Web Development

Introduction to JavaScript

Presented by Muhammad Munim

What is JavaScript?

- JavaScript (JS) is a high-level, interpreted programming language used to make web pages interactive.
- Runs directly in web browsers, allowing dynamic and responsive behavior.
- Supports both client-side (browser) and server-side (Node.js) development.

Common Uses of JavaScript:

- **Form validation** (checking user input before submission).
- **Animations and transitions** (smooth effects for UI elements).
- **Interactive elements** (buttons, modals, dropdown menus, carousels).
- **DOM manipulation** (dynamically updating content without reloading the page).
- **Event handling** (responding to user actions like clicks and keypresses).

linkedin.com/in/m-munim1 github.com/M-Munim +92 330 2021926

A Brief History of JavaScript

- **Created by:** Brendan Eich in 1995 while working at Netscape.
- **Original Name:** Mocha → Later renamed to LiveScript → Finally, JavaScript.
- **Standardization:** ECMAScript (ES) was introduced to standardize JavaScript.

Major Versions:

- **ES5 (2009):** Introduced JSON, strict mode.
- **ES6 (2015):** Introduced let, const, arrow functions, template literals, etc.
- **Latest Versions (ES2023):** Include new features like async/await, optional chaining, and more.

0330 2021926

Importance of JavaScript in Web Development

JavaScript is a core technology of modern web development along with HTML and CSS.

- **Enhances User Experience** – Enables dynamic and interactive elements.
- **Client-Side Processing** – Reduces server load by executing code in the browser.
- **Asynchronous Operations** – Allows smooth loading of data without refreshing the page (AJAX, Fetch API).
- **Cross-Platform Compatibility** – Works on all major browsers and devices.
- **Powerful Ecosystem** – Numerous libraries and frameworks for fast development.

Famous JavaScript Frameworks and Libraries

JavaScript has a wide range of libraries and frameworks for front-end and back-end development.

Front-End Frameworks/Libraries:

- **React.js** – Developed by Facebook, used for building UI components.
- **Vue.js** – Lightweight and easy-to-learn framework for interactive UI.
- **Angular.js** – Developed by Google, used for large-scale applications.
- **jQuery** – Simplifies DOM manipulation and event handling.

Back-End Frameworks:

- **Node.js** – Enables JavaScript to run on the server.
- **Express.js** – A lightweight framework for building web applications with Node.js.

JavaScript and the MERN Stack

MERN is a popular JavaScript-based full-stack development framework.

- **M** – MongoDB (NoSQL database)
- **E** – Express.js (Back-end framework)
- **R** – React.js (Front-end framework)
- **N** – Node.js (Server-side runtime)

Why MERN?

- Uses JavaScript across the stack (front-end & back-end).
- Faster development with reusable components.
- Scalable and efficient for modern web applications.

0330 2021926

Setting Up JavaScript

Using the Browser Console

- Open Developer Tools (F12 or Ctrl + Shift + I in Chrome).
- Navigate to the Console tab.
- Type and execute JavaScript code directly.

0330 2021926

Example:

```
console.log("Hello, World!");
```

Adding JavaScript to HTML

JavaScript can be added inside an HTML file in two ways:

1. Internal JavaScript

Writing JavaScript code directly inside the HTML file within a <script> tag.

Example:

```
<script>
    console.log("Hello, JavaScript!");
</script>
```

0330 2021926

External JavaScript File

Writing JavaScript in a separate file and linking it using the <script> tag.

HTML File (index.html):

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

JavaScript File (script.js):

```
console.log("Hello, JavaScript!");
```

0330 2021926

Text Editor Setup

- Install Visual Studio Code (VS Code) or any code editor.
- Create an HTML file (index.html) and a JavaScript file (script.js).
- Link the JavaScript file inside the HTML file before the closing </body> tag for better performance.

0330 2021926

Writing Your First JavaScript Code

1. Console Output

- The console.log() method is used for debugging and printing messages to the console.

Example:

```
console.log("Hello, JavaScript!");
```

2. Alert Box

- The alert() function displays a popup message to the user.

Example:

```
alert("Welcome to JavaScript!");
```

0330 2021926

3. Writing into the HTML Page

- The document.write() method writes content directly into the HTML page.

Example:

```
document.write("Welcome to JavaScript!");
```

0330 2021926

Home Task 1

1. What is JavaScript?
2. How do you include JavaScript in an HTML file?
3. What is console.log() used for?
4. Write a JavaScript script to print "Hello, JavaScript!" in the console.
5. What is an alert box in JavaScript?
6. Write a script to display an alert box with the message "Welcome to JavaScript!".
7. Name some popular JavaScript frameworks and their uses.
8. What is the MERN stack?
9. Compare JavaScript and PHP.

