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

Ans

JavaScript (JS) is a high-level, interpreted **programming language** that makes web pages **interactive** and dynamic. It is one of the three core technologies of the World Wide Web, along with HTML and CSS.

1.. Client-Side (Frontend

2.Server-Side (Backend)

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

Ans.

JavaScript: Runs mainly **inside web browsers** (like Chrome, Firefox) to make web pages interactive.

- It can also run on servers using **Node.js**.

Python: Runs on the **server-side** or locally — great for automation, AI, and data analysis.

Java: Runs on the **Java Virtual Machine (JVM)** — used for large enterprise systems and Android apps.

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 **insert JavaScript code** — either directly inside the HTML file or by linking to an external JavaScript file.

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

Ans

A **variable** in JavaScript is like a **container** that stores data — such as numbers, text, or other values.

```
let name = "Ketan";
let age = 22;
var city = "Ahmedabad";
const country = "India";
```

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

Ans

Data types define the kind of value a variable can hold — like a number, text, true/false, object, etc.

Primitive Data Types

Non-Primitive (Reference) Data Types

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

Ans.

`undefined` = not assigned yet

`null` = assigned as empty intentionally

7. What are the different types of operators in JavaScript? Explain with examples. • Arithmetic operators • Assignment operators • Comparison operators • Logical operators

Ans

1. . Arithmetic Operators

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

console.log(a + b); // 15  (Addition)
console.log(a - b); // 5   (Subtraction)
console.log(a * b); // 50  (Multiplication)
console.log(a / b); // 2   (Division)
```

```
console.log(a % b); // 0   (Remainder)
```

2.Assignment Operators

```
 let x = 10;   // assign
x += 5;       // x = x + 5 → 15
x -= 3;       // x = x - 3 → 12
```

3. Comparison Operators

```
console.log(5 == "5");  // true   (equal value)
console.log(5 === "5"); // false  (equal value and type)
console.log(10 > 5);    // true
console.log(10 <= 5);   // false
```

4.Logical operators

```
let age = 20;

console.log(age > 18 && age < 30); // true (AND)
console.log(age > 25 || age === 20); // true (OR)
console.log(!(age > 25)); // true (NOT)
```

8.What is the difference between == and === in JavaScript?

Ans

`==` → checks value only

`===` → checks value **and** type

9. What is control flow in JavaScript? Explain how if-else statements work withan example.

Ans

Control flow means **the order in which statements or code are executed** in a program.

Normally, JavaScript runs code **from top to bottom**, but using **control flow statements** (like `if`, `else`, `for`, `while`, etc.), we can **change the flow** — for example, run some code only if a condition is true.

🔹 **if-else Statement**

The **`if-else`** statement is used to **make decisions** in a program.

- **`if`** → runs code if a condition is true.
- **`else`** → runs code if that condition is false.

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

A **`switch` statement** is used to **test one variable or expression against many possible values**.
It's a cleaner alternative to writing many `if...else if` conditions.

🔹 **How it works:**

- The value inside `switch()` is compared with each `case`.
- When a **match** is found, that block of code runs.
- The `break` statement stops the switch after a match.

- The `default` block runs if **no case matches** (like an `else`)

11. Explain the different types of loops in JavaScript (for, while, do-while). Provide abasic example of each.
Ans
A **loop** is used to **repeat a block of code** as long as a condition is true.
It helps avoid writing the same code again and again.
1.for-loop

```
for (let i = 1; i <= 5; i++) {
  console.log("Number:", i);
}
```

2.While loop

```
let i = 1;

while (i <= 5) {
  console.log("Count:", i);
  i++;
}
```

3.do...while loop

```
let i = 1;

