JavaScript Introduction

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

Ans.

Java Script:-

- ✓ JavaScript is a **high-level, interpreted programming language** primarily used to create interactive and dynamic content on websites.

- ✓ It can run on the client-side as well as the server-side.

Role of Java Script :-

- ✓ JavaScript is a programming language that plays a key role in web development by adding interactivity and dynamic features to web pages.

- ✓ Creating interactive content, Enabling dynamic content, Improving browser compatibility, Supporting external applications, Handling user events, Connecting to mobile APIs.

2) **How is JavaScript different from other programming languages like Python orJava?**

Ans.

- ✓ JavaScript differs from other programming languages in several ways. It has a C-style syntax, supports dynamic typing, and follows event-driven, asynchronous programming.

- ✓ Unlike compiled languages, JavaScript is interpreted directly by web browsers, allowing for quick development without the need for compilation steps.

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

Ans.

- ✓ To write js code in html file U need to write all the code in <script> tag .
- ✓ <script> Discribe the Java script code..

Link External js file:-

- ✓ To link external Js file In HTML file u need ti write **<script src="(name of js file)"></script>** in HTML file.

Variables and Data Types

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

Ans.

- ✓ In JavaScript, variables are containers that store values, such as numbers or strings. You can use variables to store information that you might need to reference multiple times.

Three types Of variables.

- ✓ **Let :-** It not allowes you to redeclerection But allows you to reassign of same variable
- ✓ **Var :-** it alloves you to redeclerection and reassign of same variable
- ✓ **Const :-** it not allowes you to redeclerection and reassign of same variable

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

Ans.

Primitive Data Types

These are immutable and represent single values.

1. **Number**
   - Represents both integer and floating-point numbers.
   - Examples:

```
let age = 25;    // Integer
let price = 99.99;  // Floating-point
let infinity = Infinity;  // Special numeric value
let notANumber = NaN;  // Result of invalid math operations
```

2. **String**
   - Represents text, enclosed in single, double, or backticks.
   - Examples:

```
let name = "Alice";
let greeting = 'Hello, World!';
let template = `My name is ${name}`;
```

3. **Boolean**
   - Represents a logical entity: true or false.
   - Examples:

```
let isOnline = true;
let hasAccess = false;
```

4. **Undefined**
   - A variable declared but not assigned a value has the undefined type.
   - Example:

```
let x;
console.log(x);  // undefined
```

5. **Null**
   - Represents the intentional absence of any value.

- Example:

```
let y = null;
```

6. **Symbol** (introduced in ES6)
   - Represents a unique, immutable identifier.
   - Example:

```
let sym1 = Symbol("unique");
let sym2 = Symbol("unique");
console.log(sym1 === sym2);  // false
```

7. **BigInt** (introduced in ES2020)
   - Used to represent integers beyond the safe range for Number.
   - Example:

```
let bigNumber = 1234567890123456789012345678901234567890n;
```

Non-Primitive (Complex) Data Types

These are mutable and can store collections or more complex entities.

1. **Object**
   - Used to store key-value pairs.
   - Example:

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

2. **Array** (a type of object)
   - Used to store an ordered list of values.
   - Example:

```
let numbers = [1, 2, 3, 4];
```

3. **Function** (a type of object)

- Functions are first-class objects in JavaScript.
- Example:

```
function greet() {
  return "Hello!";
}
```

4. **Date** (a type of object)
   - Used to represent and manipulate dates and times.
   - Example:

```
let today = new Date();
```

5. **Map and Set** (introduced in ES6)
   - **Map**: Stores key-value pairs where keys can be of any type.
   - Example:

```
let map = new Map();
map.set("name", "Alice");
```

   - **Set**: Stores unique values of any type.
   - Example:

```
let set = new Set([1, 2, 3, 3]);  // {1, 2, 3}
```

JavaScript Operators

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

- **Arithmetic operators**
- **Assignment operators**
- **Comparison operators**
- **Logical operators**

Ans..

1. Arithmetic Operators

Used for mathematical calculations.

Operator	Description	Example	Output
+	Addition	5 + 2	7
-	Subtraction	5 - 2	3
*	Multiplication	5 * 2	10
/	Division	5 / 2	2.5
%	Modulus (Remainder)	5 % 2	1
**	Exponentiation	5 ** 2	25
++	Increment	let x = 5; x++	6
--	Decrement	let x = 5; x--	4

---

2. Assignment Operators

Used to assign values to variables.

Operator	Description	Example	Output
=	Assignment	x = 5	5
+=	Add and assign	x += 5	x = x + 5
-=	Subtract and assign	x -= 5	x = x - 5
*=	Multiply and assign	x *= 5	x = x * 5
/=	Divide and assign	x /= 5	x = x / 5

Operator	Description	Example	Output
%=	Modulus and assign	x %= 5	x = x % 5

---

3. Comparison Operators

Used to compare values, returning a Boolean (true or false).

