# java script interview Questions

📀 SQL Interview!!

Ocean

MONGODB ——

React Notes

Project ideas

NodeJS Backend

DSA

java-script assignment By Nitin

Oops Master

## java script interview Questions

1. What is JavaScript?
   JavaScript is a high-level, interpreted scripting language primarily used for web development to create interactive and dynamic web pages. JavaScript runs in browsers and can be used server-side with environments like Node.js.
2. How would you explain the difference between JavaScript and ECMAScript?
   The difference between JavaScript and ECMAScript is that JavaScript is a general-purpose scripting language, while ECMAScript (often abbreviated as ES) is the standardized specification that defines its core features. Think of ECMAScript as the blueprint and JavaScript as one of the implementations of that blueprint.

3. How is JavaScript different from Java?
   The difference between JavaScript and Java is that Java is an OOP programming language while JavaScript is an OOP scripting language. JavaScript is interpreted and follows dynamic typing. Java is compiled and strongly typed.
4. What are the primitive data types in JavaScript?
   The primitive data types in JavaScript are string, number, boolean, undefined, null, symbol and BigInt.
5. How would you explain the concept of undefined in JavaScript?
   Undefined in JavaScript is a primitive value automatically assigned to variables that have been declared but not yet initialized. It represents the absence of a defined value.
6. What is null in JavaScript?
   Null in JavaScript is a special value that signifies no value or no object. It needs to be explicitly set and denotes the intentional absence of any value.
7. How can you differentiate null from undefined? You can differentiate null from undefined as undefined is the default state of a declared variable that hasn't been assigned a value, whereas null is an explicit assignment indicating the deliberate absence of a value.
8. What is NaN? How can you check if a value is NaN? NaN stands for "Not-a-Number" and indicates a value that cannot be represented as a valid number. To check if a value is NaN, use the global function NaN() or the more reliable Number is NaNO.
9. Explain variable hoisting. Variable Hoisting is a concept or behavior in JavaScript where variable and function declarations are moved to the top of their containing scope during the compilation phase.
10. What is the difference between let, const, and var? The difference between let, const, and var is that var declarations are globally scoped or function scoped while let and const are block scoped. var and let allows for reassignment, const creates a read-only reference, meaning the value it holds cannot be changed after declaration.

11. What is an Immediately Invoked Function Expression (IIFE)? An Immediately Invoked Function Expression (or IIFE) is a JavaScript function that is defined and executed immediately after its creation.

12. How can you explain closures in JavaScript? Closures in JavaScript are functions which retains access to variables from its enclosing scope, even after that outer function has finished executing

13. What is the this keyword? How does it work?
The this keyword in JavaScript refers to the current execution context or the object that the function is bound to. The value of this keyword changes if its call type is a method, constructor or a standalone function.

14. How does bind(), call(), and apply() work?
bind(), call(), and apply() are methods used to change the context of this keyword within a function. bind() creates a new function with a specified this value, while call() and apply() immediately invoke the function with the provided this value and arguments.

15. What is the prototype chain?
The prototype chain is a mechanism that defines how objects inherit properties and methods from their prototype objects. JavaScript looks up the prototype chain to find it in higher-level prototypes, when a property or method is not found on an object.

16. How can you explain event delegation?
Event delegation is a method where a single event handler is placed on a common parent element of multiple child elements. The parent element captures and handles events triggered by the child elements.

17. What are JavaScript Promises?
JavaScript Promises are objects representing the eventual completion or failure of an asynchronous operation. Promises in JavaScript provide a cleaner and more structured way to handle asynchronous code compared to callback functions, making it easier to work with asynchronous tasks like HTTP requests.

18. How do you handle errors in JavaScript?
Use try-catch block to handle errors in JavaScript. Errors are thrown explicitly using the throw statement. The try statement is used to define a code block to

be executed. The catch statement defines a code block that handles any errors that occur during execution. The finally block defines a code block that should always be executed, regardless of the outcome of the try statement. The throw statement can be used to define a custom error caught and handled by the catch statement.

19. How does the Event Loop work in JavaScript?
    Event Loop in JavaScript continuously checks the call stack for executed functions and the message queue for events or tasks to process ensuring that JavaScript remains single-threaded while handling asynchronous operations efficiently.

20. What is the difference between == and ===?
    The difference between == and === in JavaScript is that == is a loose equality operator that compares values after type coercion, while === is a strict equality operator that compares both values and types. === requires both values to be of the same type and have the same value, making it a safer choice for most comparisons to avoid unexpected type conversions.

