JavaScript Introduction Theory.

Q.**1 What is JavaScript? Explain the role of JavaScript in web development.**

Ans:

- **JavaScript** is a high-level, lightweight, and interpreted programming language.
- It is mainly used to make web pages interactive and dynamic.
- JavaScript runs in the browser (client-side), but it can also run on the server side using environments like Node.js.
-

Q.2 **How is JavaScript different from other programming languages like Python or Java?**

Ans :

- JavaScript is primarily for **front-end web development**, while Python and Java are general-purpose languages often used for backend, desktop, and other domains.

Q.3 Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?

Ans :

- The <script> tag is used in HTML to embed or reference JavaScript code.
- You can write JavaScript inline within the <script> tag, or link to an external file.
- Use the src attribute inside the <script> tag to link an external file.

- **Variables and Data Types Theory.**

Q.1 What are variables in JavaScript? How do you declare a variable using var, let, and const?

Ans :

- A variable is like a named container used to store data that can be used or changed later in the program.
- In JavaScript, you can declare variables using var, let, or const.

```
var city = "Delhi";     // using var

let score = 90;         // using let

const country = "India"; // using const
```

Q.2 Explain the different data types in JavaScript. Provide examples for each.

Ans : 1. **Primitive Data Types**

- String: Textual data.
  - Examples

  ```
  let name = "Alice";
  ```

- Number: Integers or floating points.
  - Examples

```
let age = 25;

let price = 99.99;
```

- Boolean: True/False values.
  - Examples

let isLoggedIn = true;

- Undefined: A variable declared but not assigned.
    - Examples

let user;

- Null: Intentionally empty or “nothing”.
    - Examples

let selectedItem = null;

- Symbol: Unique identifiers (advanced usage).
    - Examples

let id = Symbol('unique');

- BigInt: For very large integers.
    - Examples

let bigNum = 123456789012345678901234567890n**;**

2. Non-Primitive (Reference) Data Types

- Object: Key–value pairs.
    - Examples

let person = { name: "John", age: 30 };

- Array: Ordered collection of values.
    - Examples

let fruits = ["Apple", "Banana", "Mango"];

- Function: Reusable block of code.
    - Examples

function greet() { console.log("Hello!"); **}**

Q.3 What is the difference between undefined and null in JavaScript?

Ans : **undefined**

- A variable has been declared but **not assigned** a value.
- undefined is its own type.
- Set automatically by JavaScript.

Null

- A variable has been explicitly assigned an empty or “no value”.
- null is of type object (due to a historical bug in JS).
- Set manually by the programmer.

- **JavaScript Operators Theory**

Q.1 What are the different types of operators in JavaScript? Explain with examples.

Ans :

1. Arithmetic Operators

Used to perform basic mathematical operations.

Operator	Description	Example
+	Addition	`5 + 3 // 8`
-	Subtraction	`5 - 3 // 2`
*	Multiplication	`5 * 3 // 15`
/	Division	`6 / 3 // 2`
%	Modulus (remainder)	`5 % 2 // 1`

2. Assignment Operators

Used to assign values to variables.

Operator	Description	Example
=	Assign	x = 10
+=	Add and assign	x += 5; // x = x + 5
-=	Subtract and assign	x -= 2; // x = x - 2
*=	Multiply and assign	x *= 3; // x = x * 3
/=	Divide and assign	x /= 2; // x = x / 2

3. Comparison Operators

Used to compare two values and return a Boolean (true or false).

Operator	Description	Example	Result
==	Equal to (loose)	5 == '5'	true
===	Equal and same type	5 === '5'	false
!=	Not equal	5 != 3	true
!==	Not equal or not same type	5 !== '5'	true
>	Greater than	6 > 3	true
<	Less than	3 < 6	true
>=	Greater than or equal to	5 >= 5	true
<=	Less than or equal to	4 <= 5	true

Operator **Description** **Example Result**

4. Logical Operators

Used to combine multiple conditions.

Operator	**Description**	**Example**	**Result**
&&	AND (both true)	(5 > 3 && 2 > 1)	true
||	OR (at least one true)	`(5 > 3	
!	NOT (reverse)	!(5 > 3)	false

Q.2 What is the difference between == and === in JavaScript?

Ans : == (Loose Equality)

- Performs **type coercion** (converts values to same type before comparison).
- Checks only **value** equality.
- 5 == '5' // true

=== (Strict Equality)

- Does not perform type coercion.
- Checks **value and type** equality.
- 5 === '5' // false

- ## Control Flow (If-Else, Switch) Theory

Q.1 What is control flow in JavaScript? Explain how if-else statements work with an example.

Ans : What is Control Flow?

Control flow in JavaScript refers to the **order in which the code is executed** in a program.
Normally, JavaScript runs code **from top to bottom**, but with control flow statements like if, else, switch, for, and while, you can make the program **decide different paths** based on conditions.

How if-else works?

An if statement checks a condition:

- If the condition is **true**, the code inside the if block runs.
- If it's **false**, you can use an else block to run alternative code.

Example

```
let marks = 75;

if (marks >= 90) {

  console.log("Grade: A");

} else if (marks >= 70) {

  console.log("Grade: B");

} else {

  console.log("Grade: C");

}

// Output: Grade: B
```

Q.2 Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?

