Introduction

JavaScript is a client-side scripting language used to make web pages interactive.
It can be embedded or linked inside an HTML document so that browsers can execute the script.

There are **three main ways** to include JavaScript in an HTML page:

1. Inline JavaScript
2. Internal JavaScript
3. External JavaScript

---

1. Inline JavaScript

Definition

Inline JavaScript is written directly inside HTML tags using event attributes such as onclick, onmouseover, etc.

Syntax

<tagname onclick="JavaScript Code">

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>Inline JavaScript</title>
</head>
<body>

<button onclick="alert('Welcome to JavaScript Lab')">
   Click Me
</button>
```

</body>

</html>

Explanation

- JavaScript code is written inside the onclick attribute.
- When the button is clicked, the alert box appears.

Advantages

- Easy to use for small tasks
- No separate script section required

Disadvantages

- Not suitable for large programs
- Mixing HTML and JS reduces readability

---

2. Internal JavaScript

Definition

Internal JavaScript is written inside the same HTML file using the <script> tag.

It can be placed in:

- <head> section
- <body> section

Syntax

<script>

  JavaScript Code

</script>

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>Internal JavaScript</title>

   <script>
      function showMessage() {
         alert("Hello Students!");
      }
   </script>

</head>
<body>

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

</body>
</html>
```

Explanation

- Function is defined inside <script> tag.
- It is called when the button is clicked.

Advantages

- Code is centralized in one file
- Easier than inline for medium programs

Disadvantages

- HTML file becomes lengthy
- Not reusable across pages

---

3. External JavaScript

Definition

External JavaScript is written in a separate file with .js extension and linked to HTML.

Syntax

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

Example

HTML File

```
<!DOCTYPE html>
<html>
<head>
   <title>External JavaScript</title>
   <script src="script.js"></script>
</head>
<body>

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

</body>
</html>
```

JavaScript File (script.js)

```
function showMessage() {
   alert("Hello from External File");
}
```

Explanation

- JavaScript is stored in script.js.
- HTML links it using src attribute.

Advantages

- Reusable across multiple pages
- Clean separation of HTML and JS
- Easy maintenance

Disadvantages

- Requires separate file management
- Extra HTTP request (minor delay)

JavaScript – Objects, Functions and Arrays

---

1. JavaScript Objects

Introduction

An object is a collection of **key–value pairs** used to store related data and functionality.

It represents real-world entities like Student, Employee, Car, etc.

Syntax

```
var objectName = {
   property1: value1,
   property2: value2,
```

```
    property3: value3
};
```

Example

```
var student = {
    name: "Arun",
    regNo: 101,
    department: "CSE"
};

console.log(student.name);
```

Output

Arun

Accessing Object Properties

1. Dot Notation

```
student.name
```

2. Bracket Notation

```
student["regNo"]
```

Object with Function (Method)

```
var student = {
    name: "Divya",
    marks: 85,

    display: function() {
        return this.name + " scored " + this.marks;
```

```
    }
};

console.log(student.display());
```

Key Points

- Objects store data in properties.
- Functions inside objects are called **methods**.
- this refers to the current object.

---

2. JavaScript Functions

Introduction

A function is a block of code designed to perform a specific task.
It executes only when called.

Syntax

```
function functionName(parameters) {
    // code
}
```

Example 1 – Simple Function

```
function greet() {
    document.write("Welcome to JavaScript Lab");
}

greet();
```

Example 2 – Function with Parameters

```
function add(a, b) {
   return a + b;
}

var result = add(10, 20);
document.write(result);
```

Example 3 – Function with User Input

```
<!DOCTYPE html>
<html>
<head>
<script>
function findSquare() {
   var num =
      document.getElementById("num").value;

   var result = num * num;

   document.getElementById("output")
      .innerHTML = result;
}
</script>
</head>
<body>

Enter Number:
<input type="number" id="num">
```

```
<button onclick="findSquare()">
    Find Square
</button>

<h3 id="output"></h3>

</body>
</html>
```

Types of Functions

1. No parameter, no return value
2. Parameter, no return value
3. Parameter with return value

Advantages

- Code reusability
- Easy debugging
- Modular programming

---

3. JavaScript Arrays

Introduction

An array is used to store **multiple values in a single variable**.

Syntax

```
var arrayName = [value1, value2, value3];
```

Example

```
var subjects = ["C", "C++", "Java", "Python"];
```

```
document.write(subjects[0]);
```

Output

```
C
```

Accessing Array Elements

```
subjects[1];   // C++

subjects[2];   // Java
```

Array Using new Keyword

```
var marks = new Array(80, 85, 90);
```

Looping Through Array

```
var fruits = ["Apple", "Mango", "Orange"];


for (var i = 0; i < fruits.length; i++) {
   document.write(fruits[i] + "<br>");
}
```

Common Array Methods

Method	Description
push()	Adds element at end
pop()	Removes last element
shift()	Removes first element
unshift()	Adds at beginning
length	Returns array size

Example – Array Methods

```
var numbers = [10, 20, 30];


numbers.push(40);


console.log(numbers);
```

Difference – Objects vs Arrays

Feature	Object	Array
Data Type	Key–Value pairs	Indexed values
Access	Name / Key	Index
Example	student.name	marks[0]

JavaScript Events

Introduction

JavaScript events are actions or occurrences that happen in the browser, which can be detected and handled using JavaScript.

