JavaScript Interview Questions & Answers (2 Yrs Experience)

1. What are the different data types in JavaScript?

JavaScript has 7 primitive data types: String, Number, Boolean, null, undefined, Symbol, and BigInt. Object is a non-primitive type.

2. What is the difference between var, let, and const?

`var` is function-scoped and hoisted. `let` and `const` are block-scoped. `const` cannot be reassigned.

3. What is hoisting in JavaScript?

Hoisting moves declarations to the top of their scope. Only declarations are hoisted, not initializations.

4. What is the difference between undefined and null?

`undefined` means a variable has been declared but not assigned. `null` is an assignment value that represents no value.

5. What is a closure in JavaScript?

A closure is a function that has access to its lexical scope even when executed outside that scope.

Example:

```
function outer() {
  let count = 0;
  return function inner() {
    count++;
    console.log(count);
  }
}
```

JavaScript Interview Questions & Answers (2 Yrs Experience)

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

`==` compares values after type coercion. `===` compares both value and type.

7. What are truthy and falsy values?

Falsy: false, 0, '', null, undefined, NaN. All others are truthy.

8. What is the use of typeof operator?

`typeof` returns a string indicating the type of the operand.

9. How does 'this' keyword work?

In global context, `this` refers to the window. Inside an object method, it refers to the object.

10. What are template literals?

Template literals use backticks and allow embedded expressions:

`Hello, ${name}`

11. What are arrow functions and their differences?

Arrow functions have a shorter syntax and no own `this`. Useful for callbacks and methods.

12. Explain destructuring in JavaScript.

Destructuring allows unpacking values:

const [a, b] = [1, 2];

const {name, age} = obj;

JavaScript Interview Questions & Answers (2 Yrs Experience)

13. What is the spread and rest operator?

`...` used to expand (`spread`) or collect (`rest`) elements:

function sum(...args) { return args.reduce((a,b)=>a+b); }

14. What are default parameters?

Function parameters can have default values:

function greet(name='Guest') { console.log(name); }

15. What are callbacks, promises, and async/await?

Callback: function passed into another.

Promise: handles async code.

Async/Await: cleaner syntax for Promises.

16. What is the event loop?

Event loop handles async operations. Executes stack, then microtasks, then macrotasks.

17. What is lexical scope?

Lexical scope means variables are accessible based on where they were declared.

18. How does map, filter, and reduce work?

`map` transforms, `filter` selects, `reduce` accumulates:

arr.reduce((acc, val) => acc + val, 0)

19. What are higher-order functions?

Functions that take or return other functions.

Example: `map`, `filter`

20. Difference between forEach and map?

`forEach` executes a function but doesn't return a new array. `map` returns a new array.

21. What is a memory leak?

Memory that is no longer used but not released. Common causes: global variables, event listeners.

22. What is debouncing and throttling?

Debounce: limit function call until stop typing.

Throttle: limit function to run at intervals.

23. What is shallow copy vs deep copy?

Shallow copy copies references. Deep copy copies nested objects. Use `structuredClone` or `JSON.parse(JSON.stringify(obj))`.

24. Design patterns in JavaScript?

Common patterns: Module, Singleton, Factory, Observer.

25. Difference between sync and async code?

Synchronous blocks execution. Asynchronous doesn't block (e.g., setTimeout, fetch).

26. Explain microtasks vs macrotasks.

Microtasks: Promises, queueMicrotask.

Macrotasks: setTimeout, setInterval.

27. How does garbage collection work?

GC automatically frees memory of unreferenced variables (Mark-and-sweep algorithm).

28. What is currying?

Transforming a function with multiple arguments into a sequence of functions.

Example: f(a)(b)(c)

29. How to make object immutable?

Use Object.freeze(obj). For deep immutability, recursively freeze nested objects.

30. Explain ES6 modules.

Use `import` and `export` to organize code. Replaces CommonJS.

31. How do you handle API errors?

Use try-catch with async/await or `.catch()` with promises.

Example: try { await fetch() } catch(e) { ... }

32. Have you written custom utilities?

Yes. Example: debounce function to optimize input search.

33. How to optimize DOM performance?

Use document fragments, limit reflows, debounce resize events.

34. Explain a JS bug you solved.

Example: Closure inside loops caused incorrect variable capture. Solved using let or IIFE.

35. How do you test JS code?

Using Jest or Mocha. Write unit tests for functions, mocks for APIs.

36. How do you manage state in vanilla JS?

By using objects or event-based state updates. Example: state object + render function.

37. How do you validate forms?

Using regex and custom validation logic. Listen to input/blur events.

38. How to handle large data rendering?

Use virtual scrolling, pagination, or lazy rendering techniques.

39. How to avoid callback hell?

Use Promises or async/await to flatten nested callbacks.

40. Have you used bundlers like Webpack/Vite?

Yes. Vite offers faster HMR and tree shaking. Used `import.meta.env` for config management.