Ans : How a switch works:

A switch statement allows you to compare a single value against multiple possible matches (cases).

- Each case is checked.
- When a match is found, that block runs until a break is reached.
- If no cases match, the default block runs.

Syntax

```
 switch(expression) {
  case value1:
    // Code if expression === value1
    break;
  case value2:
    // Code if expression === value2
    break;
  default:
    // Code if no case matches
}
```

Example

```
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("Invalid day");
}
// Output: Wednesday
```

- **Loops (For, While, Do-While) Thepory**

Q.1 Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.

Ans : 1. for loop

- Used when you know **exactly how many times** you want to run the code.
- Syntax: for(initialization; condition; increment/decrement)

Example

```
        for (let i = 1; i <= 5; i++) {
  console.log("For loop count: " + i);
}
// Output: 1 2 3 4 5
```

2. while loop

- Used when you want to repeat a block **as long as a condition is true**.
- You must manually update the variable inside the loop.

```
let i = 1;
while (i <= 5) {
  console.log("While loop count: " + i);
  i++;
}
// Output: 1 2 3 4 5
```

3. do-while loop

- Similar to while, **but runs the block at least once** before checking the condition.

```
let j = 1;

do {

  console.log("Do-While loop count: " + j);

  j++;

} while (j <= 5);

// Output: 1 2 3 4 5
```

Q.2 What is the difference between a while loop and a do-while loop?

Ans : while loop

- Checks the condition **before** running the loop.
- May not run even once if the condition is false initially.

do-while loop

- Checks the condition after running the loop.
- Runs **at least once** even if the condition is false.

- **Functions Theory**

Q.1What are functions in JavaScript? Explain the syntax for declaring and calling a function.

Ans : Definition:
A function in JavaScript is a reusable block of code that performs a specific task. Instead of repeating code, you write it once in a function and call it wherever needed.

Syntax

```
// Function declaration

function greet() {

  console.log("Hello, welcome to JavaScript!");

}


// Calling the function

greet();
```

Output:

```
Hello, welcome to JavaScript!
```

Q.2 What is the difference between a function declaration and a function expression?

Ans : Function Declaration

- Declared with function keyword and a name.
- **Hoisted** – can be called before it appears in the code.

Function Expression

- Assigned to a variable (can be anonymous).
- **Not hoisted** – can only be called after the expression is defined.

Example

```
greet(); //  works because of hoisting

function greet() {

  console.log("Hello!");

}


sayHi(); //  Error: Cannot access before initialization

const sayHi = function() {

  console.log("Hi there!");

};
```

Q.3 Discuss the concept of parameters and return values in functions.

Ans : Parameters

- Inputs you pass into a function.
- Declared inside parentheses () when defining the function.
- You provide arguments when calling the function.

Example

```
function add(a, b) {
  console.log(a + b);
}

add(5, 3); // Output: 8
```

Return Values

- A function can send back a value using return.
- Without return, the function returns undefined.

Example

```
function multiply(x, y) {
  return x * y; // returns result
}

let result = multiply(4, 5);
console.log(result); // Output: 20
```

- **Arrays Theory**

Q.1 What is an array in JavaScript? How do you declare and initialize an array?

Ans : Definition:
An array in JavaScript is a special variable that can store multiple values (elements) in a single variable.
It’s used when you want to group related data together.

How to declare and initialize:

```
// Declare and initialize with values

let fruits = ["Apple", "Banana", "Mango"];


// Declare an empty array and add later

let numbers = [];

numbers[0] = 10;

numbers[1] = 20;

numbers[2] = 30;


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

console.log(numbers);  // Output: [10, 20, 30]
```

Q.2 : Explain the methods push(), pop(), shift(), and unshift() used in arrays.

Ans : These are array methods to add or remove elements:

Adds one or more elements to the **end** of the array.

```
let fruits = ["Apple", "Banana"];

fruits.push("Mango");

console.log(fruits);

// Output: ["Apple", "Banana", "Mango"]
```

Removes the **last element** from the array and returns it.

```
let fruits = ["Apple", "Banana", "Mango"];

let lastFruit = fruits.pop();

console.log(fruits);

// Output: ["Apple", "Banana"]

console.log(lastFruit);

// Output: "Mango"
```

Removes the **first element** from the array and returns it.

```
let fruits = ["Apple", "Banana", "Mango"];

let firstFruit = fruits.shift();

console.log(fruits);

// Output: ["Banana", "Mango"]

console.log(firstFruit);

// Output: "Apple"
```

Adds one or more elements to the **beginning** of the array.

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

- **Objects Theory**

Q.1 What is an object in JavaScript? How are objects different from arrays?

Ans : What is an object?

An object in JavaScript is a collection of data in the form of key–value pairs.
Each property (key) has a value.
It is used to store related information together.

Example :

```
let student = {
  name: "Rahul",
  age: 21,
  course: "Full Stack"
};
```

Objects

Stores data in key–value pairs

Access by **key name**

{name: "Rahul", age: 21}

Arrays

Stores data in a list (ordered by index)

Access by **index number**

["Rahul", 21]