JavaScript Questions and Answers

EASY LEVEL

1. JavaScript is used to make web pages interactive.

2. var is function-scoped; let and const are block-scoped. const cannot be reassigned.

3. A function is a reusable block of code.

4. An array stores multiple values in a single variable.

5. An object stores data in key–value pairs.

6. == compares value; === compares value + type.

7. Loops repeat code: for, while, do-while, for...of, for...in.

8. typeof tells the type of a value.

9. // single-line comment, /* */ multi-line comment.

10. A string is text inside quotes.

11. A boolean is true or false.

12. let x = 10; declares a variable.

13. return sends a value back from a function.

14. NaN means Not-a-Number.

15. null = empty value; undefined = value not assigned.

MEDIUM LEVEL

1. Hoisting moves declarations to the top of scope.

2. Scope defines where variables exist (global, block, function).

3. A closure is a function that remembers its outer scope.

4. map returns a new array; filter returns items that pass a condition; forEach loops but returns nothing.

5. A Promise represents a future completed/failed value.

6. async/await makes asynchronous code look synchronous.

7. The event loop handles asynchronous operations in JS.

8. A callback is a function passed into another function.

9. this refers to the context of execution.

10. Destructuring extracts values from arrays/objects.

11. Spread operator (...) expands elements.

12. Rest operator (...) collects remaining arguments.

13. localStorage stores persistent key-value data in browser.

14. JSON is JavaScript Object Notation used for data exchange.

15. isNaN() checks if value is NaN.

HARD LEVEL

1. Prototypes allow objects to inherit properties from other objects.

2. Bubbling: event goes child → parent; Capturing: parent → child.

3. Currying transforms a function into nested single-argument functions.

4. Debouncing delays execution; throttling limits execution frequency.

5. Shallow copy copies top level; deep copy copies all nested levels.

6. call, apply, bind set this for functions; apply uses array args.

7. Execution context includes variable environment + scope + this.

8. Garbage collection removes unused memory automatically.

9. Memoization caches function results.

10. JS handles async via event loop, callback queue, and microtask queue.