**Lecture 1: JavaScript Basics, Variables, Data Types**

**Part 1. JavaScript Introduction**

JavaScript is a **scripting language** used to create **dynamic and interactive** websites. It can:

* Change HTML content
* Validate forms
* Respond to events (click, hover)
* Work with APIs and much more!

**Embedding JavaScript**

There are **3 ways** to use JavaScript in HTML:

1. **Inline JavaScript:**

<button onclick="alert('Hello!')">Click Me</button>

1. **Internal JavaScript:**

<!DOCTYPE html>

<html>

<head>

<script>

function sayHi() {

alert('Hello from internal script!');

}

</script>

</head>

<body>

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

</body>

</html>

1. **External JavaScript:**

<!-- index.html -->

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

// script.js

function greet() {

alert('Hello from external file!');

}

**Script Placement: <script> tag, defer, async**

Normally JavaScript is placed:

* At the **bottom of body** tag (so page loads first)

<body>

<!-- HTML content -->

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

</body>

Or you can place in <head> with:

* defer – waits until HTML is fully parsed:

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

* async – downloads and runs immediately:

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

**Part 2. Variables and Data Types**

**Declaration Methods:**

1. var – Old way, **function scoped**
2. let – Modern way, **block scoped**
3. const – For constants (cannot be changed)

var a = 10;

let b = 20;

const c = 30;

**Primitive Data Types:**

| **Data Type** | **Example** |
| --- | --- |
| String | "Hello" |
| Number | 10, 10.5 |
| Boolean | true, false |
| Null | let x = null; |
| Undefined | let y; |
| Symbol | let s = Symbol(); |

**Non-Primitive Data Types:**

1. **Object** – Collection of key-value pairs

let student = { name: "Ali", age: 21 };

1. **Array** – List of values

let colors = ["red", "green", "blue"];

1. **Function** – Block of reusable code

function greet() {

alert("Hello!");

}

**Part 3. Operators in JavaScript**

**Arithmetic Operators:**

| **Operator** | **Description** |
| --- | --- |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulus (remainder) |

let a = 10, b = 3;

console.log(a + b); // 13

console.log(a % b); // 1

**Comparison Operators:**

| **Operator** | **Meaning** |
| --- | --- |
| == | Equal (type conversion) |
| === | Equal (strict type check) |
| != | Not equal |
| !== | Not equal (strict) |
| > | Greater than |
| < | Less than |

5 == "5" // true

5 === "5" // false

**Logical Operators:**

| **Operator** | **Meaning** |
| --- | --- |
| && | AND |
| ` |  |
| ! | NOT |

true && false // false

true || false // true

!true // false

**Part 4. Write a script to calculate the sum of user inputs and display the result.**

**Lecture 2: Control Structures, Arrays, Loops**

**1. Control Structures**

Control structures help us **make decisions** in code.

**if, else, else if**

let marks = 85;

if (marks > 90) {

console.log("Grade A+");

} else if (marks >= 80) {

console.log("Grade A");

} else {

console.log("Keep trying!");

}

**switch Statement**

let day = 3;

switch (day) {

case 1:

console.log("Monday");

break;

case 2:

console.log("Tuesday");

break;

case 3:

console.log("Wednesday");

break;

default:

console.log("Other Day");

}

**Ternary Operator: (condition) ? true : false**

Short form of if-else.

let age = 18;

let message = (age >= 18) ? "Adult" : "Minor";

console.log(message); // Output: Adult

**2. Working with Arrays**

Arrays store **multiple values** in a single variable.

**Creating Arrays**

const fruits = ['apple', 'banana', 'mango'];

console.log(fruits[0]); // Output: apple

**Common Array Methods**

| **Method** | **Description** |
| --- | --- |
| push() | Add item to end |
| pop() | Remove item from end |
| unshift() | Add item to start |
| shift() | Remove item from start |
| map() | Transform each item and return new array |
| filter() | Return array items that match condition |
| reduce() | Reduce array to a single value |

let numbers = [10, 20, 30];

// push and pop

numbers.push(40); // [10, 20, 30, 40]

numbers.pop(); // [10, 20, 30]

// shift and unshift

numbers.shift(); // [20, 30]

numbers.unshift(5); // [5, 20, 30]

// map

let doubled = numbers.map(n => n \* 2); // [10, 40, 60]

// filter

let filtered = numbers.filter(n => n > 10); // [20, 30]

// reduce

let sum = numbers.reduce((acc, val) => acc + val, 0); // 55

**3. Loops and Iteration**

Loops help repeat actions.