0330 2021926

Web Development

Class No: 02

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926

Variables, Data Types & Operators

What is a Variable?

- A variable is a container that stores data values. It allows us to store, update, and reuse values in a program.

Declaring Variables

- JavaScript provides **three ways** to declare variables:

1. **var** – Old method (Not recommended due to scope issues).
2. **let** – Preferred for variables that can change.
3. **const** – Preferred for variables that do not change.

Example:

```
var name = "John";              // Old way (avoid using)
let age = 25;                   // Preferred for changeable values
const country = "Pakistan";     // Preferred for constant values
```

Rules for Naming Variables

- Can contain letters, digits, underscores (_), and dollar signs ($).
- Cannot start with a number.
- Cannot use JavaScript reserved keywords (like var, if, else).
  (https://www.w3schools.com/js/js_reserved.asp)
- Variables are case-sensitive (name and Name are different).

Example:

```
let user1 = "Alice";        // ☑ Valid
let _score = 100;           // ☑ Valid
let $price = 50;            // ☑ Valid
let 1user = "John";         // ❌ Invalid (Cannot start with a number)
let let = 10;               // ❌ Invalid (Reserved keyword)
```

“use strict”;

Treat all JS code as newer version. With strict mode, you can not, for example, use undeclared variables

```
"use strict";
// Example of an error with "use strict"
x = 10;      // Error: x is not defined
console.log(x);
```

0330 2021926

Data Types in JavaScript

A data type defines the kind of values a variable can hold. **JS has two main types:**

A. Primitive Data Types (Stores only a single value)

1. String (Text)

```
let greeting = "Hello, World!";
console.log(greeting);          // Output: Hello, World!
```

2. Number (Whole numbers & decimals)

```
let age = 30;
let price = 99.99;
console.log(age, price);
```

0330 2021926

3. Boolean (True or False)

```
let isRaining = false;
let isSunny = true;
console.log(isRaining);
```

4. Null (Intentional absence of value, stand alone value)

```
let car = null;
console.log(car);                    // Output: null
```

5. Undefined (Declared but no value assigned)

```
let temperature;
console.log(temperature);     // Output: undefined
```

0330 2021926

B. Non-Primitive Data Types (Stores multiple values)

1. Objects (Key-value pairs for data)

```
let student = {
    name: "Ali",
    age: 20
    isEnrolled: true
};
console.log(student.name);              // Output: Ali
```

2. Arrays (Ordered lists for storing data)

```
let colors = ["red", "green", "blue"];
console.log(colors);                    // Output: ['red', 'green', 'blue']
```

0330 2021926

Type Conversion (Data Type Conversion) in JavaScript

- It is the process of converting a value from one data type to another.

Implicit Conversion (Automatic by JavaScript)

```
console.log("5" + 5);        // "55" (String)
console.log("10" - 2);       // 8 (Number)
```

Explicit Conversion (Manual by Programmer)

- **String to Number Conversion**

```
let str = "123";
let num = Number(str);
console.log(num);            // 123 (Number)
```

0330 2021926

- **Number to String**

```
let num = 456;
let str = num.toString();
console.log(str);                    // "456" (String)
```

- **Boolean to Number**

```
console.log(Number(true));   // 1
console.log(Number(false));  // 0
```

0330 2021926

Operators in JavaScript

Operators are symbols that perform operations on variables and values.

A. Arithmetic Operators

```
let a = 10, b = 5;

console.log(a + b);        // 15
console.log(a - b);        // 5
console.log(a * b);        // 50
console.log(a / b);        // 2
console.log(a % b);        // 0
```

0330 2021926

B. Comparison Operators

```
let x = 10, y = 5;
console.log(x == y);          // false
console.log(x === y);         // false
console.log(x !== y);         // true
console.log(x > y);           // true
console.log(x < y);           // false
```

C. Logical Operators

```
let a = true, b = false;
console.log(a && b);          // false
console.log(a || b);          // true
console.log(!a);              // false
```

0330 2021926

D. Assignment Operators

```
let num = 10;
num += 5;              // Same as num = num + 5;
console.log(num); // 15
```

0330 2021926

Home Task 2

1. What are variables in JavaScript?
2. What are the rules for naming variables?
3. What are the different data types in JavaScript?
4. What is the difference between == and ===?
5. Convert "10" to a number using JavaScript.
6. What will be the output of 5 + "5"?
7. What is the result of true && false?
8. What is the difference between null and undefined?
9. What is the difference between alert, prompt and confirm?
10. Write a JavaScript script(JS code) to swap two numbers.

