LAB REPORT

$\boxed{02}$

CSE416: Web Engineering Lab
Experiment Title: Basic Javascript and Form Validation

Submitted To
Mr. Monthasir Delwar Afnan
Lecturer
Department of CSE
Daffodil International University

Submitted By
Student ID: 221-15-5953
Student Name: Md. Fahimur Rahman
Section: 61_P2
Department of CSE
Daffodil International University

Date of Assignment Distribution: 06/08/2025
Date of Assignment Submission: 15/08/2025

1. Objective

The objective of this lab is to understand and practice **basic JavaScript syntax** and apply it in **form validation**. This experiment also demonstrates the use of **functions for code reusability**, **arrays to store objects**, and the importance of client-side validation.

---

2. Tools and Technologies Used

- **HTML** – to design the form.
- **CSS** – for basic styling.
- **JavaScript** – for implementing validation and storing user data.

---

3. Explanation

a) JavaScript Functions and Reusability

Functions in JavaScript are reusable blocks of code that perform specific tasks. They reduce redundancy, improve readability, and make debugging easier. For example, instead of writing validation logic multiple times, we can write one function and reuse it.

b) Arrays and Storing Objects

Arrays in JavaScript allow us to store multiple values in a single variable. By combining arrays with objects, we can store multiple users' form data efficiently.
Example:

```
let users = [
  {name: "Md. Fahimur Rahman", email: "rahman15-5953@diu.edu.bd"},
];
```

c) Form Validation and Its Importance

Form validation ensures that the user inputs correct and expected values before submission. It:

- Prevents errors in data processing.
- Improves user experience by providing instant feedback.
- Reduces invalid submissions to the server.

---

4. Code Implementation

HTML + JavaScript Code with Explanation

HTML Part:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Form Validation Example</title>
  <style>
    body {
      font-family: 'Times New Roman', Times, serif;
      margin: 20px;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      min-height: 100vh;
      text-align: center;
    }
    form {
        width: 300px; padding: 20px; border: 1px solid #ccc; border-
radius: 10px;
    }
    input {
        display: block; width: 95%; margin: 10px 0; padding: 8px;
    }
```

```html
    .error {
        color: red; font-size: 14px;
    }
    .success {
        color: green; font-size: 14px;
    }
  </style>
</head>
<body>

  <h2>User Registration Form</h2>
  <form id="userForm">
    <input type="text" id="name" placeholder="Enter your name">
    <input type="email" id="email" placeholder="Enter your email">
    <input type="password" id="password" placeholder="Enter your password"><br>
    <button type="submit">Submit</button>
    <p id="message"></p>
  </form>

  <!-- Linking External JavaScript -->
  <script src="script.js"></script>

</body>
</html>
```

JavaScript Part:

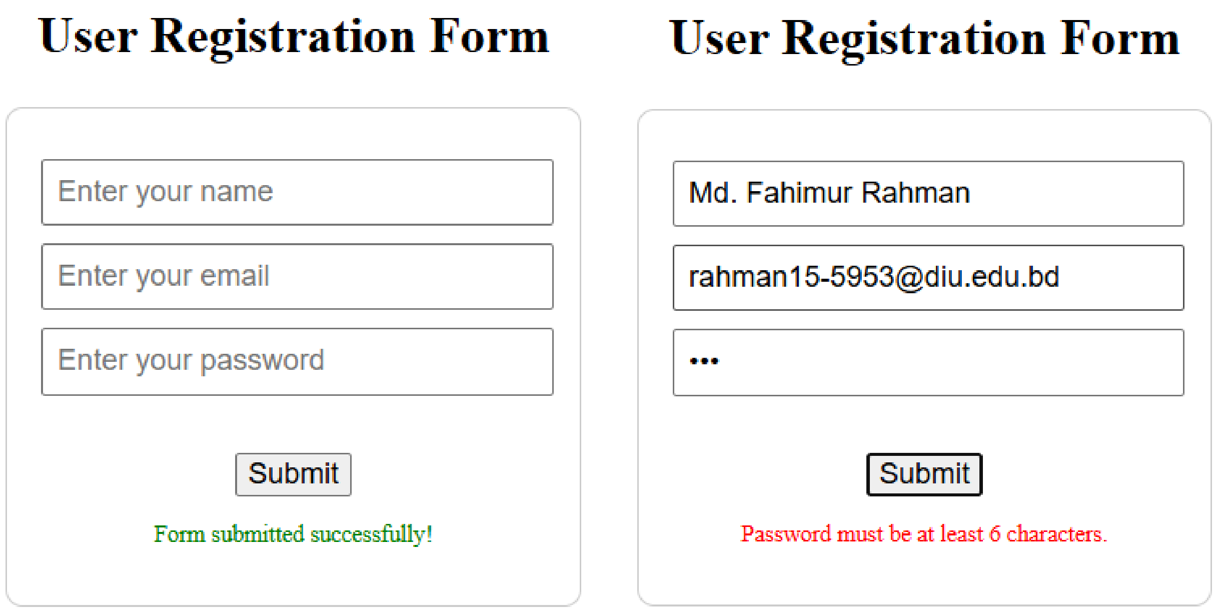
```
// Array to store user data as objects
let users = [];

