phone number.';

}

if (validator.isEmail(email) && validator.isMobilePhone(phone, 'any')) {

console.log('Email:', email);

console.log('Phone:', phone);

}

});

**Expected Behavior:**

* **Valid Input:**
  + No error messages displayed.
  + Console logs:
  + Email: test@example.com
  + Phone: 9870800776
* **Invalid Input:**
  + Error messages appear below the respective fields in red.
  + Examples:
    - Email: user@example.com, Phone: 9870800776→ Valid
    - Email: user.com → Error: “Please enter a valid email address.”
    - Phone: abc123 → Error: “Please enter a valid phone number.”

**Conclusion:**

This activity illustrates how to implement client-side form validation for email and phone number inputs, enhancing data accuracy and user interaction. It showcases a practical use of Validator.js for building reliable web forms.