0330 2021926

Weekly Assignment 1:

Task 1: Variable Declaration & Usage

- **Declare three variables:** name, age, and isStudent.
- Assign values and print them in the console

Task 2: Type Conversion

- Convert "100" (string) to a number and "true" to a boolean.
- Print the results in the console.

Task 3: Arithmetic Operations

- Take two numbers as input from the user using prompt().
- Perform addition, subtraction, multiplication, and division.
- Show the results using alert().

0330 2021926

Task 4: Null vs Undefined

- Declare a variable with null and another with undefined.
- Print both and explain their differences in a comment.

Task 5: Logical Operators

- Take two boolean values (true or false) from the user.
- Perform AND, OR, and NOT operations and show the results.

0330 2021926

Web Development

Class No: 03

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926

Control Flow – Conditions & Loops

Conditional Statements in JavaScript:

Conditional statements in JavaScript are used to make **decisions based on conditions**. If a condition is true, a specific block of code is **executed**; otherwise, another block of code runs.

0330 2021926

If-Else Statement

This is the most basic conditional statement. If a condition is true, one block of code will execute; otherwise, another block will execute.

Example:

```
let age = 18;

if (age >= 18) {
    console.log("You are an adult.");
} else {
    console.log("You are a minor.");
}
```

0330 2021926

Else-If Ladder

The else if statement allows you to test multiple conditions.

Example:

```
let marks = 85;
if (marks >= 90) {
    console.log("Grade: A+");
} else if (marks >= 75) {
    console.log("Grade: A");
} else if (marks >= 60) {
    console.log("Grade: B");
} else {
    console.log("Grade: C");
}
```

0330 2021926

Ternary Operator

The ternary operator is a shorthand way of writing an if-else statement.

Syntax:

```
condition ? expression_if_true : expression_if_false
```

Example:

```
let age = 18;
let result = (age >= 18) ? "Adult" : "Minor";
console.log(result);                    // Adult
```

0330 2021926

Loops in JavaScript

What are Loops?

Loops allow you to repeat a block of code multiple times until a specific condition is met.

0330 2021926

1. For Loop

A for loop repeats a block of code a fixed number of times.

Syntax:

```
for (initialization; condition; increment) {
        // Code to execute
}
```

Example:

```
for (let i = 0; i < 5; i++) {
    console.log(i);                    // Prints 0 to 4
}
```

0330 2021926

While Loop

A while loop repeats a block of code as long as the condition is true.

Syntax:

```
while (condition) {
    // Code to execute
}
```

Example:

```
let i = 1;
while (i < 5) {
    console.log(i);                    // Prints 0 to 4
    i++;
}
```

0330 2021926

Do-While Loop

A do-while loop is similar to a while loop, except that it executes the block at least once, even if the condition is false.

Syntax:

```
do {
    // Code to execute
} while (condition);
```

Example:

```
let i = 0;   0
do {
    console.log(i);                // Prints 0 to 4
    i++;
} while (i < 5);
```

0330 2021926

Activity-Based Task

Activity: Number Guessing Game

Task Description:

We will create a simple **Number Guessing Game** in class.

- The program will generate a random number between 1 and 10.
- Students will use a loop to guess the number until they get it right.

0330 2021926

Code for the Activity:

```
let randomNumber = Math.floor(Math.random() * 10) + 1;
let guess;

do {
    guess = Number(prompt("Guess a number between 1 and 10:"));
    if (guess > randomNumber) {
        console.log("Too high! Try again.");
    } else if (guess < randomNumber) {
        console.log("Too low! Try again.");
    }
} while (guess !== randomNumber);

console.log("Congratulations! You guessed the correct number: " + randomNumber);
```

0330 2021926

Key Differences Between Loops

Loop Type	Description	Example Use Case
For Loop	Repeats a block a fixed number of times.	Iterating over an array.
While Loop	Repeats a block as long as a condition is true.	Validating user input.
Do-While Loop	Executes the block at least once.	Asking for input until it's valid.

Home Task 3

- What is a conditional statement?
- Write a program to check if a number is positive, negative, or zero.
- What is the difference between if-else and ternary operator?
- Write a script to print all even numbers from 1 to 10 using a for loop.
- Write a JavaScript program to check if a number is divisible by 5 using a while loop.
- Explain the difference between while and do-while loops with examples.
- Write a program that prints the multiplication table of a number using a for loop.
- Create a program to count the number of digits in a given number using a while loop.
- Write a program to print numbers from 10 to 1 using a for loop (reverse order).
- Write a script that asks the user to input numbers until the sum of all entered numbers is greater than 100.

linkedin.com/in/m-munim1 github.com/M-Munim +92 330 2021926

Web Development

Class No: 04

linkedin.com/in/m-munim1
github.com/M-Munim
+92 330 2021926

Functions, Arrays, and Objects

What is a Function?

