1. What is JavaScript? How is it different from Java?

JavaScript is a high-level, interpreted scripting language primarily used to create interactive effects within web browsers.

Unlike Java, which is a compiled, object-oriented programming language, JavaScript is dynamic and interpreted at

runtime.

2. What are the data types supported in JavaScript?

Primitive: Number, String, Boolean, Null, Undefined, Symbol, BigInt

Non-primitive: Object, Array, Function

3. What are var, let, and const? How do they differ?

- var: Function-scoped, can be re-declared and updated.

- let: Block-scoped, cannot be re-declared in the same scope.

- const: Block-scoped, cannot be re-declared or reassigned.

4. What is hoisting in JavaScript?

Hoisting is JavaScript's default behavior of moving declarations to the top of the current scope during compilation.

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution. Only declarations are hoisted, not initializations.

* var x is hoisted to the top, so it becomes:

var x;

console.log(x); // undefined

x = 5;

**🔹 Example 2: Let/Const are Not Hoisted (or are in TDZ)**

javascript

CopyEdit

console.log(y); // ReferenceError: Cannot access 'y' before initialization

let y = 10;

* let and const are hoisted but stay in the **Temporal Dead Zone (TDZ)** until their line of declaration.

**Example 3: Function Hoisting**

javascript

CopyEdit

sayHello(); // Output: Hello!

function sayHello() {

console.log("Hello!");

}

**Explanation:**

* Entire function sayHello() is hoisted to the top.

**Example 4: Function Expression is NOT Hoisted**

javascript

CopyEdit

greet(); // TypeError: greet is not a function

var greet = function () {

console.log("Hi!");

};

**Explanation:**

* var greet is hoisted (initialized as undefined), but the function is not yet assigned.

5. What is the difference between == and ===?

- == compares values after type coercion.

- === compares values and types strictly (no coercion).

6. Explain null vs undefined.

- null: Assigned value representing no value.

- undefined: Variable declared but not assigned.

7. What are truthy and falsy values?

In JavaScript, truthy and falsy values refer to how non-boolean values behave in a boolean context (like an if statement).

Falsy: false, 0, '', null, undefined, NaN

| **Falsy Value** | **Description** |
| --- | --- |
| false | The boolean false |
| 0 | The number zero |
| -0 | Negative zero |
| 0n | BigInt zero |
| "" | Empty string |
| null | Null value |
| undefined | Undefined |
| NaN | Not a Number |

Truthy: All other values.

if ("hello") // true

if (42) // true

if ([]) // true

if ({}) // true

if ("0") // true

if (new Date())// true

8. How does type coercion work in JavaScript?

JavaScript automatically converts data from one type to another when required during comparison or arithmetic operations.

9. What are template literals?

Strings enclosed in backticks (``) that allow embedded expressions using ${} , multi-line string , call function in string , Expression Evaluation.

10. What is the difference between typeof and instanceof?

- typeof returns the data type of a variable.

typeof 123 // 'number'

typeof "hello" // 'string'

typeof true // 'boolean'

typeof undefined // 'undefined'

typeof null // 'object' ❗ (a JavaScript quirk)

typeof {} // 'object'

typeof [] // 'object'

typeof function(){} // 'function'

- instanceof checks whether an object is an instance of a specific class or constructor.

[] instanceof Array // true

{} instanceof Object // true

new Date() instanceof Date // true

"hello" instanceof String // false ❗ because "hello" is a primitive, not a String object

new String("hello") instanceof String // true

11. What is the difference between for, for...in, and for...of?

- for: Traditional loop for indexing.

- for...in: Iterates over object keys.

const person = { name: "Krish", age: 25 };

for (let key in person) {

console.log(key, person[key]);

// name Krish

// age 25

}

- for...of: Iterates over iterable values like arrays.

const arr = [10, 20, 30];

for (let value of arr) {

console.log(value); // 10 20 30

}

const str = "JS";

for (let char of str) {

console.log(char); // J S

}

12. How does the switch statement work in JavaScript?

Evaluates an expression and executes code blocks based on matching case labels.