

## Exercise 9

230701221

### **Design input forms that validate data and display error messages using HTML/CSS, JavaScript**

#### **Aim:**

To design and develop an input form that validates user data (email and phone number) using JavaScript and Validator.js, and displays appropriate error messages.

#### **Software & Tools Used:**

- HTML
- CSS
- JavaScript

#### **Objective:**

This experiment focuses on implementing client-side form validation to improve user experience and data accuracy. When a user submits the form, JavaScript will check whether the email and phone number entered are in a valid format. If invalid, it will display error messages beside the respective fields.\

#### **Procedure:**

- **HTML Structure:**
  - A form is created with input fields for email and phone number.
  - Each input has a corresponding `<span>` to display error messages.
  - A submit button is provided to trigger validation.
- **CSS Styling:**
  - The form is centered and styled using basic CSS.
  - Error messages are styled in red for better visibility.
  - The layout is responsive and user-friendly.

- **JavaScript Validation (with Validator.js):**

- An event listener is added to intercept form submission.
- The values of email and phone number fields are retrieved.
- `validator.isEmail()` and `validator.isMobilePhone()` methods are used to validate input.
- If validation fails, appropriate error messages are shown.
- If both validations pass, the data is logged to the console.

**HTML CODE:**

```
index.html x # style.css JS script.js
index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6   <title>Form Validation</title>
7   <link rel="stylesheet" href="style.css" />
8 </head>
9 <body>
10   <div class="container">
11     <form id="myForm" action="#" method="post">
12       <label for="email">Email:</label>
13       <input type="email" id="email" name="email" required />
14       <span id="emailError" class="error" aria-live="polite"></span>
15
16       <label for="phone">Phone Number:</label>
17       <input type="text" id="phone" name="phone" required />
18       <span id="phoneError" class="error" aria-live="polite"></span>
19
20       <button type="submit">Submit</button>
21     </form>
22   </div>
23
24   <script src="https://cdnjs.cloudflare.com/ajax/libs/validator/13.6.0/vali"
25   <script src="script.js"></script>
26 </body>
27 </html>
```

CSS Code:

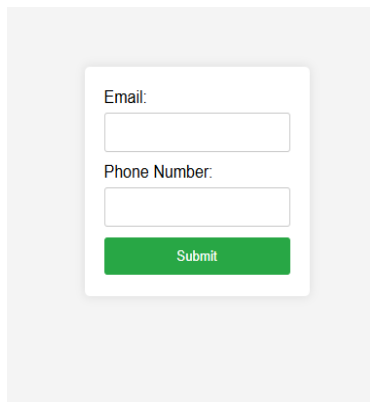
```
index.html # style.css X JS script.js ...
# style.css > ...
11 .container {
12     padding: 20px;
13     border-radius: 5px;
14     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
15 }
16
17
18 form {
19     display: flex;
20     flex-direction: column;
21 }
22
23 label {
24     margin-bottom: 5px;
25 }
26
27 input {
28     margin-bottom: 10px;
29     padding: 10px;
30     border: 1px solid #ccc;
31     border-radius: 3px;
32 }
33
34 button {
35     padding: 10px;
36     background-color: #28a745;
37     color: white;
38     border: none;
39     border-radius: 3px;
40     cursor: pointer;
41 }
42
43 button:hover {
44     background-color: #218838;
45 }
46
47 .error {
48     color: red;
49     font-size: 0.875em;
50 }
51
```