A function is a block of code designed to perform a **specific task**. You can "call" or "invoke" the function whenever you need that task to be performed.

0330 2021926

Function Syntax

```
function functionName(parameters) {
        // Code to execute
}

functionName(“hello”)        //calling the function
```

Types of Functions in JavaScript

Function with No Parameters

A function that does not take any input values (parameters) when called. It simply performs its task and may or may not return a value.

Example:

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

0330 2021926

Function with Parameters

A function that takes input values (called parameters) when called. The parameters allow the function to work with different data.

Example:

```
function greet(name) {
    console.log("Hello, " + name + "!");
}
greet("Ali");                    // Output: Hello, Ali!
greet(“rabia”);
greet(“Abdullah”);
```

Function with Return Value

A function that sends a result back to the code that called it. The return keyword is used to give back the result.

Example:

```
function addNumber(a, b) {          //a,b are parameters here
    return a + b;
}
let sum = addNumber(5, 10);
console.log("Sum:", sum);                  // Output: Sum: 15
```

0330 2021926

Arrow Functions in JavaScript

Arrow functions provide a shorter syntax to define functions. They are defined using the => (arrow) symbol.

Syntax:

```
const functionName = (parameters) => {
        // Code to execute
};
```

Example (Arrow Function with No Parameters):

```
const greet = () => {
    console.log("Hello, World!");
};
greet();                    // Output: Hello, World!
```

0330 2021926

Example (Arrow Function with Parameters):

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

Simplified Arrow Function (Single Line):

If the function has only one expression, you can omit the braces {} and the return keyword:

```
const multiply = (a, b) => a * b;
console.log(multiply(5, 3));                // Output: 15
```

0330 2021926

Arrays in JavaScript

What is an Array?

An array is a collection of multiple values stored in a single variable. Each value in an array is called an "element.“

Array Syntax

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

Example (Declaring an Array):

```
let fruits = ["Apple", "Banana", "Mango"];
console.log(fruits);                // Output: ["Apple", "Banana", "Mango"]
```

0330 2021926

Accessing Array Elements

You can access an array element by its index (position). Indexes start from 0.

Example:

```
let fruits = ["Apple", "Banana", "Mango"];
console.log(fruits[0]);                    // Output: Apple
```

Common Array Methods

- **push():** Adds an element to the end of the array.

```
let fruits = ["Apple", "Banana"];
fruits.push("Mango");
console.log(fruits);                       // ["Apple", "Banana", "Mango"]
```

- **pop():** Removes the last element.

```
let fruits = ["Apple", "Banana"];
fruits.pop();
console.log(fruits);          // ["Apple"]
```

- **shift():** Removes the first element.

```
let fruits = ["Apple", "Banana"];
fruits.shift();
console.log(fruits);          // ["Banana"]
```

0330 2021926

- **unshift():** Adds an element to the beginning of the array.

```
let fruits = ["Banana"];
fruits.unshift("Apple");
console.log(fruits);                // ["Apple", "Banana"]
```

0330 2021926

Objects in JavaScript

What is an Object?

An object is a collection of key-value pairs. Each key represents a property, and the value represents the property's data.

Object Syntax

```
let objectName = {
    key1: value1,
    key2: value2,
    key3: value3
};
```

0330 2021926

Example:

```
let student = {
    name: "Ali",
    age: 20,
    isEnrolled: true
};
console.log(student.name);          // Output: Ali
```

0330 2021926

Accessing and Modifying Object Properties

- You can access object properties using dot notation or bracket notation.
- You can also modify or add new properties.

Example:

```
let student = { name: "Ali", age: 20 };
console.log(student.name);          // Access property using dot notation
student.age = 21;                   // Modify property
student.grade = "A";                // Add new property
console.log(student);
```

0330 2021926

Activity-Based Task: Student Report Generator

In this task, we will build a Student Report Generator using functions and objects.

Task Description:

1. Create an object student with properties like name, age, and marks (an array of subject marks).
2. Write a function calculateAverage() to calculate the average marks.
3. Display the student details and average marks.

