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

Ans:

JavaScript is a high-level, interpreted programming language used to make web pages interactive. It runs directly in the browser and it is client-side scripting language.

Role of JavaScript in web development:

1. Enhances User Interaction (Client-Side Scripting)
    - Makes static HTML pages interactive.
    - Examples:
        - Show/hide elements
        - Validate forms before submission
        - Create image sliders or tabs
2. Manipulates Web Page Content (DOM Manipulation)
    - Can access and modify HTML and CSS using the Document Object Model (DOM).
    - Example:
      Changing the text of a heading when a button is clicked.
3. Enables Dynamic Content Loading (AJAX)
    - Can communicate with servers in the background using AJAX (Asynchronous JavaScript and XML).
    - Example:
        - Load new data (e.g., products, comments) without refreshing the entire page.
4. Event Handling
    - JavaScript can respond to user actions like clicks, keypresses, scrolls, and mouse movements.
    - Example:
      Trigger an animation when the user scrolls down.

Question: How is JavaScript different from other programming languages like Python or Java?

Ans:

JavaScript is client-side scripting language mainly run in the browser, directly used for web development where python and java area backend language runs on server.

JavaScript syntax style is like C language uses of curly brackets and semi colons where in python syntax is very clean and readable.

JavaScript and python both are dynamically typed where java is statically typed.in JavaScript and python variable no need to define where in java must define.

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

Ans:

The <script> tag in HTML is used to embed JavaScript code in an HTML document. JavaScript is used to make web pages interactive, and the <script> tag tells the browser to execute the JavaScript code.

Link an external JavaScript file:

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

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

Ans:

Variables are used to store data values.

There are 3 way to declare variable:

1. Var:
    - Can redeclare and reassign.
    - Global scope.
2. Let:
    - Cannot be redeclared in the same scope.
    - Can be reassigned**.**
    - Block-scoped
3. Const:
    - Cannot be redeclared or reassigned
    - Block-scoped

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

Ans:

Data type tells us which type of data variable stores, JavaScript is dynamically typed language it means it decides the type of variable during run time.

1) Primitive data type:

- Number

  Represents integers and floating-point numbers.

  let age = 25;

  let price = 99.99;

- String

  Represents text enclosed in " ", ' ', or backticks ` `.

  let name = "Raj";

  let a = 'Hello';

  let msg = `Hiiii`;

- Boolean

  Represents logical values: true or false.

  let status = true;

  let statusFal = false;

- Null

  Represents an intentional “empty” value.

  let x = null;

- Undefined

  A variable declared but not assigned a value is undefined.

  let x;

2) Non-primitive data type:

- Array

  An ordered list of values , indexed starting at 0.

  let fruits = ["apple", "banana", "cherry"];

- Object

  Collection of key–value pairs.

  let car = {

  brand: "Mahindra",

  model: "Scorpio",

  year: 2020

  };

Question: What is the difference between undefined and null in JavaScript?

Ans:

Undefined:

A variable has been declared but not assigned any value. JavaScript itself sets variables to undefined if no value is assigned.

Null:

An intentional absence of any value. We generally assign null to indicate no value or empty.

Question:

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 to perform mathematical operations.

Operator	Meaning	Example	Output
+	Addition	5 + 2	7
-	Subtraction	5 - 2	3
*	Multiplication	5 * 2	10
/	Division	10 / 2	5
%	Modulus	10 % 3	1
**	Exponentiation	2 ** 3	8
++	Increment by 1	let x=5; x++;	6
--	Decrement by 1	let x=5; x--;	4

2. Assignment Operators

Used to assign values to variables.

Operator	Meaning	Example	Equivalent
=	Assign value	x = 5	—
+=	Add and assign	x += 3	x = x + 3
-=	Subtract and assign	x -= 2	x = x - 2
*=	Multiply and assign	x *= 4	x = x * 4
/=	Divide and assign	x /= 2	x = x / 2
%=	Modulus and assign	x %= 2	x = x % 2

3. Comparison Operators

Used to compare two values. They return a boolean (true or false).

Operator	Meaning	Example	Output
==	Equal (values only)	5 == "5"	true
===	Strict equal (value + type)	5 ==="5"	false
!=	Not equal (values only)	5 != "5"	false
!==	Strict not equal (value+type)	5 !== "5"	true
>	Greater than	7 > 3	true
<	Less than	3 < 7	true
>=	Greater than or equal	5 >= 5	true
<=	Less than or equal	3 <= 2	false

4. Logical Operators

Used to combine conditions.

