Java Script **PART-1**

***JavaScript**: Introduction, variables, functions, Event handling, DOM, Form validation*

Introduction to JavaScript

JavaScript (**JS**) is a **Lightweight, Object Based Programming Language** mainly used to add **Interactivity, Dynamic Content and Logic** to web pages.

❖ It is one of the **core technologies of the web**, along with **HTML** and **CSS**.

What is JavaScript?

- JavaScript is a **Scripting Language** that runs inside the **web browser**.
- It allows developers to:
  - Change webpage content dynamically.
  - Validate forms.
  - Create animations and effects.
  - Interact with users.
  - Communicate with servers (AJAX, APIs).

Features of JavaScript

- **Lightweight** → Runs directly in browsers without extra setup.
- **Cross-platform** → Works on all modern browsers.
- **Interpreted** → Doesn't need compilation.
- **Event-driven** → Responds to user actions like clicks, hovers, typing.
- **Object-Based** → Supports objects and prototypes.
- **Versatile** → Used for frontend (React, Angular, Vue) and backend (Node.js).

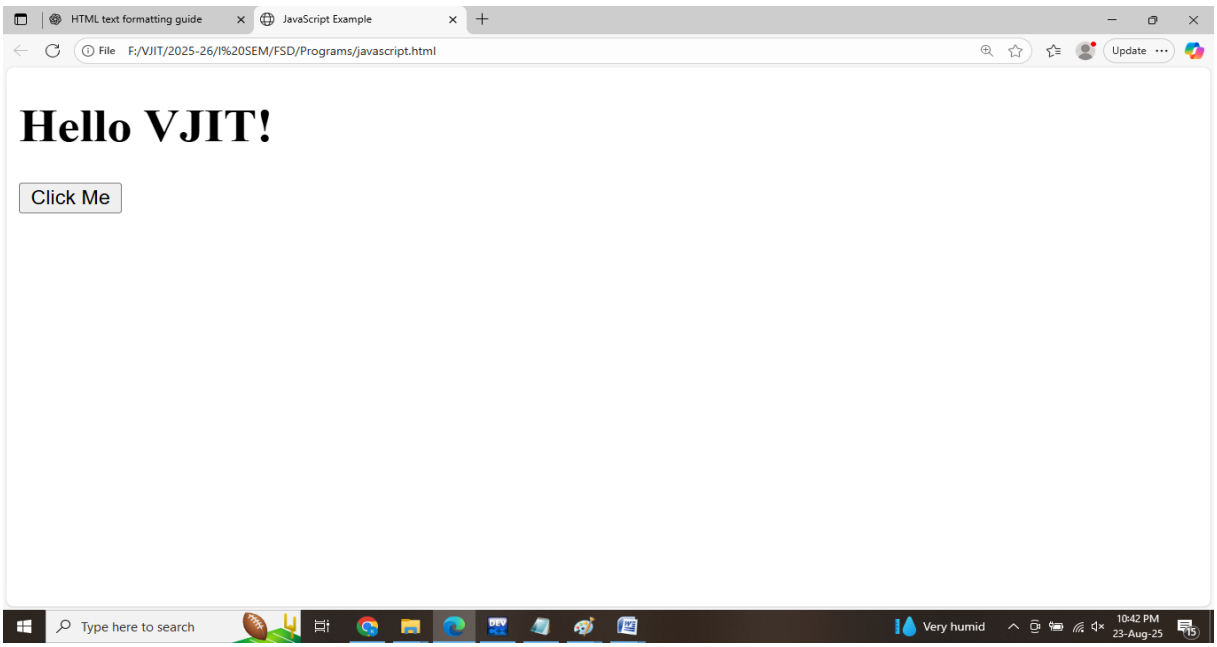
Where JavaScript is Used?

1. **Frontend Development** → Dynamic web pages, animations, validations.
2. **Backend Development** → Using Node.js.
3. **Mobile Apps** → React Native, Ionic, etc.
4. **Game Development** → Browser-based games.
5. **Server Communication** → AJAX, Fetch API, JSON.

