Interactive Form Validation

Interactive Form Validation - Full Guide

This assignment will help you understand JavaScript form validation, event handling, and DOM manipulation. Let's break it down step by step.

---

1. Key Features of the Form Validation

✅ Real-time validation for:

- Name (only letters, at least 3 characters)
- Email (valid format)
- Password (at least 8 characters, one uppercase, one number, one special character)
- Confirm Password (matches password)
  ✅ Dynamic error messages
  ✅ Submit button disabled until all fields are valid

---

2. Step-by-Step Implementation

1. HTML Structure

We will create a form with:

- Input fields for **name, email, password, and confirm password**
- A submit button (initially disabled)
- Error messages for invalid inputs

```
html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Interactive Form Validation</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h2>Register</h2>
        <form id="signupForm">
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" id="name" required>
                <small class="error"></small>
            </div>

            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" id="email" required>
                <small class="error"></small>
```

```
            </div>

            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" required>
                <small class="error"></small>
            </div>

            <div class="form-group">
                <label for="confirmPassword">Confirm Password:</label>
                <input type="password" id="confirmPassword" required>
                <small class="error"></small>
            </div>

            <button type="submit" id="submitBtn" disabled>Submit</button>
        </form>
    </div>

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

---

2. CSS for Styling

We will style the form to look clean and highlight invalid fields.

```
css

/* styles.css */
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f5f5f5;
}

.container {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    width: 300px;
}

h2 {
    text-align: center;
}

.form-group {
    margin-bottom: 15px;
}

label {
    font-weight: bold;
}

input {
    width: 100%;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

input:focus {
    border-color: #007bff;
    outline: none;
}
```

```
.error {
    color: red;
    font-size: 12px;
}

button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
}

button:disabled {
    background-color: #ccc;
    cursor: not-allowed;
}
```

---

3. JavaScript for Validation Logic

Now, let's add JavaScript to validate user input dynamically.

```
js

// script.js

document.addEventListener("DOMContentLoaded", function () {
    const form = document.getElementById("signupForm");
    const nameInput = document.getElementById("name");
    const emailInput = document.getElementById("email");
    const passwordInput = document.getElementById("password");
    const confirmPasswordInput = document.getElementById("confirmPassword");
    const submitBtn = document.getElementById("submitBtn");

    // Validation functions
    function validateName() {
        const nameValue = nameInput.value.trim();
        const error = nameInput.nextElementSibling;
        if (nameValue.length < 3 || !/^[a-zA-Z\s]+$/.test(nameValue)) {
            error.textContent = "Name must be at least 3 characters and contain only letters.";
            return false;
        }
        error.textContent = "";
        return true;
    }

    function validateEmail() {
        const emailValue = emailInput.value.trim();
        const error = emailInput.nextElementSibling;
        const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailPattern.test(emailValue)) {
            error.textContent = "Enter a valid email.";
            return false;
        }
        error.textContent = "";
        return true;
    }

    function validatePassword() {
        const passwordValue = passwordInput.value;
        const error = passwordInput.nextElementSibling;
        const passwordPattern = /^(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/;
        if (!passwordPattern.test(passwordValue)) {
            error.textContent = "Password must be 8+ chars, 1 uppercase, 1 number, 1 special char."
            return false;
        }
        error.textContent = "";
```

```
        return true;
    }

    function validateConfirmPassword() {
        const confirmPasswordValue = confirmPasswordInput.value;
        const error = confirmPasswordInput.nextElementSibling;
        if (confirmPasswordValue !== passwordInput.value) {
            error.textContent = "Passwords do not match.";
            return false;
        }
        error.textContent = "";
        return true;
    }

    // Check if all fields are valid
    function checkFormValidity() {
        submitBtn.disabled = !(validateName() && validateEmail() && validatePassword() && validateC
    }

    // Event Listeners for real-time validation
    nameInput.addEventListener("input", () => { validateName(); checkFormValidity(); });
    emailInput.addEventListener("input", () => { validateEmail(); checkFormValidity(); });
    passwordInput.addEventListener("input", () => { validatePassword(); checkFormValidity(); });
    confirmPasswordInput.addEventListener("input", () => { validateConfirmPassword(); checkFormValic

    // Prevent form submission if invalid
    form.addEventListener("submit", function (e) {
        e.preventDefault();
        if (validateName() && validateEmail() && validatePassword() && validateConfirmPassword()) {
            alert("Form Submitted Successfully!");
            form.reset();
            checkFormValidity(); // Disable button again
        }
    });

    checkFormValidity(); // Initial check to disable button
});
```

---

4. Explanation

1️⃣ HTML

- Form includes **text, email, and password fields**.
- `` `<small>` `` elements show error messages dynamically.
- The **Submit button is initially disabled**.

2️⃣ CSS

- The form has a **clean design with error messages in red**.
- **Disabled button** appears gray.
- Focus effect on input fields.

3️⃣ JavaScript

- **Validation functions** check if input values meet the requirements.
- **Event listeners** trigger validation as the user types.
- **Form submission** is prevented if any field is invalid.
- **Submit button only enables when all fields are valid**.

5. Summary

✅ **You learned:**

- Event listeners (`input`, `submit`)
- Regular Expressions (`RegExp`)
- Dynamic DOM Manipulation (`textContent`, `nextElementSibling`)
- Disabling/enabling buttons dynamically

🚀 **Next Steps:**

- Try adding more validations like **phone number, date of birth, or custom error messages**.

This approach ensures you **learn core validation logic** while making an **interactive, user-friendly form**. 🎯