21. How can you explain the importance of the use strict directive? The importance of use strict directive is that it enforces a stricter set of rules and helps catch common coding mistakes. It prevents the use of undeclared variables, eliminates ambiguous behavior, and encourages o cleaner, more reliable codebase.

22. How would you explain the concept of Callback Hell or Pyramid of Doom?
    Callback Hell, or Pyramid of Doom, occurs when multiple nested collback functions are used in asynchronous JavaScript code, leading to deeply indented and hard-to-read code structures making code maintenance and debugging challenging.

23. How do you avoid Callback Hell? To avoid callback Hell use Promises, async/await, or modularize the code into smaller functions.

24. What is a callback function? A callback function is a function passed as an argument to another function, which is then invoked or executed at a later point in the program's execution.

25. Can you explain how map, reduce, and filter methods work? map), reduce), and filter() are array methods in JavaScript. map() transforms each element in

an array into a new array based on a provided function, reduce() reduces an array to a single value by applying a function cumulatively to elements, filter creates a new array with elements that pass a given test.

26. What is a closure? Can you give a practical example? Closure in JavaScript is when a function retains access to variables from its enclosing scope, even after that outer function has finished executing. Practical examples include creating private variables and functions in JavaScript, maintaining state in event handlers, and implementing data hiding patterns.
27. What is a closure? Can you give a practical example? Closure in JavaScript is when a function retains access to variables from its enclosing scope, even after that outer function has finished executing. Practical examples include creating private variables and functions in JavaScript, maintaining state in event handlers, and implementing data hiding patterns.
28. How do you clone an object in JavaScript? To clone an object in JavaScript use methods like Object.assign), the spread operator (...), or by create a custom cloning functions.
29. Can you explain how to create and use JavaScript Promises? To create a Promise, use the Promise constructor with two parameters: resolve and reject. Then use then) and .catch() to handle successful and failed outcomes, respectively.
30. How does JavaScript handle asynchronous operations? JavaScript handles asynchronous operations using mechanisms like callbacks, Promises, async/await, and the Event Loop.
31. What distinguishes a function expression from a function declaration? A function expression is distinguished from a function declaration as function declaration defines a named function with a specific name and is hoisted to the top of its containing scope. A function expression assigns an anonymous or named function to a variable, and it is not
hoisted.
32. How can you explain the Document Object Model (DOM)? The Document Object Model (DOM) is a programming interface and representation of structured documents. It represents a web page's structure as a tree of

objects, where each object corresponds to a part of the page, such as an element or an attribute.

33. How do you select an element in the DOM? To select an element in the DOM use methods like document.getElementById(), document.querySelector(), or document.getElementsByClassName(), depending on the selection criteria.
34. What is the difference between innerHTML and textContent? The difference between innerHTML and text Content is that innerHTML retrieves or sets the HTML content within an element, while textContent retrieves or sets only the text content, excluding any HTML tags.
35. How do you add or remove a class from an element in the DOM? To add a class using the element.classList.add('classname) method and remove it using the element.classList.remove(classname') method.
36. How would you explain event bubbling and event capturing? Event bubbling and event capturing are two phases of event propagation in the DOM. In event bubbling, the event starts from the target element that triggered the event and bubbles up to the root of the DOM. In event capturing, the process is reversed and the event descends from the root to the target element.
37. What is the purpose of the data- attribute? The dato-attribute allows to storage of custom data on an element, providing a way to store extra information that doesn't have any visual representation.
38. How do you create a new element and add it to the DOM? To create a new element and add it to the DOM, use the document.createElement('elementName') method. Use the parentNode.appendChild(new Element) method after setting its Properties.
39. How do you attach an event handler to a DOM element? To attach an event handler to a DOM element use the element.addEventListener('eventname', handler Function) method.
40. How can you prevent the default behavior in an event handler? To prevent the default behavior in an event handler use the event.preventDefault() method within the handler function.
41. How would you explain the importance of document.ready in jQuery? document.ready in jQuery ensures that the DOM is fully loaded and ready for

manipulation. It guarantees that scripts run only after the entire page's elements are available, preventing potential errors or unexpected behaviors.

42. Can you explain the concept of shadow DOM?
The shadow DOM enables web developers to encapsulate their code by attaching separate DOM trees to specific elements. This mechanism isolates the internal structure, composition, and styling of these elements from the main document, ensuring functional and styling independence. It becomes easier to manage and maintain the code, by keeping them hidden from the rest of the document.

43. What is a JavaScript generator and how is it different from a regular function?
A JavaScript generator is a special type of function that can pause its execution and later resume from where it left off. Regular functions run to completion when invoked and generators produce a sequence of values using the yield keyword and are controlled by the next() method.