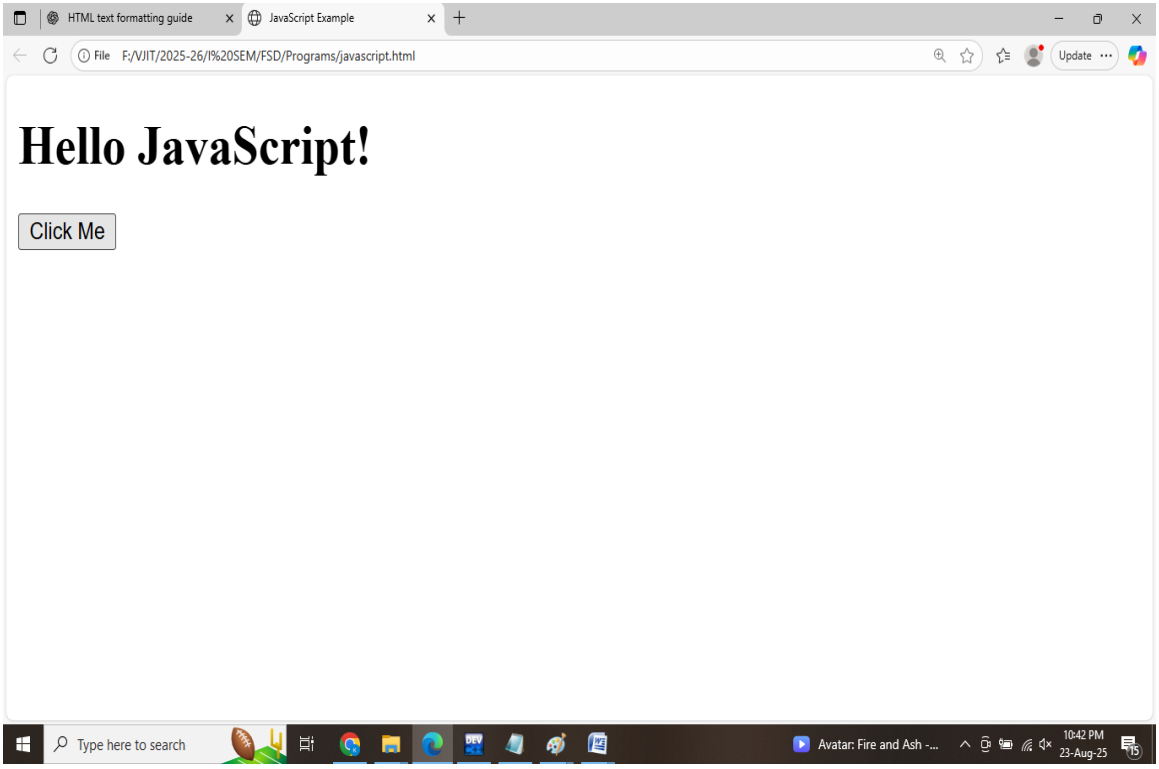
Example of JavaScript in HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Example</title>
</head>
<body>
  <h1 id="demo">Hello World!</h1>

  <button onclick="changeText()">Click Me</button>

  <script>
    function changeText()
  {
      document.getElementById("demo").innerHTML = "Hello JavaScript!";
    }
  </script>
</body>
</html>
```

Output:



- When you **click** the **button**, the heading **text changes dynamically**.



Advantages of JavaScript

- Makes web pages **interactive and dynamic**.
- Runs in the **browser without extra software**.
- Supported by **all major browsers**.
- Works with APIs (Google Maps, Weather, etc.).
- Used in **full-stack development**.

JavaScript = Brain of the Website

- **HTML** → Structure
- **CSS** → Styling
- **JavaScript** → Logic + Interactivity

JavaScript Variables

A **variable** is a container used to store data values (like numbers, text, objects, etc.).

- Think of a variable like a **box with a label** → You put data inside, and later you can use or update it.

Declaring Variables in JavaScript

JavaScript provides **three keywords** to declare variables:

1. var

- Old way of declaring variables (before ES6).
- Function-scoped (limited to the function where it is declared).
- Can be **re-declared** and **updated**.

```
var name = "John";
var name = "David"; // ✅ allowed
```

---

2. let

- Introduced in **ES6 (2015)**.
- **Block-scoped** (only available inside { }).
- Can be **updated**, but **not re-declared** in the same scope.

```
let age = 25;
age = 30;  // ✅ allowed
let age = 40;  // ❌ error
```

---

3. const

- Stands for **constant** (value cannot change).
- **Block-scoped**.
- Must be initialized at declaration.

```
const PI = 3.14;
PI = 3.1416;  // ❌ error (cannot update)
```

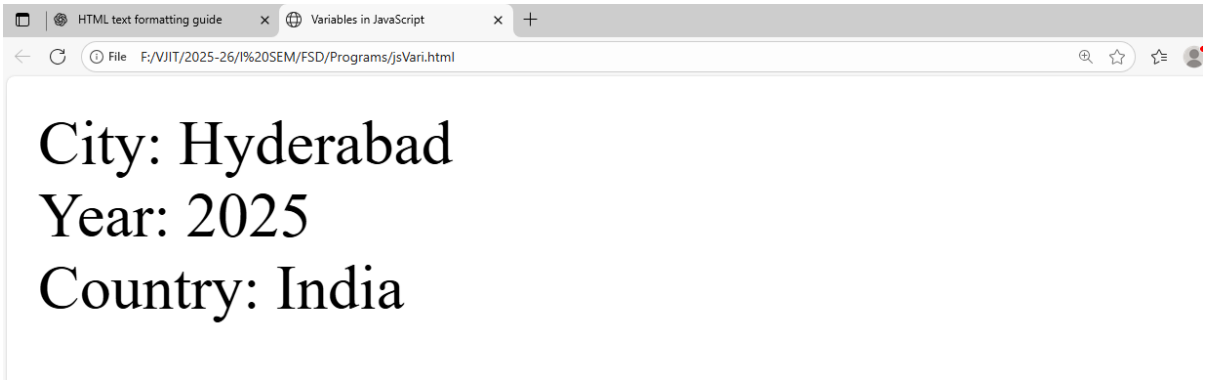
Variable Naming Rules

✅Names must start with a **letter, underscore (_), or dollar sign ($)**.
✅Can contain letters, digits, underscores, and $ symbols.
✅Case-sensitive (name ≠ Name).
❌Cannot use reserved keywords (like let, if, return).

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Variables in JavaScript</title>
</head>
<body>
  <script>
    var city = "Delhi";        // using var
    let year = 2025;           // using let
    const country = "India";   // using const

    document.write("City: " + city + "<br>");
    document.write("Year: " + year + "<br>");
    document.write("Country: " + country);
  </script>
</body>
</html>
```