// Function to validate name
function validateName(name) {
  return name.trim() !== "";
}

// Function to validate email
function validateEmail(email) {
  let pattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
  return pattern.test(email);
}

// Function to validate password (min 6 chars)
function validatePassword(password) {
  return password.length >= 6;
}

// Form submission handling
document.getElementById("userForm").addEventListener("submit",
function(event) {
  event.preventDefault(); // prevent page reload

  let name = document.getElementById("name").value;
  let email = document.getElementById("email").value;
  let password = document.getElementById("password").value;
  let message = document.getElementById("message");

  if (!validateName(name)) {
    message.textContent = "Name cannot be empty.";
    message.className = "error";
  } else if (!validateEmail(email)) {
    message.textContent = "Invalid email format.";
    message.className = "error";
  } else if (!validatePassword(password)) {
    message.textContent = "Password must be at least 6 characters.";
    message.className = "error";
  } else {
    // Store data in array as an object
    users.push({ name, email, password });
    message.textContent = "Form submitted successfully!";
    message.className = "success";

    // Clear inputs
    document.getElementById("userForm").reset();

    console.log("Users Array:", users);
  }
});// ...existing code...

// Example: Improved script.js structure
// Wait for the DOM to load before running scripts
document.addEventListener('DOMContentLoaded', () => {
    // Cache DOM elements
    const button = document.getElementById('myButton');
    const output = document.getElementById('output');

    // Utility function to display messages
    const showMessage = (message) => {
        output.textContent = message;
    };

    // Event handler for button click
    button.addEventListener('click', () => {
        // Example logic: show a message
        showMessage('Button was clicked!');
    });

    // Add more event listeners or logic as needed
});
```

Explanation of Code

1. **Functions** (`validateName`, `validateEmail`, `validatePassword`) make the code modular and reusable.
2. **Array `users[]`** stores submitted user data as objects.
3. **Form validation** checks user input before saving.
4. **Console log** shows stored data for verification.

5. Output / Screenshots

- **Form Interface:** Displays a simple user registration form.
- **Successful Validation:** Shows “Form submitted successfully!” and stores data.
- **Unsuccessful Validation:** Displays relevant error messages like “Invalid email format.”



Form submitted successfully!

Md. Fahimur Rahman

rahman15-5953@diu.edu.bd

•••

Password must be at least 6 characters.

Fig 5.1: *showing success and showing error validation.*

6. Conclusion

- **Importance of Client-Side Validation:** It prevents invalid data from reaching the server, improves efficiency, and enhances user experience by providing instant feedback.
- **Modular Code Advantages:** Using functions and arrays makes the program easier to maintain, debug, and extend in future projects.

7. GitHub Link

Code Link: Lab Report 2