44. What is a Proxy in JavaScript?
A Proxy in JavaScript is an object that wraps another object (target) and intercepts operations, like reading or writing properties, offering a way to customize default behaviors.

45. How does JavaScript's async and await work?
An asynchronous function in JavaScript returns a promise. The await keyword can be used inside the function to pause its execution until the promise settles, simplifying the flow of asynchronous code.

46. Can you explain JavaScript's Module pattern?
JavaScript's Module pattern provides a way to create private scopes and expose only chosen parts to the external world. It's a design pattern that utilizes closures to encapsulate functionality, allowing for public and private access levels

47. What is a Service Worker and what is it used for? A Service Worker is a script running in the background of a browser, separate from the web page. A Service Worker is used for caching resources, enabling offline capabilities, and intercepting network requests to enhance performance and user experience.

48. What is a Web Worker? A Web Worker is a script executed in the background, on a separate thread from the main execution thread. A Web Worker ensures long- running tasks don't block the main thread, keeping web applications responsive.
49. How can you explain the concept of memoization? Memoization in JavaScript is an optimization technique that stores the results of expensive function calls and returns the cached result when the same inputs reoccur thus reducing the computational time.
50. How do you handle state management in large applications? Large applications handle state management using centralized state management solutions such as Redux, Vuex, or MobX. These libraries offer a structured and predictable method for managing and updating state across components.
51. What are the advantages and disadvantages of using JavaScript frameworks/libraries like React or Angular? Using JavaScript frameworks/libraries like React or Angular provides advantages such as rapid development, reusable components, and robust ecosystems. The main disadvantages include steep learning curves. potential overengineering, and increased initial load times due to library overheads.
52. What are template literals in ES6? Template literals in ES6 are string literals allowing embedded expressions. They utilize backticks () instead of quotes and can span multiple lines, making string interpolation more intuitive.
53. Can you explain the spread and rest operators? The spread operator (...) expands an array or object into its elements or properties. The rest operator, also represented by.... collects the remaining elements or properties into an array or object.
54. What are arrow functions? How are they different from regular functions? Arrow functions are a concise way to write functions in ES6. Arrow functions differ from regular functions in syntax and behavior. particularly in their handling of the this keyword, which arrow functions do not bind.
55. How can you explain destructuring assignment? Destructuring assignment in ES6 allows unpacking values from arrays or properties from objects into distinct variables, simplifying the extraction of multiple properties or values.
    <

56. What are default parameters in ES6? Default parameters in ES6 allow functions to have predefined values for arguments that are not passed, ensuring the function behaves correctly even if some parameters are missing.

57. How would you explain the import and export statements in ES6 modules?
The import and export statements in ES6 modules facilitate modular programming by allowing developers to split code into multiple files and then import or export functionalities as required.

58. What are JavaScript Symbols?
JavaScript Symbols are unique and immutable data types introduced in ES6. They often serve as object property keys to ensure uniqueness and avoid overwriting existing properties

59. What is the concept of iteration in ES6?
Iteration in ES6 is a new mechanism for traversing over data. New constructs like the for...of loop, are used to seamlessly iterable objects, such as arrays, strings, maps, and sets, providing a more intuitive way to loop through items.

60. What are JavaScript Sets and Maps?
JavaScript Sets are collections of values where each value must be unique. JavaScript Maps are collections of key-value pairs where keys can be any type. Both of these collections are introduced in ES6 to supplement traditional objects and arrays.

61. What is async functions in ES2017?
Async functions are an enhancement to promises, allowing to write asynchronous code in a more synchronous-like fashion using the async and await keywords.

62. What is the Virtual DOM in React? The Virtual DOM in React is a lightweight representation of the actual DOM. Virtual DOM enables efficient updates and rendering by minimizing direct interactions with the actual DOM.

63. How can you explain the difference between state and props in React? The difference between state and props is that the state in React represents the internal data of a component that can change over time. Props are immutable data passed from parent to child components.

64. How does Angular's two-way data binding work? Angular's two-way data binding automatically synchronizes the model and the view. When the model changes, the view reflects the change, and vice versa, without additional code to detect and respond to these changes.

65. What is the difference between Angular and React? The difference between Angular and React is that Angular is a full- fledged framework offering a wide array of built-in tools and features, while React is a library primarily focused on building user interfaces. They have different philosophies, such as Angular using two-way data binding and React opting for one-way data flow.

66. How can you explain the concept of a hook in React? A hook in React is a function that lets you tap into React features, like state and lifecycle methods, from function components, making them more versatile without converting them to class components.

67. How do you handle state management in React? React handles state management using component local state, context, or third-party libraries like Redux or MobX, depending on the application's complexity and requirements.