Operator	Description	Example	Output
==	Equal to (loose comparison)	5 == "5"	true
===	Equal to (strict comparison)	5 === "5"	false
!=	Not equal to (loose comparison)	5 != "5"	false
!==	Not equal to (strict comparison)	5 !== "5"	true
>	Greater than	5 > 3	true
<	Less than	5 < 3	false
>=	Greater than or equal to	5 >= 5	true
<=	Less than or equal to	5 <= 3	false

---

4. Logical Operators

Used for logical operations, returning true or false.

Operator	Description	Example	Output
&&	Logical AND	true && false	false

Operator	Description	Example	Output
||		||	Logical OR
!	Logical NOT	!true	false

Question 2: What is the difference between == and === in JavaScript?

--> In JS (==) this operator only check the value of two variables are same or not..

Ex:- a=35 \\numeric

a="35" \\String

//It returns true because it's values are same ...

--> In JS (===) this operator check value and datatype of values

Ex:- a=35\\numeric

a="35" \\String

//It returns false because it's data type is different...

Control Flow (If-Else, Switch)

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

Ans.

- Control flow refers to the order in which a program's statements are executed.
- In JavaScript, like most programming languages, the default control flow is top-to-bottom: the code runs line by line, starting from the first statement and moving to the last.

✓ **If-else Statement: -**

The **if-else statement** is a conditional construct that allows you to execute different blocks of code based on whether a specified condition evaluates to true or false.

➢ Example:-

```javascript
const number = -5;

if (number >= 0) {
    console.log("The number is positive.");
} else {
    console.log("The number is negative.");
}
```

✓ **How Switch Statements Work in JavaScript:-**

A **switch statement** in JavaScript is a control flow structure that allows you to execute one block of code among multiple options based on the value of an expression. It is an alternative to using multiple if-else statements, especially when comparing the same variable or expression to multiple values.

✓ You should use a **switch statement** instead of if-else when:

a. Multiple Specific Values
b. Code Readability
c. Equality Checks

Loops (For, While, Do-While)

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

Ans..

1. for Loop:-

The for loop is the most commonly used loop in JavaScript. It allows you to run a block of code a specific number of times.

javascript Copy code

```javascript
// Example of a for loop
for (let i = 0; i < 5; i++) {
    console.log("Iteration number: " + i);
}
```

2. while Loop:-

The while loop continues to execute as long as the given condition evaluates to true. The loop checks the condition before each iteration, so if the condition is false initially, the loop will not execute.

javascript Copy code

```javascript
// Example of a while loop
let i = 0;
while (i < 5) {
    console.log("Iteration number: " + i);
    i++;
}
```

3. do-while Loop:-

The do-while loop is similar to the while loop, but with one key difference: the condition is checked after the block of code has been executed. This guarantees that the code will run at least once, even if the condition is false initially.

javascript Copy code

```javascript
// Example of a do-while loop
let i = 0;
do {
    console.log("Iteration number: " + i);
    i++;
} while (i < 5);
```

Question 2: What is the difference between a while loop and a do-while loop?

Aspect	`while` Loop	`do-while` Loop
Condition Check	The condition is checked **before** the loop executes.	The condition is checked **after** the loop executes.
Execution Guarantee	If the condition is `false` initially, the loop does not run at all.	The loop will execute **at least once**, even if the condition is `false` initially.
Use Case	Used when the loop may not need to run if the condition is false initially.	Used when the loop must execute at least once regardless of the condition.
Syntax	`javascript while (condition) {  // code }`	`javascript do {  // code } while (condition);`

Example 1: `while` Loop

javascript Copy code

```javascript
let i = 5;
while (i < 5) {
    console.log("This will not print because the condition is false.");
}
```

Example 2: `do-while` Loop

javascript Copy code

```javascript
let i = 5;
do {
    console.log("This will print once, even though the condition is false.");
} while (i < 5);
```

Functions

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

Ans..

A function in JavaScript is a block of reusable code that performs a specific task. Functions make code more modular, maintainable, and efficient by avoiding repetition.

Syntax for Declaring a Function

Functions can be declared in multiple ways in JavaScript. The most common way is using the `function` keyword.

Syntax:

javascript Copy code

```javascript
function functionName(parameters) {
    // Code to be executed
    return value; // (Optional) Returns a value
}
```

- `functionName`: The name of the function.
- `parameters`: Input values (optional) provided to the function. You can pass multiple parameters separated by commas.
- `return`: (Optional) Used to return a value from the function.