Output



JavaScript Functions

A **function** is a block of code designed to perform a particular task.

- It allows us to **reuse code** instead of writing it multiple times.
- Functions make code **modular, readable, and maintainable**.

Defining a Function

1. Function Declaration

```
function greet()
{
  console.log("Hello, Welcome to JavaScript!");
}
```

2. Function Expression (storing function in a variable)

```
let greet = function()
{
  console.log("Hello, Welcome!");
};
```

3. Arrow Function (ES6)

Shorter syntax for functions.

```
let greet = () => {
  console.log("Hello, Welcome!");
};
```

Calling a Function

You must **call** (execute) a function to run its code.

```
function greet()
{
  console.log("Hello, World!");
}
greet();   // Calling the function
```

Function with Parameters

You can pass values (inputs) into functions.

```
function add(a, b)
{
  return a + b;
}
console.log(add(5, 10));  // Output: 15
```

Function with Default Parameters

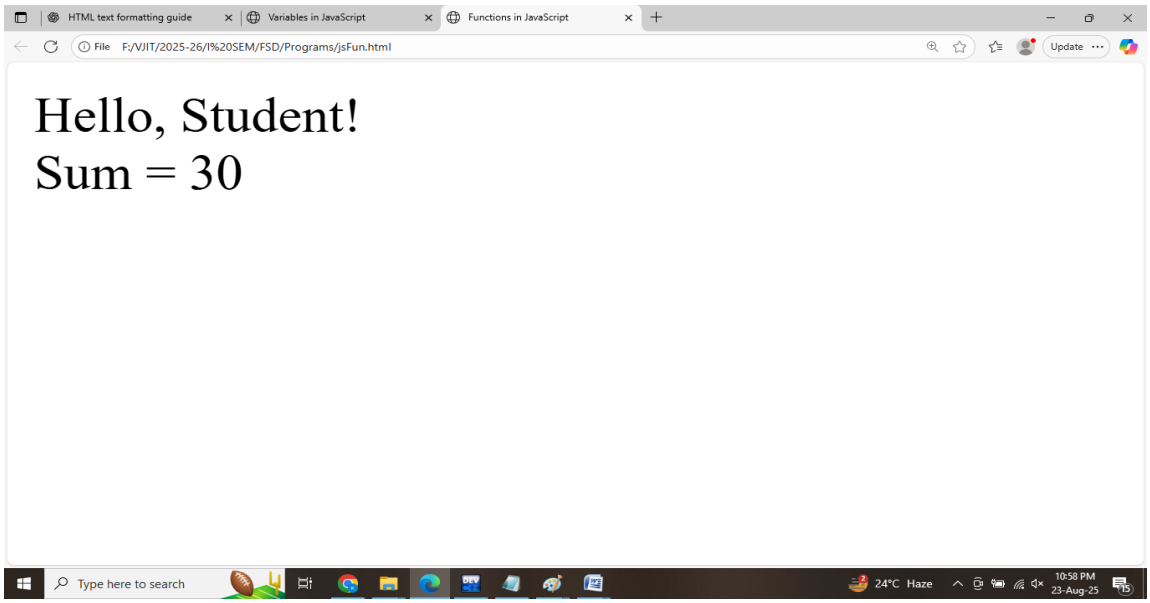
```
function multiply(x, y = 2)
{
  return x * y;
}
console.log(multiply(5));    // Output: 10 (since y=2 by default)
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Functions in JavaScript</title>
</head>
<body>
  <script>
    // Function without parameters
    function greet()
    {
      document.write("Hello, Student!<br>");
    }

    // Function with parameters
    function sum(a, b)
   {
      return a + b;
    }
```

```
    greet();  // call function
    document.write("Sum = " + sum(10, 20));
  </script>
</body>
</html>
```

Output



Summary

- Functions are reusable blocks of code.
- Can have **parameters** (inputs) and **return values** (outputs).
- Three ways: function, function expression, arrow function.

JavaScript Event Handling

Events are actions or occurrences that happen in the browser (e.g., user clicks a button, hovers over text, presses a key, etc.).
Event handling means writing JavaScript code to respond to these events.