## JAVA SCRIPT CODE:

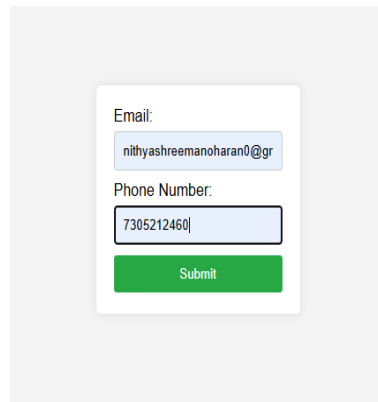
```
index.html # style.css JS script.js X ...
JS script.js > @ addEventListener(submit) callback
1 document.getElementById('myForm').addEventListener('submit', function (e) {
2     e.preventDefault();
3
4     // Get input values
5     const email = document.getElementById('email').value.trim();
6     const phone = document.getElementById('phone').value.trim();
7
8     // Get error message containers
9     const emailError = document.getElementById('emailError');
10    const phoneError = document.getElementById('phoneError');
11
12    // Clear previous error messages
13    emailError.textContent = '';
14    phoneError.textContent = '';
15
16    // Validate email
17    if (!validator.isEmail(email)) {
18        emailError.textContent = 'Please enter a valid email address.';
19    }
20
21    // Validate phone number
22    if (!validator.isMobilePhone(phone, 'any')) {
23        phoneError.textContent = 'Please enter a valid phone number.';
24    }
25
26    // If both validations pass
27    if (validator.isEmail(email) && validator.isMobilePhone(phone, 'any')) {
28        console.log('Email:', email);
29        console.log('Phone:', phone);
30        alert('Form submitted successfully!');
31        // Optional: Reset the form
32        // document.getElementById('myForm').reset();
33    }
34 });
35
```

## Code Files:

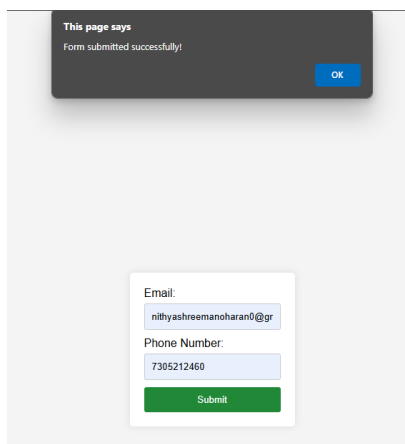
- **HTML (index.html):**  
Contains the form structure with inputs and placeholders for errors.
- **CSS (style.css):**  
Adds layout and styling to the form and its elements.
- **JavaScript (script.js):**  
Performs validation checks and manages error display.



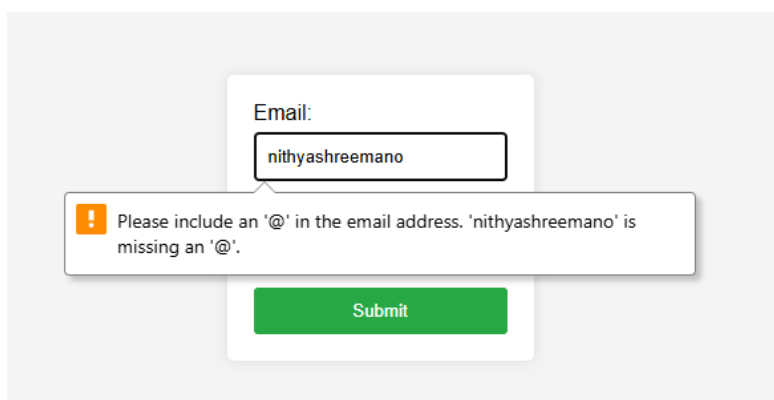
A form with two input fields and a submit button. The first field is labeled "Email:" and the second is labeled "Phone Number:". Both fields are empty. The submit button is green and labeled "Submit".



The form is filled with valid data. The "Email:" field contains "nithyashreemanoharan0@gr" and the "Phone Number:" field contains "7305212460". The submit button is green and labeled "Submit".



The form is shown with a success message at the top: "This page says Form submitted successfully!". The form fields are filled with the same data as the previous state. The submit button is green and labeled "Submit".



The form is shown with a validation error. The "Email:" field contains "nithyashreemano". A red error message box is displayed below the field, stating: "Please include an '@' in the email address. 'nithyashreemano' is missing an '@'." The submit button is green and labeled "Submit".

### ***Output / Result:***

When the form is submitted:

- If the input values are valid, the console displays the email and phone number.
- If invalid, relevant error messages are shown next to each input.

### **Conclusion:**

The experiment successfully demonstrates client-side form validation using Validator.js. This method is efficient for real-time user input checks and ensures better data integrity before processing on the server.