Operator	Meaning	Example	Output
&&	AND → true if both true	(5 > 2 && 10 > 5)	true
||	OR → true if one is true	(5 > 2 || 10 < 5)	true
!	NOT → reverses boolean value	!(5 > 2)	false

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

Ans:

==:

Compares only values, not types. If the types are different, JavaScript automatically converts (coerces) one type to match the other before comparison.

===:

Compares both value and type. No type conversion happens. Values must be exactly the same and of the same type.

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

Ans:

Control flow is the order in which statements are executed in a program. By default, JavaScript runs code from top to bottom, line by line. But with control flow statements like if-else, loops, switch, we can change that order depending on conditions.

The if-else statement allows you to execute certain code only if a condition is true, and a different block if it is false.

```
let age = 18;

if (age >= 18) {

   console.log("You are eligible to vote.");

}

else {

   console.log("You are not eligible to vote.");

}
```

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

Ans:

A switch statement is used when you want to execute different blocks of code based on the value of a single expression. It is an alternative to writing multiple if-else if statements. It makes code cleaner and easier to read.

```
let userInput = parseInt(prompt("Enter the number (1-7) : "));

    let dayname;

    switch(userInput){

        case 1:

            dayname = "Monday";
```

```
        break;
    case 2:
        dayname = "Tuesday";
        break;
    case 3:
        dayname = "Wedenesday";
        break;
    case 4:
        dayname = "Thursday";
        break;
    case 5:
        dayname = "Friday";
        break;
    case 6:
        dayname = "Saturday";
        break;
    case 7:
        dayname = "Sunday";
        break;
    default:
        dayname="Invalid input!!!";
}
alert("Day : " + dayname);
```

Question: 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 in advance how many times you want to repeat the code.

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

2. while loop

Used when you don't know in advance how many times you need to loop. The loop continues as long as the condition is true.

```
let i = 1;
while (i <= 5) {
  console.log(i);
  i++;
}
```

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

Ans:

1. while loop

   Condition is checked before executing the loop body.

   If the condition is false initially, the loop body never runs.

2. do-while loop

   Condition is checked after executing the loop body.

   The loop body always runs at least once, even if the condition is false.

Question: 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. Functions help you reuse code, make your program more organized, and avoid repetition.

Function declaration:

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

Function calling:

```
function greet(name) {
    console.log("Hello, " + name + "!");
}
greet(name);
```

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

Ans:

Function declaration:

We can call it before and after declaration. Can not be anonymous. Declare using function keyword.

```
function name () {.....}
```

Function expression:

A function is assigned to a variable. Can not be called before it is defined. Can be anonymous.

```
const name = function () {...}
```

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

Ans:

1. Parameters in Functions

Parameters are placeholders used in function definitions to accept input values when the function is called.

2. Return Values

Functions can return a value using the return keyword.

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

Ans:

An array is a special type of object that stores multiple values in a single variable. The values in an array are ordered and indexed starting from 0. Arrays can store numbers, strings, booleans, objects, or even other arrays.

```
let fruits = ["apple", "banana", "cherry"];

console.log(fruits);

console.log(fruits[0]);
```

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

Ans:

1. push()

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

```
let fruits = ["apple", "banana"];

fruits.push("cherry");

console.log(fruits); // ["apple", "banana", "cherry"]
```

2. pop()

Removes the last element from an array. Returns the removed element.

```
let fruits = ["apple", "banana", "cherry"];

let removed = fruits.pop();

console.log(fruits);   // ["apple", "banana"]

console.log(removed);  // "cherry"
```

3. shift()

Removes the first element from an array. Returns the removed element.

```
let fruits = ["apple", "banana", "cherry"];

let first = fruits.shift();

console.log(fruits);  // ["banana", "cherry"]

console.log(first);   // "apple"
```

4. unshift()

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

```
let fruits = ["banana", "cherry"];

fruits.unshift("apple");

console.log(fruits); // ["apple", "banana", "cherry"]
```

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

Ans:

An object is a collection of key-value pairs (also called properties). Objects are used to store and organize data in a structured way. Keys are always strings (or symbols), and values can be any data type.

```
let car = {
        brand: "Toyota",
        model: "Corolla",
        year: 2020,
        colors: ["red", "blue"]
};
```

Feature	Object	Array
Structure	Key-value pairs	Ordered list of values
Access	object.key or object["key"]	array[index]
Indexing	Keys	Numeric index (0,1,2...)
Example	{name:"John", age:30}	["apple","banana","cherry"]

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

Ans:

Dot Notation

Use the dot (.) followed by the property name.