Calling a Function

Once a function is defined, you can call or invoke it using its name followed by parentheses.

Syntax:

javascript Copy code

```javascript
functionName(arguments);
```

- `arguments`: Values passed to the function corresponding to its parameters.

Question 2: What is the difference between a function declaration and a function expression?

Ans..

Comparison Table

Aspect	Function Declaration	Function Expression
Definition	Starts with the `function` keyword and has a name.	Defined as part of an expression, often assigned to a variable.
Hoisting	Hoisted to the top of its scope.	Not hoisted. Must be defined before it's called.
Anonymous Functions	Not anonymous; always named.	Can be anonymous or named.
Example	`function greet() { ... }`	`const greet = function() { ... };`
Common Use Cases	General-purpose functions that need to be reused.	Inline callbacks, event handlers, or dynamic assignment.

Example Comparing Both

javascript Copy code

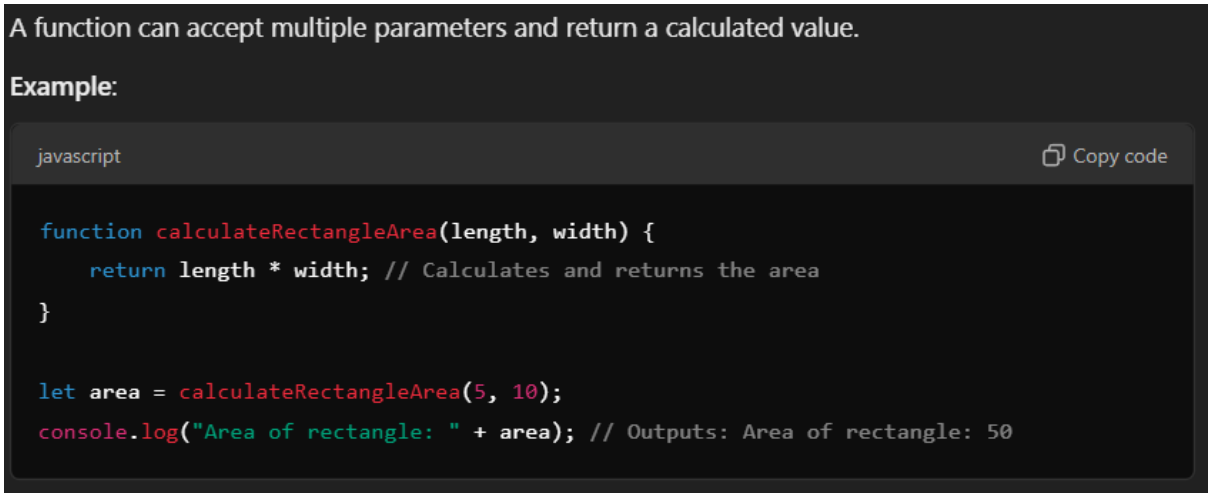
```javascript
// Function Declaration
console.log(add(5, 10)); // Works because of hoisting
function add(a, b) {
    return a + b;
}

// Function Expression
const subtract = function(a, b) {
    return a - b;
};
console.log(subtract(10, 5)); // Works only after the declaration
```

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

Ans.

1. Parameters in Functions

- Definition: Parameters are placeholders defined in a function's declaration that accept input values (called arguments) when the function is called.
- Purpose: They allow functions to be more flexible by operating on variable data.



Arrays

Question 1: What is an array in JavaScript? How do you declare and initialize an array?

Ans..

An array in JavaScript is a special variable that can hold multiple values in a single container. Arrays are used to store a collection of data, which can be of the same or different types, and are ordered by index starting from zero(0).

You can declare and initialize an array in JavaScript in multiple ways.

1. Using Square Brackets ([])

This is the most common and concise way to create an array.

Syntax:

let arrayName = [value1, value2, value3];

2. Using the Array Constructor

You can also create an array using the Array constructor, though this method is less commonly used.

Syntax:

let arrayName = new Array(value1, value2, value3);

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

Ans..

```
//1)array push function can add the element at the end of the array
document.write('<br> ');
document.write('<br> '+arr)
arr.push(65)
document.write('<br> '+arr)

//1)array pop function can remove the element at the end of the array
document.write('<br> ');
arr.pop()
document.write('<br> '+arr)

//3)unshift function can add the element at the starting of the array
document.write('<br> ');
arr.unshift(111)
document.write('<br> '+arr)

//4)shift function can remove the element at the starting of the array
document.write('<br> ');
arr.shift()
document.write('<br> '+arr)
```

Objects

Question 1: What is an object in JavaScript? How are objects different from arrays?

Ans..