0330 2021926

Code for Activity:

```
let student = {
   name: "Ali",
   age: 20,
   marks: [85, 90, 78, 88, 92]
};

function calculateAverage(marks) {
   let sum = 0;
   for (let i = 0; i < marks.length; i++) {
    sum += marks[i];
   }
   return sum / marks.length;
}

console.log("Student Name:", student.name);
console.log("Student Age:", student.age);
console.log("Average Marks:",
calculateAverage(student.marks));
```

0330 2021926

Home Task 4

1. What is a function in JavaScript? Create a function to calculate the square of a number.
2. Write a function that takes two numbers as input and returns their product.
3. Declare an array of colors. Add a new color to the end and remove the first color.
4. Create an array of numbers and find the sum of all numbers using a loop.
5. Create an object car with properties make, model, and year. Print each property.
6. Write a function isEven() that checks if a number is even or odd.
7. Create an array of student names. Use a loop to print each name.
8. Write a function that takes an array of numbers and returns the largest number.
9. Create an object person with properties firstName and lastName. Write a function to display the full name.
10. Write a program to reverse the elements of an array without using any built-in methods.

linkedin.com/in/m-munim1 github.com/M-Munim +92 330 2021926

Web Development

Class No: 05

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926

Introduction to the DOM

What is the DOM?

The DOM (Document Object Model) is a representation of the structure of an HTML document as a tree of nodes. It allows JavaScript to access, manipulate, and modify the structure and content of web pages.

DOM Structure:

HTML elements are represented as nodes (e.g., <html>, <body>, , etc.). Text content is also represented as text nodes.

0330 2021926

Example (HTML and its DOM Structure):

```
<body>
   <h1 id="heading">Hello</h1>
   <p>This is a paragraph.</p>
</body>
```

DOM Tree Structure:

0330 2021926

body
- h1 (with id heading)
- p (Text: "This is a paragraph.")

Parts of the Document

- console.log(document.title); // Shows the page title
- console.log(document.URL); // Shows the page URL
- console.log(document.body); // Shows the body element
- console.log(document.head); // Shows the head element
- console.log(document.images); // Shows all images on the page
- console.log(document.links); // Shows all links on the page

Modify the Document in Real-Time

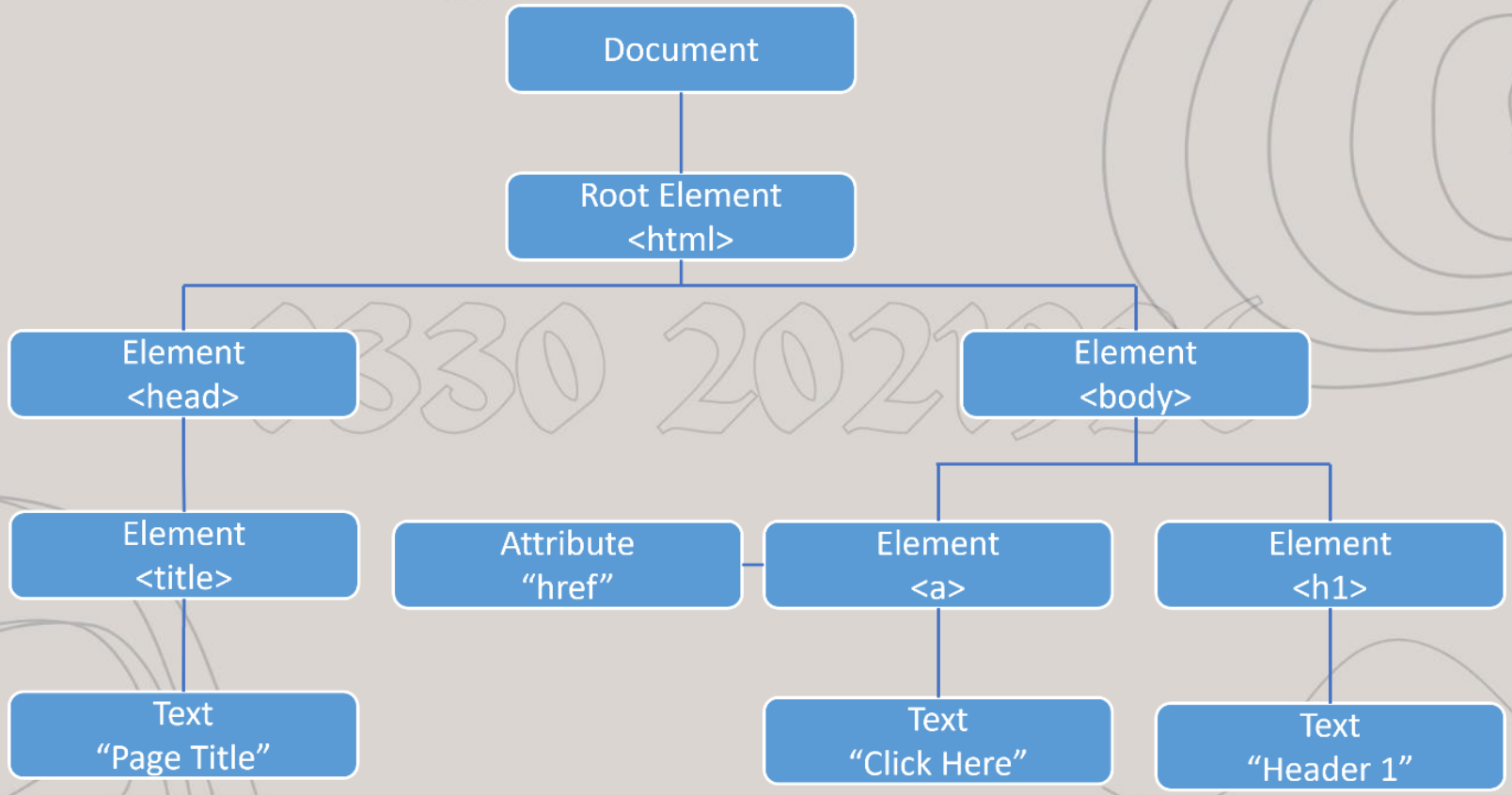
- document.title = "Hello, Students!"; // Changes the title dynamically
- document.body.style.backgroundColor = "lightblue"; // Changes background color
- document.body.innerHTML += "<h2>Welcome to JavaScript DOM!</h2>"; // Adds a heading

