JAVASCRIPT TECHNICAL QUESTIONS

Here are 50 common JavaScript technical questions often asked in interviews for MERN stack developers:

### Basic JavaScript

1. \*\*What is JavaScript, and how is it different from Java?\*\*

- JavaScript is a scripting language primarily used for client-side web development, while Java is a programming language used for building standalone applications.

2. \*\*Explain the difference between `let`, `const`, and `var`.\*\*

- `var` is function-scoped, while `let` and `const` are block-scoped. `const` variables cannot be reassigned.

3. \*\*What are data types available in JavaScript?\*\*

- Primitive types: `number`, `string`, `boolean`, `undefined`, `null`, `symbol`, and `bigint`. Non-primitive: `object`.

4. \*\*What is a closure in JavaScript?\*\*

- A closure is a function that retains access to its outer scope even after the outer function has executed.

5. \*\*Explain the concept of hoisting in JavaScript.\*\*

- Hoisting is JavaScript's default behavior of moving variable and function declarations to the top of their containing scope.

6. \*\*What is the difference between `==` and `===`?\*\*

- `==` checks for value equality with type coercion, while `===` checks for strict equality without type coercion.

7. \*\*What are arrow functions, and how do they differ from regular functions?\*\*

- Arrow functions provide a concise syntax and do not have their own `this` context, making them ideal for callbacks.

8. \*\*What is the purpose of the `this` keyword in JavaScript?\*\*

- `this` refers to the object in which the function is currently being executed.

9. \*\*Explain event delegation in JavaScript.\*\*

- Event delegation involves using a single event listener to manage all events of a specific type on multiple child elements.

10. \*\*What is a promise in JavaScript?\*\*

- A promise is an object representing the eventual completion or failure of an asynchronous operation.

### Advanced JavaScript

11. \*\*What is the difference between synchronous and asynchronous programming?\*\*

- Synchronous code executes in sequence, blocking subsequent operations until completion. Asynchronous code allows other operations to continue before the current one finishes.

12. \*\*Explain the concept of callback functions.\*\*

- A callback is a function passed as an argument to another function and executed after a specific event or operation.

13. \*\*What are JavaScript modules, and how do you export/import them?\*\*

- Modules allow code organization and reuse. Use `export` to share and `import` to include in another file.

14. \*\*What is an Immediately Invoked Function Expression (IIFE)?\*\*

- An IIFE is a function that runs immediately after it is defined, often used to create a private scope.

15. \*\*Explain the concept of prototypal inheritance.\*\*

- Prototypal inheritance allows objects to inherit properties and methods from other objects via the prototype chain.

16. \*\*What is an async/await in JavaScript?\*\*

- `async/await` syntax is used to handle promises in a more readable way, allowing asynchronous code to be written like synchronous code.

17. \*\*How do you handle errors in JavaScript?\*\*

- Use `try...catch` blocks to handle errors gracefully in JavaScript.

18. \*\*What is the event loop, and how does it work?\*\*

- The event loop allows JavaScript to perform non-blocking operations by offloading operations to the system kernel.

19. \*\*What are generators in JavaScript?\*\*

- Generators are functions that can be paused and resumed, allowing control over the function execution with `yield`.

20. \*\*What is the purpose of `Object.freeze()`?\*\*

- `Object.freeze()` prevents modification of existing property attributes and values, making an object immutable.

### JavaScript and DOM

21. \*\*How do you select DOM elements in JavaScript?\*\*

- Use methods like `getElementById()`, `querySelector()`, and `querySelectorAll()` to select DOM elements.

22. \*\*How can you add and remove classes from an element using JavaScript?\*\*

- Use `classList.add()` to add and `classList.remove()` to remove classes.

23. \*\*What is event bubbling and capturing in the DOM?\*\*

- Event bubbling propagates events from child to parent, while capturing goes from parent to child.

24. \*\*How do you prevent the default behavior of an event in JavaScript?\*\*

- Use `event.preventDefault()` to stop the default behavior of an event.

25. \*\*What is the difference between `innerHTML` and `innerText`?\*\*

- `innerHTML` retrieves or sets HTML content, while `innerText` works only with text content, ignoring HTML tags.