Works when the property name is a valid identifier no spaces, special characters, doesn't start with a number.

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

Bracket Notation

Use square brackets [] and pass the property name as a string.

Useful when the property name has spaces, special characters, or is dynamic.

```
let person = {
 "full name": "Alice Johnson"
};
console.log(person["full name"]);
```

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

Ans:

An event is an action or occurrence that happens in the browser, often triggered by the user or the browser itself. Events allow JavaScript to react dynamically to user interactions.

Examples of events:

User actions: click, mouseover, keydown, submit

Browser actions: load, resize, scroll

An event listener is a JavaScript function that waits for an event to occur on a specific element

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

Ans:

addEventListener() is a JavaScript method that attaches an event handler to an HTML element. It allows your code to react to events like clicks, mouse movements, key presses.

```
<button id="myBtn">Click me</button>
  <script>
    let btn = document.getElementById("myBtn");
    btn.addEventListener("click",function() {alert("Button clicked!");});
  </script>
```

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

Ans:

DOM (Document Object Model) is a programming interface for web documents.

It represents the HTML or XML document as a tree of objects.

The DOM allows JavaScript to access, modify, and manipulate the web page dynamically.

JavaScript can read, modify, and create DOM elements using DOM methods.

1) Selecting elements:

   ```
   let heading = document.getElementById("title");
   let paragraph = document.querySelector("p");
   ```

2) Changing content:

   ```
   heading.textContent = "Welcome to JavaScript!";
   paragraph.innerHTML = "This paragraph has <b>bold</b> text now.";
   ```

3) Changing style:

   ```
   heading.style.color = "blue";
   paragraph.style.fontSize = "18px";
   ```

4) Creating elements:

```
let newDiv = document.createElement("div");

newDiv.textContent = "I am a new div!";

document.body.appendChild(newDiv);
```

5) Handling events:

```
heading.addEventListener("click", function() {

  alert("You clicked !");

});
```

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

Ans:

1. getElementById()

   Selects a single element by its id.

   Returns the first element with the given id.

   ```
   let element = document.getElementById("elementId");
   ```

2. getElementsByClassName()

   Selects all elements with a given class name.

   ```
   let elements = document.getElementsByClassName("myClass");
   ```

3. querySelector()

   Selects the first element that matches a CSS selector.

   Very flexible, works with id, class, tags, or any CSS selector.

   ```
   let element = document.querySelector("selector");
   ```

Question: Explain the setTimeout() and setInterval() functions in JavaScript. How are they used for timing events?

Ans:

1. setTimeout()

   Executes a function once after a specified delay in milliseconds.

```
setTimeout(function() {
  console.log("This message appears after 3 sec");
}, 3000);
```

2. setInterval()

   Executes a function repeatedly at specified intervals in milliseconds.

```
setInterval(function() {
  console.log("This message appears every 2 sec");
}, 2000);
```

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

Ans:

```
<script>
    setTimeout(() => {alert("This appears after 2 seconds!");}, 2000);
</script>
```

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

Ans:

Error handling is the process of detecting and responding to runtime errors in a program. It prevents the script from crashing and allows the program to handle errors gracefully.

1. try

   Contains the code that might throw an error.

   If an error occurs, execution jumps to the catch block.

2. catch

   Executes only if an error occurs in the try block.

   Receives the error object that contains information about the error.

3. finally

   Executes regardless of whether an error occurred or not.

```
try {
  let result = 10 / 0;
  console.log("Result:", result);
throw new Error("Something went wrong!");
}
catch (error) {
  console.log("Error caught:", error.message);
}
finally {
  console.log("This always runs");
}
```

Question: Why is error handling important in JavaScript applications?

Ans:

Without error handling, a runtime error can stop the entire script from executing.

Using try-catch, you can catch errors and allow the application to continue running smoothly.

By catching and logging errors, developers can identify bugs and fix them faster.

Makes large applications more maintainable

Improves User Experience, users won’t see broken pages or crashes.