Count Elements/Tag on the Page

- console.log(`This page has ${document.body.children.length} elements.`);

0330 2021926

Web Development



linkedin.com/in/m-munim1
github.com/M-Munim
+92 330 2021926

Accessing DOM Elements

We can use JavaScript methods to access elements and manipulate them.

- document.getElementById()
  Retrieves an element by its id.

html

```
<h1 id="heading">Hello, World!</h1>
```

javascript

```
let heading = document.getElementById(“heading");
console.log(heading.innerText);                    // Output: Hello, World!
```

0330 2021926

- document.getElementsByClassName()
  Retrieves elements by their class name (returns an HTMLCollection).

html

```
<div class="box">Box 1</div>
<div class="box">Box 2</div>
```

javascript

```
let boxes = document.getElementsByClassName("box");
console.log(boxes[1].innerText);                    // Output: Box 1
```

0330 2021926

- document.getElementsByTagName()
  Retrieves all elements with the specified tag name.

html

```
<p>Paragraph 1</p>
<p>Paragraph 2</p>
```

javascript

```
let paragraphs = document.getElementsByTagName("p");
console.log(paragraphs[1].innerText);                    // Output: Paragraph 2
```

0330 2021926

- querySelector()
  Selects the first element that matches the CSS selector.

html

```
<h1 id="heading“ class=“head”>Hello!</h1>
```

javascript

```
let heading = document.querySelector("#heading");
console.log(heading.innerText);                    // Output: Hello!
```

0330 2021926

- querySelectorAll()
  Selects all elements that match the CSS selector (returns a NodeList).

html

```
<div class="box">Box 1</div>
<div class="box">Box 2</div>
```

javascript

```
let boxes = document.querySelectorAll(".box");
console.log(boxes[0].innerHTML);
```

0330 2021926

Modifying the DOM

Change Text Content:

Using innerText or innerHTML, we can change the text inside an element.

html

```
<h1 id="heading">Original Text</h1>
```

javascript

```
let heading = document.getElementById("heading");
heading.innerText = "Updated Text";                    // Changes text to "Updated Text"
```

0330 2021926

Change Attributes:

Using setAttribute(), we can modify the attributes of an element.

html

```
<img id="image" src="old-image.jpg“ alt=“img of car” >      // old car image
```

javascript

```
let img = document.getElementById("image");
img.setAttribute("src", "new-image.jpg");            // Changes the image source
img.setAttribute("alt", “image of new car");
```

0330 2021926

Change Styles:

Using the style property, we can modify the element’s CSS.

javascript

```
heading.style.color = "blue";          // Changes text color to blue
heading.style.fontSize = "24px";       // Changes font size
```

0330 2021926

Advanced DOM Manipulation and Events

Creating and Adding Elements

- createElement(): Creates a new element in JavaScript.
- appendChild(): Adds the newly created element to an existing element in the DOM.

Example:

```
<ul id="list"></ul>
```

javascript

```
let list = document.getElementById("list");
  // Creating a new list item
let newItem = document.createElement("li");
newItem.innerText = "New Item";
  // Appending the new item to the list
list.appendChild(newItem);
```

0330 2021926

Removing and Replacing Elements

- removeChild(): Removes a specified child element from its parent.
- replaceChild(): Replaces an existing child element with another.