Common JavaScript Events

Event	Description
onclick	Triggered when user clicks on an element
onmouseover	When mouse pointer moves over an element
onmouseout	When mouse pointer leaves an element
onkeydown	When a key is pressed down
onkeyup	When a key is released
onchange	When input value changes
onsubmit	When a form is submitted
onload	When a page has finished loading

Ways to Handle Events

1. Inline Event Handling

Add event directly in HTML.

```
<button onclick="alert('Button Clicked!')">Click Me</button>
```

2. JavaScript Event Property

Attach event in script.

```
<button id="btn">Click Me</button>
<script>
  document.getElementById("btn").onclick = function() {
    alert("Button was clicked!");
  };
</script>
```

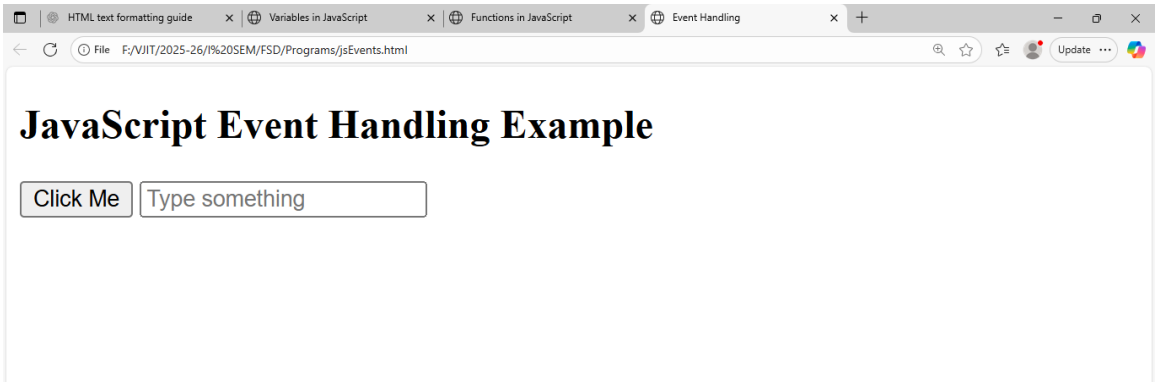
3. Event Listener Method

```
<button id="btn">Click Me</button>

<script>
  document.getElementById("btn").addEventListener("click", function() {
    alert("Hello! Event Listener Example");
  });
</script>
```

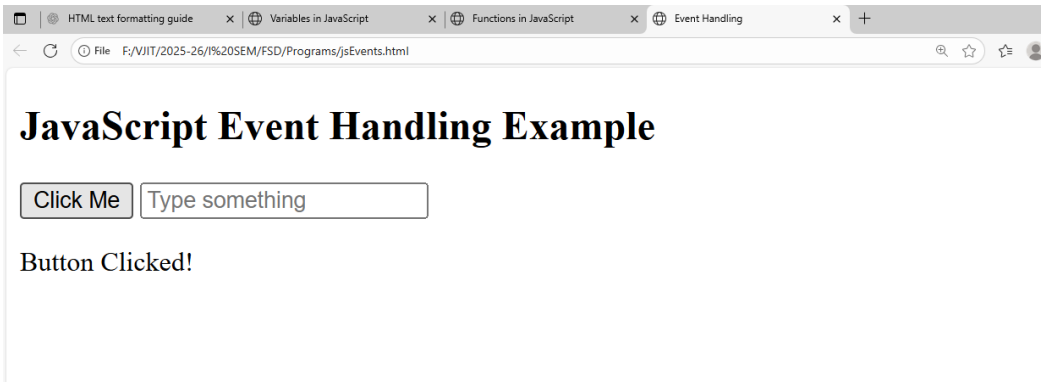
Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Event Handling</title>
</head>
<body>
  <h2>JavaScript Event Handling Example</h2>

  <button id="btn">Click Me</button>
  <input type="text" id="txt" placeholder="Type something">

  <p id="output"></p>

  <script>
    // Click Event
    document.getElementById("btn").addEventListener("click", function() {
      document.getElementById("output").innerText = "Button Clicked!";
    });

    // Keyup Event
    document.getElementById("txt").addEventListener("keyup", function() {
      document.getElementById("output").innerText = "You typed: " + this.value;
    });
  </script>
</body>
</html>
```

Output



- When you **click** the **button** → "Button Clicked!" shows.



- When you **type** in the **textbox** → shows "You typed: <your text>".

Summary

- Events let web pages react to user actions.
- 3 ways: **inline**, **event property**, **event listener**.
- Event listeners are the modern & flexible way.

Introduction to DOM

The **DOM** is a programming interface for web documents.

- It represents the **structure of an HTML document as a tree** of nodes (elements, attributes, text).
- With DOM, JavaScript can **access, modify, add, or delete** elements dynamically.

DOM Tree Example

```
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

DOM Tree Representation:

```
• document
  └── html
  ├──      head → title
  └──      body → h1, p
```

Accessing DOM Elements

1. By ID

document.getElementById("myId");

2. By Class

document.getElementsByClassName("myClass");

3. By Tag Name

document.getElementsByTagName("p");

4. Query Selector

```
document.querySelector(".myClass");   // first match
document.querySelectorAll("p");      // all <p> elements
```