68. What is Vue.js and how is it different from React and Angular? Vue.js is different from Angular and React as Vue.js is a progressive JavaScript framework for building user interfaces. Unlike React, which is a library, and Angular, a comprehensive framework. Vue offers a middle ground with an easier learning curve, combined with a flexible and modular approach.

69. How would you explain the concept of a service in Angular? A service in Angular is a class that provides reusable data or functionalities across the application. It is used for tasks like HTTP requests, logging, or data sharing between components.

70. What is Redux and how does it work? Redux is a predictable state management library for JavaScript applications. Redux maintains application state in a single immutable object, with changes made through pure functions called reducers. ensuring a consistent and centralized data flow.
    <

71. What is the purpose of a reducer in Redux? A reducer in Redux is a pure function that takes the current state and an action, then returns a new state. It

defines how the application's state changes in response to actions dispatched to the store.

72. How can you explain the concept of the Critical Rendering Path? The Critical Rendering Path is the sequence of steps browsers undergo to convert HTML, CSS, and JavaScript into pixels rendered on the screen. Optimizing this path ensures web pages render quickly and responsively.
73. What are the different ways to include JavaScript in HTML? JavaScript is included in HTML using the <script> tag either inline, by placing code directly between the tags, or externally, by linking to an external js file using the src attribute.
74. What are Progressive Web Apps (PWAs)? Progressive Web Apps (PWAs) are web applications that offer native- app-like experiences with features like offline access, push notifications, and installation on the home screen, while still being accessible via browsers.
75. How can the performance of a JavaScript application be improved? Performance of a JavaScript application is improved by optimizing code. minimizing DOM interactions, leveraging browser caching, deferring non-critical JavaScript, and using Webpack tool for bundling and minification.
76. How can you explain the concept of Lazy Loading?
Lazy Loading is a performance optimization technique where specific assets, like images or scripts, are loaded only when needed or when they appear in the viewport, reducing initial load times.
77. How do you ensure that your JavaScript code is cross-browser compatible? Ensuring cross-browser compatibility involves using feature detection, leveraging Babel for transpilation, and testing the code across various browsers and browser versions.
78. How do you debug JavaScript code?
JavaScript code is debugged using browser developer tools, setting breakpoints, inspecting variables, and utilizing console.log() statements or more advanced methods like profiling and monitoring.
79. What are the common performance bottlenecks in JavaScript applications? Common performance bottlenecks include extensive DOM manipulations,

memory leaks, unoptimized images, synchronous blocking calls, and redundant or unused code.

30. What is the importance of Webpack in modern web development? Webpack is vital in modern web development as it's a module bundler and task runner. Webpack bundles JavaScript files, stylesheets, image and other assets into a single package, optimizing and transforming them for optimal performance,

31. How can you explain the concept of tree shaking?
    Tree shaking is an optimization technique used in module bundling. where dead code, or unused modules, are eliminated from the final bundled file, resulting in smaller and more efficient output.

32. What is unit testing in JavaScript? Unit testing in JavaScript involves testing individual units or components of code in isolation to ensure that each part functions as intended.

33. How do you perform testing in JavaScript? Testing in JavaScript is performed using testing libraries or frameworks, writing test cases, and then running these tests to verify code correctness.

34. What are some JavaScript testing libraries/frameworks? Popular JavaScript testing libraries/frameworks are Jasmine, Jest, Mocha, Chai, and Karma.

35. What is Test-Driven Development (TDD)? Test-Driven Development (TDD) is a software development methodology where tests are written before the actual code, ensuring that code is developed with testing in mind and meets defined requirements.
    BS. Explain the concept of mocking in testing?
    Mocking in testing involves creating mock objects or functions to replicate and control the behavior of real objects, enabling isolated testing of specific parts without actual dependencies.

36. What is the difference between end-to-end testing and unit testing? The difference between end-to-end testing and unit testing is that end-to- end testing evaluates the entire application's flow in a real-world scenario. while unit testing focuses on individual components in isolation.

37. What is the importance of code linting?
    Code linting is crucial for maintaining code quality, as it detects and warns

about stylistic errors, potential bugs, and deviations from coding standards.

88. How do you ensure that your JavaScript code follows coding standards? Ensuring JavaScript code adheres to coding standards involves using tools like ESLint or TSLint, setting up custom rules, and integrating them into the development process.

89. What is the importance of Continuous Integration (CI) and Continuous Deployment (CD) in software development? Continuous Integration (CI) and Continuous Deployment (CD) are vital for ensuring code consistency, automating testing, and speeding up the delivery of software updates or features to end users.