HTML:

```
<ul id="list">
  <li id="item">Item 1</li>
</ul>

<button id="removeBtn">Remove Item</button>
<button id="replaceBtn">Replace Item</button>
```

0330 2021926

```
let list = document.getElementById("list");
let item = document.getElementById("item");
let removeBtn = document.getElementById("removeBtn");
let replaceBtn = document.getElementById("replaceBtn");

removeBtn.addEventListener("click", function () {                    // Removing an Element
    if (item) {
  list.removeChild(item);
   }
});

replaceBtn.addEventListener("click", function () {                   // Replacing an Element
 let newItem = document.createElement("li");
 newItem.innerText = "New Item";
   if (item) {
      list.replaceChild(newItem, item);
 }
});
```

0330 2021926

Adding Events Using addEventListener()

Events allow us to add **interactivity** to our webpages. The addEventListener() method listens for user interactions such as clicks, key presses, and mouse movements.

HTML:

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

JavaScript:

```
let button = document.getElementById("btn");

// Adding a click event listener
button.addEventListener("click", function () {
    alert("Button was clicked!");
});
```

0330 2021926

HTML Form Validation

Form validation ensures users enter valid data before submitting a form. JavaScript helps check if the required fields are filled correctly.

Example:

```
<form id="myForm">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required>
   <br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
   <br>
  <button type="submit">Submit</button>
</form>
<p id="error-message" style="color:red;"></p>
```

0330 2021926

```
let form = document.getElementById("myForm");
let errorMessage = document.getElementById("error-message");

// Adding submit event listener
form.addEventListener("submit", function (event) {
    let name = document.getElementById("name").value;
    let email = document.getElementById("email").value;

    if (name === "" || email === "") {
     errorMessage.innerText = "All fields are required!";
event.preventDefault();                          // Prevent form submission
 } else {
    alert("Form submitted successfully!");
 }
});
```

0330 2021926

Web Development

Class No: 07

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926

JavaScript Events & ES6 Features

JavaScript Events

JavaScript events allow interaction with users when they perform actions such as clicking, hovering, or pressing keys. Events make web pages dynamic and interactive.

Common JavaScript Events:

- onclick - Triggers when an element is clicked.
- onmouseover - Triggers when the mouse hovers over an element.
- onkeydown - Triggers when a key is pressed.

Handling Events in JavaScript

1. Using HTML Attribute (Not Recommended):

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

This method is not recommended because it makes the HTML code less readable and harder to manage.

2. Using JavaScript Event Listener (Recommended):

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

Js:

```
let btn = document.getElementById("myButton");
btn.addEventListener("click", function() {
alert("Button Clicked!");    });
```

This method is preferred as it keeps the JavaScript code separate from HTML, making it easier to manage and maintain.

0330 2021926

ES6 Features (Modern JavaScript)

1. Let & Const (Block-scoped variables):

- let allows reassignment.
- const does not allow reassignment.

```
let x = 10;
x = 15;                 // Allowed

const y = 20;
y = 25;                 // Error! Cannot reassign a constant variable.
```

0330 2021926

2. Template Literals (String Interpolation):

Allows embedding variables directly into strings using backticks (`).

```
let name = "John";
console.log(`Hello, ${name}!`);          // Output: Hello, John!
```

3. Spread & Rest Operators:

- Spread (...): Expands arrays or objects.
- Rest (...): Collects multiple values into an array.

```
let arr = [1, 2, 3];
let newArr = [...arr, 4, 5];
console.log(newArr); // Output: [1, 2, 3, 4, 5]
```

0330 2021926

Rest Operator Example:

```
function showNames(...names) {
  console.log(names);
}
showNames("Alice", "Bob", "Charlie");              // Output: ["Alice", "Bob", "Charlie"]
```

0330 2021926

4. Destructuring Assignment:

Extract values from arrays or objects easily.

```
let person = {
  name: "Alice",
  age: 25
  };

let { name, age } = person;
console.log(name, age);                    // Output: Alice 25
```

0330 2021926

HTML & JavaScript Example: Interactive Button

```
<body>
    <button id="myButton">Click Me</button>
    <p id="message"></p>

    <script>
        let btn = document.getElementById("myButton");
        let message = document.getElementById("message");

        btn.addEventListener("click", function() {
            message.textContent = "Button was clicked!";
        });
    </script>
</body>
```

0330 2021926

2: Display List of Names

```
<body>
  <button id="showNames">Show Names</button>
  <ul id="nameList"></ul>
  <script>
    function displayNames(...names) {
      let list = document.getElementById("nameList");
      list.innerHTML = "";
      names.forEach(name => {
        let li = document.createElement("li");
        li.textContent = name;
        list.appendChild(li);
      });
    }
    document.getElementById("showNames").addEventListener("click", function() {
      displayNames("Alice", "Bob", "Charlie");
    });
  </script>
</body>
```

0330 2021926

Web Development

Class No: 08

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926

JavaScript Async/Await & API Handling

Understanding Asynchronous JavaScript

- JavaScript is Single-Threaded
- Imagine JavaScript as a chef in a small kitchen who can cook only one dish at a time (single-threaded).
- If the chef waits for water to boil before cutting vegetables, cooking will take forever!
- Instead, the chef multitasks—they put water to boil, then start chopping, and later check the water.