Events allow web pages to become interactive.

Examples of Events

- Clicking a button
- Moving the mouse
- Pressing a key
- Submitting a form

- Loading a page

Event Handling

Event handling means executing JavaScript code when an event occurs.

Syntax

Method 1 – Inline Event Handling

<tagname event="JavaScript Code">

Example

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

Common JavaScript Events

Event	Description
onclick	Triggered when element is clicked
ondblclick	Double click event
onmouseover	Mouse moves over element
onmouseout	Mouse leaves element
onchange	Value of input changes
onkeydown	Key is pressed
onload	Page finishes loading
onsubmit	Form is submitted

---

1. Click Event

Definition

Occurs when the user clicks on an element.

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function showMessage() {
   alert("Button Clicked!");
}
</script>
</head>
<body>

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

</body>
</html>
```

---

2. Double Click Event

```
<p ondblclick="changeText()">
   Double Click Me
</p>

<script>
```

```
function changeText() {
   document.write("Text Changed");
}
</script>
```

---

3. Mouse Over and Mouse Out Events

Definition

- onmouseover → When mouse enters element
- onmouseout → When mouse leaves element

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function changeColor() {
   document.body.style.backgroundColor =
      "yellow";
}

function resetColor() {
   document.body.style.backgroundColor =
      "white";
}
</script>
</head>
<body>
```

```
<h2 onmouseover="changeColor()"
   onmouseout="resetColor()">
   Move Mouse Over Me
</h2>


</body>
</html>
```

---

4. Onchange Event

Definition

Occurs when value of input field changes.

Example

```
Enter Name:
<input type="text"
      onchange="displayName(this.value)">


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


<script>
function displayName(name) {
   document.getElementById("output")
       .innerHTML = "Hello " + name;
}
</script>
```

---

5. Onload Event

Definition

Occurs when the web page finishes loading.

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function pageLoaded() {
   alert("Page Loaded Successfully");
}
</script>
</head>

<body onload="pageLoaded()">

<h2>Welcome to JavaScript</h2>

</body>
</html>
```

---

6. Onsubmit Event

Definition

Occurs when a form is submitted.

Used for form validation.

Example

```
<!DOCTYPE html>
<html>
```

```
<head>
<script>
function validateForm() {

   var name =
      document.getElementById("uname").value;