90. What are the key principles of writing clean code? Key principles of writing clean code are writing readable and understandable code, keeping functions and modules short, using meaningful names, avoiding code duplication, and adhering to the Single Responsibility Principle.

91. Write a function to reverse a string Use the built-in JavaScript methods, to reverse a string.

92. How would you find the first non-repeating character in a string? Iterate through the string and use a frequency counter. <

93. How can you write a function to determine if a string is a palindrome? A string is a palindrome if it reads the same backward as forward.

94. How would you merge two sorted arrays? Use a two-pointer technique to merge sorted arrays.

95. How to write a function to implement a debounce? Debounce ensures a function doesn't fire too frequently.

96. How to implement a basic version of Promise? A basic Promise implementation involves resolve and reject functions.

97. How to write a function to deep clone an object? You can use recursion to deep clone objects.

98. How would you implement a basic pub-sub system? A pub-sub system involves subscribing to events and publishing them.

99. How can you write a function to calculate the Fibonacci sequence? A recursive approach to compute Fibonacci.

## Given by Pawan sir Questions

1 What is Javascript.

2 How Borwser Execute Javascript.
3 What are features of Javascript.
4 What is DOM ?
5 How to perform click event on Dom Using Javascript
6 What is alert() and confirm() ?
7 What is difference between undefined and null
8  What is difference between == and ===
9  What is function . Can we overload function in javascript.
10 What is callback ? Explain with suitable example.
11 What is promise? What are advantage of promise over callback.
12 What is callback hell.
13 What is  closure ?
14 What is IIFEs(Immediately Invoked Function expression).
15 What is annonymous function ?
16 What is hoisting in javascript.
17 What is difference between var, let and const keyword.
18 Explain use strict
19 What is event bubbling and event capturing
20 What are the primitive datatype in javascript
21 What are different types of popup boxes available in javascript
22 What will happen if an infinite while loop is run in javascript
23 List HTML DOM mouse events onclick,ondblclick, onmouseover, onmouseout, onmouseup,onmousemove
24 How to get the last index of a string in javascript lastIndexOf
25 Describe negative infinity in javascript
26 Explain await and aync ? How to use await and async?
27 How to handle the excption in javascript.

28 What is Node.js? Explaing the advantage of Node.js Over java and php
29 What are the limitation of Node.Js
30 How Node.Js Works
31 How Node.js is single threaded
32 Explain any five built-in package/Dependency name  in node.js
33 What is module in node.js
34 what is Module.exports
35 How to create server in node.js
36 How node.js handle multiple request
37 How to use url module in node.js
38 What is setInterval, setTimeout
39 What is _dirname and _filename
40 What is synchronous/Blocking and Asynchronous/Non-blocking code in node.js
41 What is file system in node.js
42 What are the differenct type of flag using in node.js
43 What is stream in node.js? Explain the types of stream
44 How to pipe stream in node.js
45 What is request and response in node.js
46 What is package.json and package.lock
47 What is npm ? How to install dependency/module at application level and Environment elevel
48 How do you manage packages in your node.js project
49 How Node.js is better then other framework
50 What are the some commonly used timing features of Node.js
51 What is fork in node.js
52 How do you create a simple server in node.js that return hello world
53 How many types of API functions are there in node.js
54 What is REPL and how to use it?
55 What is purpose of module.exports.
56 What is an event-loop in Node.js
57 If Node.js is single threaded then how does it handle multiple request/ concurrency
58 Differenctiate between process.nextTick() and setImmediate()
59 What is node.js stream
60 What is middleware

61 Explain what a reactor pattern in node.js
62 Describes the exit code of Node.js
63 What is an EventEmitter in node.js
64 What is a thread pool and which library handles it in node.js
65 What is purpose of NODE_ENV
66 How would you connection mongodb database to node application
67 What are different type of http request
68 What is difference between get and post
69 What is query string and how to send the data in get request
70 What is the use body parser
71 How to set the path of static file in express
72 What are types of Middleware in express ? Explain with suitable example
73 Does order of middleware matters in express.
74 What is express.js
75 What are some distinctive features of Express
76 is Express.js fron-end or backend framework?
77 Why do we use express.js
78 What is difference between express.js and node.js
79 What do you understand by Scaffolding in Express.Js
80 Which are the argument available to an Express.Js route handler function
81 How can you allow CORS in Express.jS?
82 How can you deal with Error handling in Express.js? Explain with an example
83 Write a code to start serving static file Express.JS
84 How can we render plain HTML in express
85 How can we send the data while rendering page in express
86 How to enable debugging in express app?
87 What is routing and How routing works in express
88 How dynamic routing works in express.js