This is how asynchronous JavaScript works!

0330 2021926

Why Do We Need Asynchronous JavaScript?

JavaScript executes one operation at a time, but some tasks take time, like:

- Fetching data from a server
- Reading a file
- Waiting for a timer

If JavaScript waited for each task to finish, the whole program would freeze! Instead, it delegates long tasks and continues running other code.

0330 2021926

What is Async/Await?

Async/Await is a way to handle **asynchronous** code in JavaScript, making it easier to read and write.

Why do we need it?

JavaScript runs code line by line, but sometimes we need to wait for tasks like:

- Fetching data from a server (API calls)
- Reading a file
- Waiting for a timer

Earlier, we used callbacks (which led to "callback hell") or promises (.then()), but async/await makes it much cleaner!

0330 2021926

Breaking it down:

- async keyword

  When you add async before a function, it automatically returns a promise.

```
async function getData() {
    return "Hello, World!";
}
getData().then(console.log);                    // Output: Hello, World!
```

0330 2021926

await keyword

- await pauses the function until the promise is resolved.
- It must be used inside an async function.

```
async function fetchData() {
    let response = await fetch("https://api.example.com/data");
    let data = await response.json();
    console.log(data);              // This runs **only after** the data is fetched
}
fetchData();
```

03302021926

Example: Without and With Async/Await

- ❌ Without Async/Await (Using .then())

```
fetch("https://api.example.com/data")
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.log(error));
```

03302021926

With Async/Await (Cleaner & Easier)

```
async function getData() {
    try {
        let response = await fetch("https://api.example.com/data");
        let data = await response.json();
        console.log(data);
    } catch (error) {
        console.log(error);
    }
}
getData();
```

0330 2021926

Working with APIs using Fetch & Async/Await

What is an API?

An API (Application Programming Interface) allows different applications to communicate with each other.

For example:

- A weather app gets live weather updates from an API.
- A shopping app fetches product details from an API.

0330 2021926

What is fetch?

JavaScript provides the fetch() function to get data from APIs.

- It sends a request to the API.
- The API sends back data (usually in JSON format).
- We then process and use the data in our program.

0330 2021926

Fetching Data from an API

Example: Fetching Random Users from an API

```
async function fetchUser() {
        try {
            let response = await fetch("https://randomuser.me/api/"); // Fetch data from a public API

            let data = await response.json();              // Convert response to JSON format

            let user = data.results[0];                    // Extract user details
            console.log("User Name:", user.name.first, user.name.last);
            console.log("Email:", user.email);
            console.log("Country:", user.location.country);
        } catch (error) {
            console.log("Error fetching data:", error);
        }
    }

    fetchUser();
```

CSD 2021926

Breaking Down the Code:

- fetch("https://randomuser.me/api/") → Sends a request to the API.
- await response.json() → Converts the response into JSON format.
- Extracts user data (e.g., name, email, country).
- Handles errors using try...catch.

0330 2021926

How the API Response Looks (JSON Format)

When we call the API, it returns data like this:

JSON:

```
{
    "results": [
      {
        "name": { "first": "John", "last": "Doe" },
        "email": "johndoe@example.com",
        "location": { "country": "USA" }
      }
    ]
}
```

We extract name.first, name.last, email, and location.country from this JSON.

0330 2021926

Example: Fetching API Data and Displaying in HTML

```
<body>
  <button id="fetchBtn">Fetch User</button>
  <p id="userData"></p>

  <script>
    document.getElementById("fetchBtn").addEventListener("click", async function() {
      try {
        let response = await fetch("https://jsonplaceholder.typicode.com/users/1");
        let data = await response.json();
        document.getElementById("userData").textContent = `Name: ${data.name}, Email: ${data.email}`;
      } catch (error) {
        document.getElementById("userData").textContent = "Failed to fetch data";
      }
    });
  </script>
</body>
```

Submission Notes

Capture Evidence:

- Take screenshots or record a short video for each task.
- Ensure your evidence clearly shows the completed task.

Upload Tasks:

- Upload each task as a separate folder or file in your GitHub repository.

LinkedIn Post:

- Create a short post summarizing what you learned and how you completed the tasks.
- Highlight your key takeaways.
- Mention the following account in your post:
- **Muhammad Munim**

0330 2021926

Thank You

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926