**For Loop**

for (let i = 0; i < 5; i++) {

console.log(i); // 0 to 4

}

**for...of (for arrays)**

let fruits = ['apple', 'banana', 'mango'];

for (let fruit of fruits) {

console.log(fruit);

}

**for...in (for objects)**

let student = { name: "Ali", age: 20 };

for (let key in student) {

console.log(key + ": " + student[key]);

}

**while Loop**

let i = 0;

while (i < 3) {

console.log("Number: " + i);

i++;

}

**do...while Loop**

let j = 0;

do {

console.log("Value: " + j);

j++;

} while (j < 3);

**Loop Control: break, continue**

for (let i = 1; i <= 5; i++) {

if (i === 3) continue; // skips 3

if (i === 5) break; // stops at 5

console.log(i);

}

// Output: 1, 2, 4

**Activity 1: Develop a dynamic list renderer using arrays and loops.**

**Activity 2:**  
A school wants a simple program to check student marks and print their grades. You need to write a small JavaScript script that looks at each student's marks and shows their grade.

**Tasks:**

1. Create an array with 3 students. Each student should have a name and marks.  
   Example:

const students = [

{ name: "Ali", marks: 85 },

{ name: "Sara", marks: 72 },

{ name: "Ahmed", marks: 58 }

];

1. Use a for loop or for...of loop to print each student's name and grade:
   * If marks ≥ 80 → Grade A
   * If marks ≥ 60 → Grade B
   * Otherwise → "Fail"
2. Add a new student to the array using the push() method.
3. Remove the first student using the shift() method.
4. Use the filter() method to display only the students who passed (Grade A or B).

**Expected Output Example:**

Ali: Grade A

Sara: Grade B

Ahmed: Fail

Pass Students:

Ali

Sara

**Lecture 3: Functions, Scope, Hoisting**

**What is a Function?**

A **function** is a block of code that does something.  
We can **reuse** it again and again, so we don't repeat code.

**1. Function Declaration**

function greet() {

console.log("Hello!");

}

greet(); // Output: Hello!

**Explanation:**

* This is a normal function.
* We define it using the function keyword.
* We call it using its name: greet().

**Function Expression**

const greet = function() {

console.log("Hello from expression!");

};

greet(); // Output: Hello from expression!

**Explanation:**

* We create a function and save it in a variable.
* This function doesn’t have a name (anonymous).
* It works the same way when called.

**Arrow Function (ES6+)**

const greet = () => {

console.log("Hello from arrow!");

};

greet(); // Output: Hello from arrow!

**Explanation:**

* This is a shorter way to write a function.
* Mostly used for small tasks or inside other functions (like in loops or events).

**Function Parameters and Return Values**

**Parameter** = input  
**Return** = output

function greet(name) {

return "Hello " + name;

}

let message = greet("Ali");

console.log(message); // Output: Hello Ali

**Explanation:**

* name is the input (parameter).
* The function returns a message using that name.
* The result is saved in message.

**Scope in JavaScript**

**Scope** means **where** you can use or access a variable.

**Global Scope**

let x = 5;

function showX() {

console.log(x); // Output: 5

}

showX();

**Explanation:**

* x is declared outside the function.
* It can be used **anywhere** in the code.

**Local Scope (Function Scope)**

function demo() {

let y = 10;

console.log(y); // Output: 10

}

demo();

console.log(y); // Error: y is not defined

**Explanation:**

* y only exists **inside** the demo() function.
* It **cannot** be used outside the function.

**Block Scope (let and const)**

if (true) {

let a = 10;

const b = 20;

console.log(a, b); // Output: 10 20

}

console.log(a); // Error: a is not defined

**Explanation:**

* let and const only work **inside the block** (inside {}).

**Function Scope (with var)**

function test() {

if (true) {

var z = 50;

}

console.log(z); // Output: 50

}

test();

**Explanation:**

* var does **not** follow block scope.
* It is available in the whole **function**.

**Hoisting in JavaScript**

**Hoisting** means that JavaScript **moves declarations to the top** of the code before running it.

**Variable Hoisting (with var)**

console.log(a); // Output: undefined

var a = 10;

**Explanation:**

* JavaScript moves var a; to the top.
* But it does **not** move the value 10.

**No Hoisting with let and const**

console.log(b); // ReferenceError

let b = 20;

**Explanation:**

* let and const are **not accessible** before they are declared.
* This gives an error, not undefined.

**Function Hoisting (Declaration)**

sayHi(); // Output: Hello!

function sayHi() {

console.log("Hello!");

}

**Explanation:**

* Full function is moved to the top.
* So it works **even before** it is written.

**Function Expressions are NOT Hoisted**

sayHello(); // TypeError

const sayHello = function() {

console.log("Hi!");

};

**Explanation:**

* JavaScript only moves the variable sayHello, not the function value.
* So calling it before the definition gives an error.

**Activity: Simple Calculator Using Functions**

<input type="number" id="num1" />

<input type="number" id="num2" />

<button onclick="add()">Add</button>

<p id="result"></p>

<script>

function add() {

const a = Number(document.getElementById("num1").value);

const b = Number(document.getElementById("num2").value);

const sum = a + b;

document.getElementById("result").textContent = "Result: " + sum;

}

</script>

**How it works:**

* You enter two numbers.
* When you click the button, the function add() runs.
* It adds the numbers and shows the result.