Modifying Elements

```
<h2 id="title">Old Title</h2>
<script>
  document.getElementById("title").innerHTML = "New Title";  // change text
  document.getElementById("title").style.color = "red";      // change CSS
</script>
```

Creating & Adding Elements

```
<div id="container"></div>
<script>
  let newPara = document.createElement("p");
  newPara.innerText = "This is a new paragraph";
  document.getElementById("container").appendChild(newPara);
</script>
```

Removing Elements

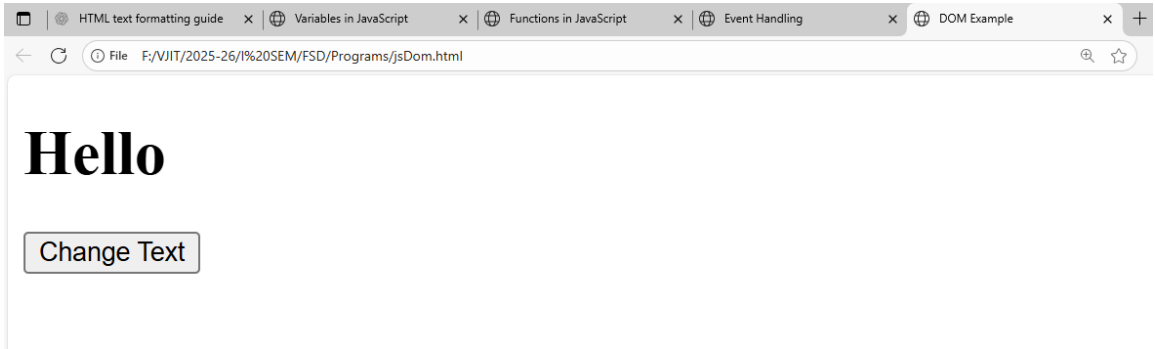
```
let para = document.getElementById("removeMe");
para.remove();
```

Example :

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Example</title>
</head>
<body>
  <h1 id="heading">Hello</h1>
  <button id="btn">Change Text</button>
  <div id="container"></div>
```

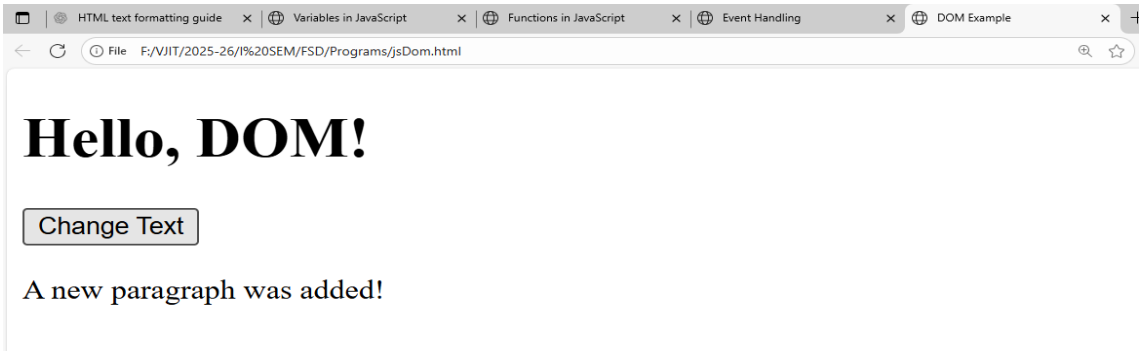
```
<script>
  // Change text on click
  document.getElementById("btn").addEventListener("click", function() {
    document.getElementById("heading").innerHTML = "Hello, DOM!";