do {
  console.log("Value:", i);
  i++;
} while (i <= 5);
```

12.: What is the difference between a while loop and a do-while loop?
Ans
Checks the **condition first** before running the code.

- If the condition is **false at the start**, the code **won't run** even once.

do...while loop

- Runs the code **at least once**, then checks the condition.
- Even if the condition is false, the loop runs **one time**.

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

Ans

A **function** is a **block of code** designed to **perform a specific task**.
It helps you **reuse code** instead of writing it again and again.

u **declare** it with `function name() { }` and **call** it using `name()`.

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

Ans

A **function declaration** is when you define a function using the `function` keyword **with a name**.

Function Expression

A **function expression** means you **store a function inside a variable**.

15. Discuss the concept of parameters and return values in functions.

Ans

What are Parameters?

- **Parameters** are like **placeholders** or **inputs** for a function.
- They let you pass data **into** a function.

What is a Return Value?

- A function can **send a result back** using the `return` keyword.
- `return` stops the function and gives back a value.

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

Ans

An **array** in JavaScript is a **collection of values** stored in a single variable.

It can hold **many items** like numbers, strings, or even objects.

17. Explain the methods push(), pop(), shift(), and unshift() used in arrays.

Ans

push()

- Adds a new element **to the end** of the array

. pop()

- Removes the **last element** from the array.

shift()

- Removes the **first element** from the array.

unshift()

- Adds a new element **to the beginning** of the array.

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

Ans

An **object** is a collection of **key–value pairs**.

Arrays are collections of **ordered elements** stored by **index numbers**

19. Explain how to access and update object properties using dot notation and bracket notation.

Ans

Accessing and Updating Object Properties

In JavaScript, you can access or change (update) object values in **two ways**:
👉 **dot notation (.)** and **bracket notation ([ ])**.

🔸 **1. Dot Notation**

- You use a **dot (.)** followed by the property name.

- It’s the most common and simple way.

🧩 Example (Theory + Explanation):

- If an object has a property like `name`, you write it as `object.name`.

- You can also update it directly using the same way.

20. What are JavaScript events? Explain the role of event listeners.

Ans

In JavaScript, an **event** is an action or activity that happens in the browser —

something that the **user** or the **browser** does.

What are Event Listeners?

An **event listener** is a way to **make JavaScript react** when an event happens.
It **waits (listens)** for an event — like a click — and then runs some code.

21.How does the addEventListener() method work in JavaScript? Provide an example.

Ans

What is `addEventListener()`?

The `addEventListener()` method is used to **attach an event** (like a click or keypress) to an HTML element.
It tells JavaScript:

> "When this event happens on this element, run this function."

22. What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

Ans

What is the DOM (Document Object Model)?

The **DOM** stands for **Document Object Model**.
It is a **tree-like structure** that represents your entire webpage — all its elements (like `<h1>`, `<p>`, `<div>`, etc.) — in a way that JavaScript can **understand and manipulate**.

🧠 **In simple words:**

> The DOM is how JavaScript **sees and controls** your HTML page.

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

Ans

getElementById()

- Used to **select one element** by its **ID**.

getElementsByClassName()

- Used to select **all elements** with the same **class name**.
- Returns a **collection (array-like list)** of elements.

querySelector()

- Selects the **first element** that matches a **CSS selector**.
- You can use **ID (`#id`)**, **class (`.class`)**, or **tag (`tagname`)**.

24. Explain the setTimeout() and setInterval() functions in JavaScript. Howare they used for timing events?

Ans

setTimeout()

- The `setTimeout()` function is used to **run a piece of code once**, **after a certain time delay** (in milliseconds).

setInterval()

- The `setInterval()` function is used to **run a piece of code repeatedly** after a fixed time interval.
- It keeps running the code **again and again** until you stop it.

25.Provide an example of how to use setTimeout() to delay an action by 2 seconds.

Ans

setTimeout(function() {

```
  console.log("This message appears after 2 seconds!");
}, 2000);  // 2000 milliseconds = 2 seconds
```

26.What is error handling in JavaScript? Explain the try, catch, and finally blocks with an example.

Ans

Error handling means managing mistakes in your code so the program **doesn't crash**.

JavaScript uses the `try...catch...finally` blocks to handle such errors safely.

27. Why is error handling important in JavaScript applications?

Ans

✅ **Main Reasons:**

1. **Prevents program crashes**
   → Without error handling, one small mistake can stop the entire program.

2. **Makes debugging easier**
   → You can show clear messages about what went wrong.

3. **Improves user experience**
   → Users see friendly error messages instead of a broken webpage.

4. **Helps handle unexpected situations**
   → For example, missing data from a server or wrong user

input.

5. **Keeps code more reliable and secure**
   → Errors are caught and managed safely without exposing internal issues.