Sure! Here's a comprehensive list of answers to commonly asked JavaScript interview questions:

---

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.

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?

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

Truthy: All other values.

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 ${}.

10. What is the difference between typeof and instanceof?

- typeof returns the data type of a variable.
- instanceof checks whether an object is an instance of a specific class or constructor.

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

- for: Traditional loop for indexing.
- for...in: Iterates over object keys.
- for...of: Iterates over iterable values like arrays.

12. How does the switch statement work in JavaScript?

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

13. Explain break and continue.

- break: Exits a loop or switch statement.
- continue: Skips current iteration and continues with the next.

14. What is a function in JavaScript?

A reusable block of code that performs a specific task.

15. What are arrow functions?

Concise syntax for writing functions using =>. They don't bind their own this.

16. Function declaration vs expression?

- Declaration: hoisted and named.
- Expression: assigned to variables; not hoisted.

17. What is scope in JavaScript?

The current context of execution, which determines variable accessibility.

18. What is lexical scope?

Scope determined by the physical placement of code in the source file.

19. What is a closure? Give an example.

A function that retains access to its lexical scope even after the outer function has executed.

20. What is an IIFE?

Immediately Invoked Function Expression: executes immediately after being defined.

21. How to create an object in JavaScript?

Using object literals, constructors, or Object.create().

22. Object.create() vs object literal?

Object.create() creates an object with a specified prototype. Literals create plain objects.

23. Array methods like map, filter, reduce, forEach:

- map: returns a new array by applying a function.
- filter: returns a new array with elements that match a condition.
- reduce: reduces array to a single value.
- forEach: executes a function on each array element.

24. How do you clone an object or array?

Using spread operator [...arr] or {...obj}, or Object.assign().

25. What is the spread operator?

Allows expansion of elements in arrays or objects.

26. What is the DOM?

Document Object Model: a programming interface for HTML and XML documents.

27. How do you select elements in the DOM?

Using methods like getElementById, querySelector, getElementsByClassName, etc.

28. What is event bubbling and capturing?

Bubbling: event propagates from target to top.

Capturing: event propagates from top to target.

29. How do you add and remove event listeners?

Using addEventListener() and removeEventListener().

30. What are event delegation and propagation?

Delegation: assigning event handlers to parent elements to handle events from children.

Propagation: flow of event through DOM tree.

31. Synchronous vs asynchronous code?

Synchronous: code runs line by line.

Asynchronous: code runs in parallel, allowing non-blocking operations.

32. What are callbacks?

Functions passed as arguments to be executed later.

33. What is a Promise?

An object representing eventual completion or failure of an async operation.

34. Explain async and await.

Syntax for working with promises in a cleaner way. await pauses execution until promise resolves.

35. What is the Event Loop?

Handles asynchronous callbacks in JavaScript by placing them in a queue to be executed after the main stack is empty.

36. setTimeout vs setInterval?

- setTimeout: runs code once after delay.

- setInterval: runs code repeatedly at intervals.

37. What is this keyword?

Refers to the context in which a function is called.

38. Difference between call, apply, and bind?

- call: calls function with arguments.
- apply: calls function with array of arguments.
- bind: returns a new function with bound context.

39. What is a prototype?

An object from which other objects inherit properties.

40. What is prototypal inheritance?

Inheritance where objects inherit directly from other objects.

41. What are higher-order functions?

Functions that take or return other functions.

42. What is a pure function?

A function that has no side effects and returns same output for same input.

43. What is memoization?

An optimization technique to cache function results.

44. What are template literals?

Strings with embedded expressions using backticks and ${}.

45. What are default parameters?

Function parameters with default values if no value is passed.

46. Rest and spread operators?

- Rest: collects multiple elements into an array.
- Spread: expands elements from an array/object.

47. What is destructuring?

Extracting values from arrays or objects into variables.

48. What are modules (import/export)?

Modules allow code to be split into reusable files. Use export to expose and import to include.

49. Promises and async/await?

Async/await is syntactic sugar over promises, making async code easier to read and write.

50. What are generators?

Functions that can pause execution and resume using yield.

51. try, catch, finally?

Used for error handling. finally always executes.

52. throw vs console.error?

- throw: stops execution and can be caught.

- console.error: logs error to console.

53. How to create a custom error?

Extend the Error class and define your own.

54. What is localStorage and sessionStorage?

APIs to store data in the browser. localStorage persists; sessionStorage lasts for session.

55. Cookies vs localStorage?

Cookies are sent to the server with requests; localStorage is not.

56. What is CORS?

Cross-Origin Resource Sharing: mechanism for allowing restricted resources to be requested from another domain.

57. Difference between document and window?

- window: global object representing browser window.

- document: object representing the DOM.

58. What is debouncing and throttling?

- Debouncing: delays function call until pause.

- Throttling: limits function call rate.

59. Deep copy vs shallow copy?

- Shallow copy: copies top-level properties.

- Deep copy: copies all levels recursively.

60. What are service workers?

Scripts that run in background for features like offline caching and push notifications.

61. What is fetch API?

Modern way to make HTTP requests. Returns a Promise.

62. How to handle JSON in JavaScript?

Use JSON.stringify() and JSON.parse().

63. How to handle global exceptions?

Use window.onerror or try-catch globally to capture unhandled errors.