    // Add new element
    let newPara = document.createElement("p");
    newPara.innerText = "A new paragraph was added!";
    document.getElementById("container").appendChild(newPara);
  });
 </script>
</body>
</html>
```

Output:



When you click the button:

- The heading text changes.
- A new paragraph is added dynamically.



Summary

- DOM = HTML as a tree structure.
- JavaScript can **access, modify, add, and remove** elements.
- DOM makes web pages **dynamic & interactive**.

Form Validation

Form Validation ensures that the user enters correct and complete data before submitting a form.

- It improves **data accuracy** and prevents **errors in backend processing**.

Types of Form Validation

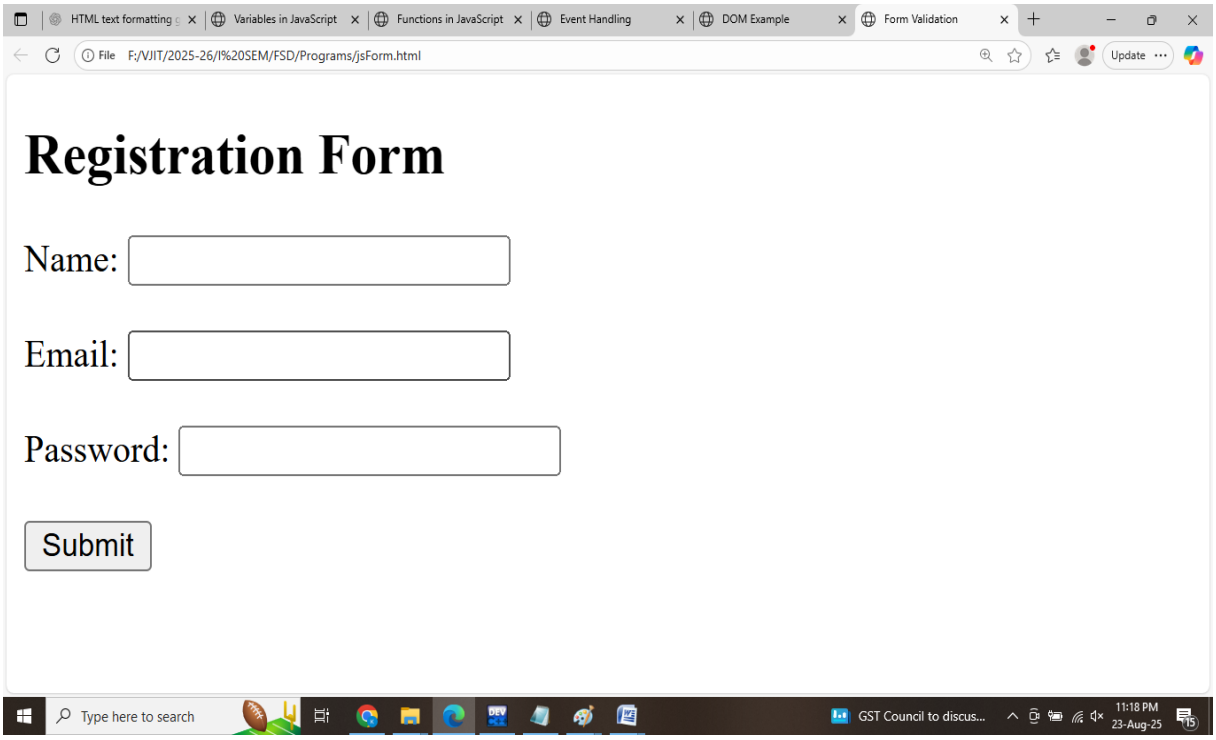
1. **Client-Side Validation** (using JavaScript/HTML5)
   - Fast, immediate feedback.
   - Example: checking empty fields, email format, password length.
2. **Server-Side Validation** (using PHP, Node.js, etc.)
   - More secure (cannot be bypassed easily).
   - Always required in addition to client-side validation.

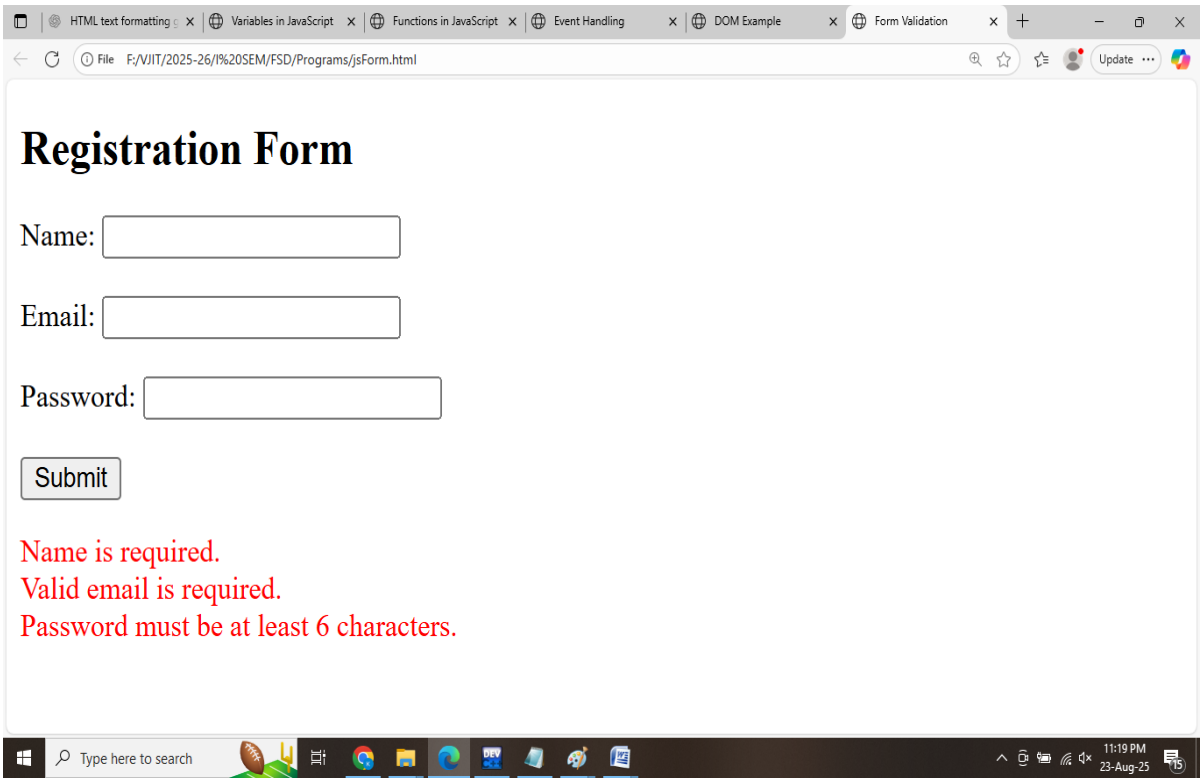
Example: Form

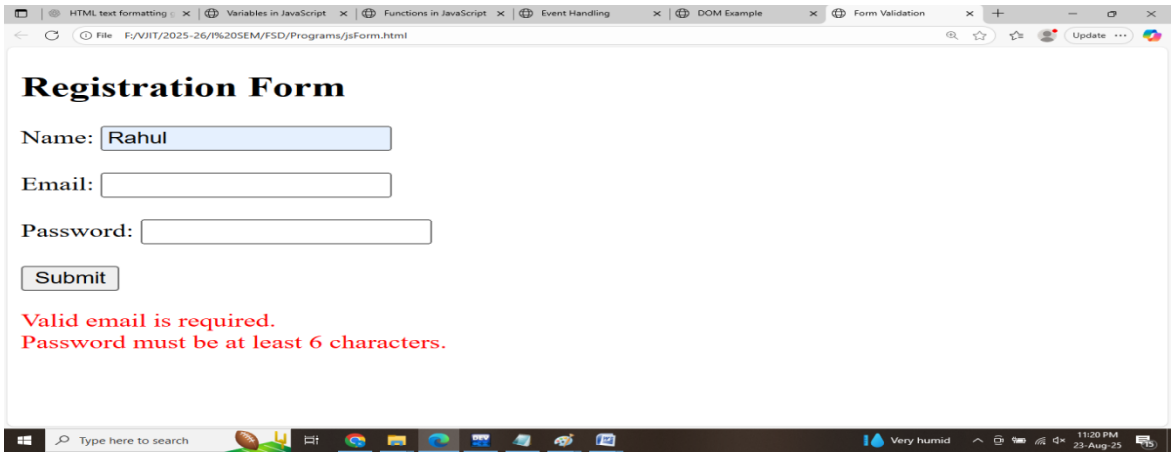
```
<!DOCTYPE html>
<html>
<head>
  <title>Form Validation</title>
</head>
<body>
  <h2>Registration Form</h2>
  <form id="myForm">
    Name: <input type="text" id="name"><br><br>
    Email: <input type="text" id="email"><br><br>
    Password: <input type="password" id="password"><br><br>
    <button type="submit">Submit</button>
  </form>