   if (name == "") {
      alert("Name cannot be empty");
      return false;
   }

   return true;
}
</script>
</head>
<body>

<form onsubmit="return validateForm()">

Name:
<input type="text" id="uname">

<input type="submit" value="Submit">

</form>

</body>
```

</html>

Event Handling Using addEventListener()

Definition

A modern method to attach events using JavaScript instead of HTML attributes.

Syntax

element.addEventListener("event", function);

Example

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


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

Advantages of Events

- Makes web pages interactive
- Improves user experience
- Enables validation
- Supports dynamic content changes

Key Points

- Events are user/browser actions.
- JavaScript responds using event handlers.

- onclick is the most commonly used event.
- addEventListener() is the preferred modern method.

DOM (Document Object Model)

Introduction to DOM

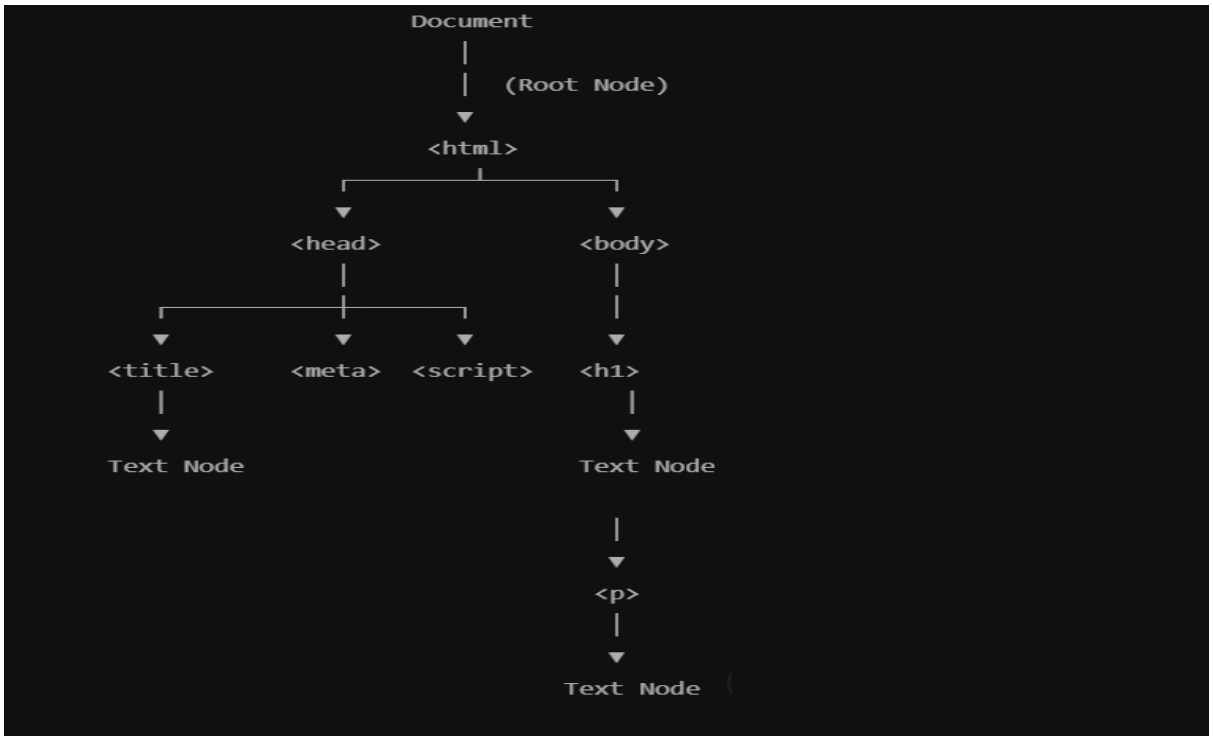
DOM stands for **Document Object Model**.

It is a programming interface that allows JavaScript to:

- Access HTML elements
- Modify content
- Change styles
- Add or delete elements

In DOM, the entire HTML document is treated as a **tree structure of objects**. “DOM represents an HTML document as a tree of nodes which JavaScript can access and manipulate.”

DOM Structure (Hierarchy)



Each part of the HTML page is considered a **node** in the DOM tree.

Why DOM is Used

DOM enables JavaScript to make web pages dynamic by allowing:

- Content updates without reloading page
- Form validation
- Interactive UI effects
- Event handling

Accessing HTML Elements Using DOM

JavaScript provides methods to select elements.

---

1. getElementById()

Selects element using ID.

Syntax

document.getElementById("idName");

Example

```
<h2 id="title">Welcome</h2>


<script>
document.getElementById("title")
    .innerHTML = "Hello Students";
</script>
```

---

2. getElementsByClassName()

Selects elements using class name.

Syntax

document.getElementsByClassName("className");

Example

```
<p class="msg">Message 1</p>

<p class="msg">Message 2</p>


<script>

document.getElementsByClassName("msg")[0]

    .innerHTML = "Updated Text";

</script>
```

---

3. getElementsByTagName()

Selects elements using tag name.

Syntax

```
document.getElementsByTagName("tagName");
```

Example

```
<p>Paragraph 1</p>


<script>

document.getElementsByTagName("p")[0]

    .style.color = "red";

</script>
```

---

4. querySelector()

Selects first matching element.

```
document.querySelector("p");
```

---

5. querySelectorAll()

Selects all matching elements.

```
document.querySelectorAll(".className");
```

Modifying HTML Content

---

innerHTML

Changes element content.

```
document.getElementById("demo")
    .innerHTML = "Content Changed";
```

---

innerText

Changes only text (ignores HTML tags).

```
element.innerText = "New Text";
```

Changing CSS Styles Using DOM

Syntax

```
element.style.property = "value";
```

Example

```
<p id="text">JavaScript DOM</p>


<script>
document.getElementById("text")
    .style.color = "blue";


document.getElementById("text")
    .style.fontSize = "25px";
</script>
```

Creating and Adding Elements

---

createElement()

Creates new element.

```
var para =
   document.createElement("p");
```

---

appendChild()

Adds element to page.

```
document.body.appendChild(para);
```

Full Example

```
<!DOCTYPE html>
<html>
<body>

<button onclick="addText()">
   Add Paragraph
</button>

<script>
function addText() {

   var para =
      document.createElement("p");

   para.innerHTML =
      "New Paragraph Added";
```

```
    document.body.appendChild(para);
}
</script>

</body>
</html>
```

Removing Elements

element.remove();

Example

document.getElementById("demo").remove();

DOM Events Connection

DOM works closely with events.

Example:

```
document.getElementById("btn")
        .onclick = function() {
            alert("DOM Event Triggered");
        };
```

Key DOM Properties & Methods

Property / Method	Purpose
innerHTML	Change content
style	Change CSS
value	Get input value
createElement()	Create element
appendChild()	Add element
remove()	Delete element

Advantages of DOM

- Dynamic page updates
- Interactive UI
- Real-time validation
- Easy element manipulation

Key Points to Remember

- DOM represents HTML as objects.
- JavaScript uses DOM to control web pages.
- getElementById() is most commonly used.
- **DOM + Events = Interactive websites.**