26. \*\*How do you clone a DOM node?\*\*

- Use `node.cloneNode()` to create a copy of a DOM node.

27. \*\*Explain the concept of a virtual DOM.\*\*

- A virtual DOM is a lightweight copy of the real DOM, used to optimize updates and improve performance in libraries like React.

28. \*\*What is the difference between `addEventListener` and `onclick`?\*\*

- `addEventListener` allows multiple event handlers, while `onclick` overwrites any existing handler.

29. \*\*How do you create a new element in the DOM using JavaScript?\*\*

- Use `document.createElement()` to create a new DOM element.

30. \*\*What is the `dataset` property in JavaScript?\*\*

- The `dataset` property provides access to custom data attributes (`data-\*`) on DOM elements.

### JavaScript and JSON

31. \*\*What is JSON, and how is it used in JavaScript?\*\*

- JSON (JavaScript Object Notation) is a lightweight data interchange format used to store and exchange data.

32. \*\*How do you parse a JSON string in JavaScript?\*\*

- Use `JSON.parse()` to convert a JSON string into a JavaScript object.

33. \*\*How do you stringify a JavaScript object to JSON?\*\*

- Use `JSON.stringify()` to convert a JavaScript object into a JSON string.

34. \*\*What is the purpose of `JSONP`?\*\*

- JSONP (JSON with Padding) is used to overcome the cross-origin resource sharing (CORS) restrictions in browsers.

35. \*\*How do you handle nested JSON data in JavaScript?\*\*

- Access nested JSON data using dot notation or bracket notation.

36. \*\*What are the common data types used in JSON?\*\*

- JSON supports `object`, `array`, `string`, `number`, `boolean`, and `null`.

37. \*\*How do you merge two JSON objects in JavaScript?\*\*

- Use `Object.assign()` or the spread operator (`...`) to merge JSON objects.

38. \*\*Explain how to handle JSON arrays in JavaScript.\*\*

- Use array methods like `map()`, `filter()`, and `reduce()` to manipulate JSON arrays.

39. \*\*What are the security considerations when working with JSON data?\*\*

- Validate and sanitize JSON data to prevent injection attacks and ensure secure data handling.

40. \*\*How do you fetch JSON data from an API in JavaScript?\*\*

- Use `fetch()` or `XMLHttpRequest` to retrieve JSON data from an API.

### JavaScript Best Practices

41. \*\*What are some best practices for writing clean JavaScript code?\*\*

- Use meaningful variable names, write modular code, comment your code, and follow consistent coding styles.

42. \*\*How do you optimize JavaScript performance?\*\*

- Minimize DOM manipulations, use asynchronous programming, and optimize loops and algorithms.

43. \*\*What is the purpose of using linting tools in JavaScript?\*\*

- Linting tools help identify and fix errors and enforce coding standards in JavaScript.

44. \*\*How do you handle cross-browser compatibility issues in JavaScript?\*\*

- Use feature detection, polyfills, and testing frameworks to ensure cross-browser compatibility.

45. \*\*What are some common JavaScript design patterns?\*\*

- Common patterns include module, singleton, observer, and factory patterns.

46. \*\*How do you ensure code reusability in JavaScript?\*\*

- Write modular code, use functions and classes, and follow DRY (Don't Repeat Yourself) principles.

47. \*\*What is the purpose of using `use strict` in JavaScript?\*\*

- `use strict` enforces stricter parsing and error handling in JavaScript, reducing silent errors.

48. \*\*How do you debug JavaScript code effectively?\*\*

- Use browser developer tools, `console.log()`, and breakpoints to debug JavaScript code.

49. \*\*What is functional programming, and how is it applied in JavaScript?\*\*

- Functional programming is a paradigm that treats computation as the evaluation of mathematical functions, emphasizing immutability and pure functions.

50. \*\*How do you manage state in JavaScript applications?\*\*

- Use state management libraries like Redux, Context API, or React's useState and useReducer hooks to manage state.

These questions cover a wide range of fundamental and advanced topics that are important for MERN stack developers to master.