  <p id="errorMsg" style="color:red;"></p>

  <script>
    document.getElementById("myForm").addEventListener("submit",
function(event) {
      event.preventDefault(); // Stop form from submitting automatically
```

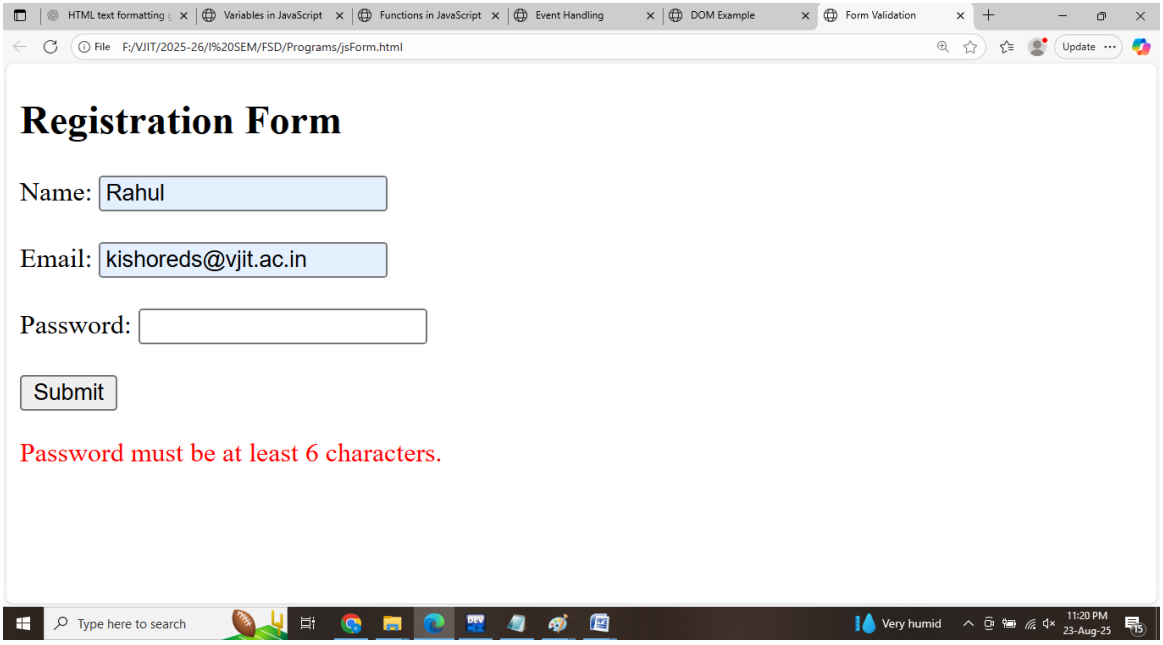
```
        let name = document.getElementById("name").value;
        let email = document.getElementById("email").value;
        let password = document.getElementById("password").value;
        let errorMsg = "";