An object in JavaScript is a collection of key-value pairs where:

- Keys are strings (or Symbols) used as property names.
- Values can be of any data type, including numbers, strings, arrays, functions, or other objects.

Objects are used to model real-world entities and represent structured data.

Syntax for Declaring an Object

Objects are created using curly braces `{}`.

Syntax:

javascript Copy code

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

Example:

javascript Copy code

```javascript
let person = {
    name: "Alice",
    age: 25,
    isStudent: true
};
console.log(person.name); // Outputs: Alice
console.log(person["age"]); // Outputs: 25
```

How Objects are Different from Arrays

Aspect	Objects	Arrays
Structure	Stores data as key-value pairs.	Stores data as an ordered list of values.
Access	Accessed using keys (e.g., `obj.key` or `obj["key"]`).	Accessed using numeric indices (e.g., `arr[0]`).
Order	Keys are unordered (in most cases).	Indices are ordered (0, 1, 2, ...).
Purpose	Best for modeling complex data with named properties.	Best for ordered collections or lists of items.
Data Representation	Can include methods (functions).	Typically holds similar data types.

Question 2: Explain how to access and update object properties using dot notation and bracket notation.

Ans..

1. Accessing Properties

Dot Notation

- The property name is specified directly after a dot (`.`).
- It's simpler and more readable but requires that the property name is a valid identifier (e.g., no spaces or special characters).

Example:

javascript Copy code

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

Bracket Notation

- The property name is specified as a string inside square brackets (`[]`).
- It allows dynamic access and is necessary when the property name contains special characters, spaces, or starts with a number.

Example:

javascript Copy code

```javascript
let person = { "first name": "Alice", age: 25, "1stPlace": "Gold" };
console.log(person["first name"]); // Outputs: Alice
console.log(person["1stPlace"]);   // Outputs: Gold

// Dynamic property access
let key = "age";
console.log(person[key]); // Outputs: 25
```

2. Updating Properties

You can use either notation to update the value of an object's property or add new properties.

Using Dot Notation

Example:

javascript Copy code

```javascript
let car = { brand: "Toyota", model: "Corolla" };
car.model = "Camry"; // Updates an existing property
car.year = 2020;     // Adds a new property
console.log(car);    // Outputs: { brand: "Toyota", model: "Camry", year: 2020 }
```

Using Bracket Notation

Example:

javascript Copy code

```javascript
let car = { brand: "Toyota", model: "Corolla" };
car["model"] = "Camry";   // Updates an existing property
car["color"] = "Blue";    // Adds a new property
console.log(car);         // Outputs: { brand: "Toyota", model: "Camry", color: "Blue" }
```

DOM Manipulation

Question 1: What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

Ans..

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a web page as a tree of objects, where each part of the document (like elements, attributes, and text) is a node.

- The DOM allows developers to dynamically access, manipulate, and update the content, structure, and styles of a web page using JavaScript.

- It acts as a bridge between JavaScript and the HTML/CSS of a webpage.

How JavaScript Interacts with the DOM

- JavaScript interacts with the DOM using the **DOM API** (Application Programming Interface), which provides methods and properties to manipulate the web page.

1. Accessing DOM Elements

JavaScript can locate elements using methods like:

- **document.getElementById()**: Finds an element by its id.
- **document.querySelector()**: Finds the first element matching a CSS selector.
- **document.getElementsByClassName()**: Finds elements by class name.

```javascript
let heading = document.getElementById("main-heading");
console.log(heading.textContent); // Outputs the text inside the element
```

2. Modifying DOM Elements

- JavaScript can modify the content, attributes, and styles of DOM elements.

```javascript
let heading = document.getElementById("main-heading");
heading.textContent = "Welcome to JavaScript!"; // Changes the text
heading.style.color = "blue";                   // Changes the text color
```

3. Adding and Removing Elements

- Use **document.createElement()** to create new elements.
- Use **parentElement.appendChild()** to add elements.
- Use **element.remove()** to delete elements.

```javascript
// Adding a new paragraph
let newPara = document.createElement("p");
newPara.textContent = "This is a dynamically added paragraph.";
document.body.appendChild(newPara);

// Removing an element
let heading = document.getElementById("main-heading");
heading.remove();
```

Question 2: Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM.

Ans..

- **getElementById() use to get element by Tag's Id.**
- **getElementByClassName() Use to get element by it's class Name.**
- **querySelector() is only select first tag from body of HTML.**

Method	Returns	Selects By	When to Use
`getElementById()`	A single element (`null` if none).	`id` attribute	When you know the `id` of the element.
`getElementsByClassName()`	A live HTMLCollection (array-like).	`class` name	To select multiple elements with the same class.
`querySelector()`	The first matching element (`null` if none).	CSS selector	For flexibility with complex CSS selectors.