    // Validation Rules
        if (name.trim() === "")
       {
          errorMsg += "Name is required.<br>";
        }

        if (email.trim() === "" || !email.includes("@"))
       {
          errorMsg += "Valid email is required.<br>";
        }

        if (password.length < 6)
       {
          errorMsg += "Password must be at least 6 characters.<br>";
        }

        // Display Errors or Success
        if (errorMsg !== "")
       {
          document.getElementById("errorMsg").innerHTML = errorMsg;
        }
      else
       {
        document.getElementById("errorMsg").innerHTML = "Form submitted
successfully!";
        }
       });
     </script>
    </body>
    </html>
```

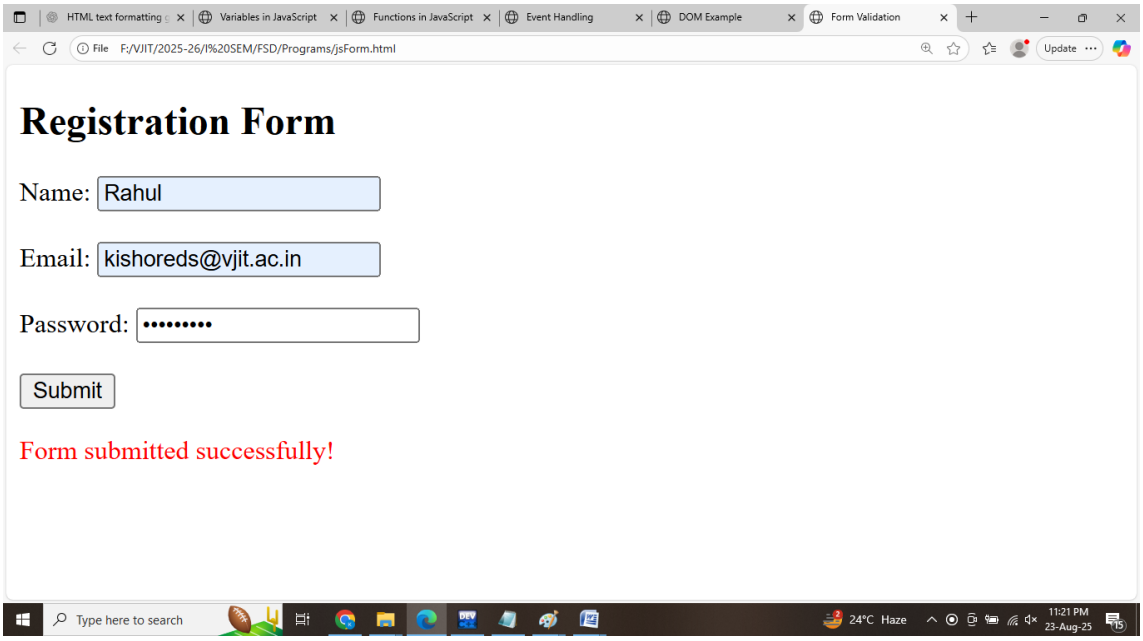
Output:


HTML text formatting
Variables in JavaScript
Functions in JavaScript
Event Handling
DOM Example
Form Validation
File F:/VJIT/2025-26/I%20SEM/FSD/Programs/jsForm.html
Registration Form
Name:
Email:
Password:
Submit
Type here to search
GST Council to discus...
11:18 PM
23-Aug-25


HTML text formatting
Variables in JavaScript
Functions in JavaScript
Event Handling
DOM Example
Form Validation
File F:/VJIT/2025-26/I%20SEM/FSD/Programs/jsForm.html
Registration Form
Name:
Email:
Password:
Submit
Name is required.
Valid email is required.
Password must be at least 6 characters.
Type here to search
11:19 PM
23-Aug-25


Registration Form
Name: Rahul
Email:
Password:
Submit
Valid email is required.
Password must be at least 6 characters.


Registration Form
Name: Rahul
Email: kishoreds@vjit.ac.in
Password:
Submit
Password must be at least 6 characters.


Registration Form
Name: Rahul
Email: kishoreds@vjit.ac.in
Password: •••••••••
Submit
Form submitted successfully!

Key Validations

✔Required Field Check

```
if (name.trim() === "") alert("Name is required");
```

✔Email Format Check (Regex)

```
let emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
if (!emailPattern.test(email)) alert("Enter a valid email");
```

✔Password Strength Check

```
if (password.length < 6) alert("Password too short");
```

Summary:

- Form validation ensures correct user input.
- Can be done using **JavaScript (custom rules)**.
- Always combine **client-side** and **server-side** validation for security.