Sign in with Google

"Huge timesaver. Worth the money"

"It's an excellent tool"

"Fantastic catalogue of questions"

your next tech interview with

Explore our carefully curated catalog of interview essentials covering full-stack, data structures and algorithms, system design, data science, and machine learning interview questions

Start preparing now

Web & Mobile Dev | Data Structures & Algorithms | System Design | Machine Learning & Data Science

ADO.NET | Agile & Scrum | Android | Angular | AngularJS

ASP.NET | ASP.NET MVC | ASP.NET Web API | AWS

Azure | C++ | C# | Cosmos DB | CSS

Dependency Injection | DevOps | Django | Entity Framework

Express.js | Flutter | GIT | Golang | GraphQL

HTML5 | Ionic | Java | JavaScript | jQuery

Kotlin | Laravel | LINQ | MongoDB | .NET Core

Next.js | Node.js | Objective-C | OOP | PHP

PWA | Python | React | React Native

Reactive Programming | Reactive Systems | Redis | Redux

Ruby Ruby on Rails Rust Spring SQL Swift

T-SQL Testing TypeScript UX Design Vue.js

WCF Web Security WebSockets WPF Xamarin

100 JavaScript interview questions

Only coding challenges ○ Reset progress Search...

Topic progress: 0%

JavaScript Fundamentals

1. What are the **data types** present in JavaScript? Question ○
2. What is the difference between **null** and **undefined**? Question ○
3. How does JavaScript handle **type coercion**? Question ○
4. Explain the concept of **hoisting** in JavaScript. Question ○
5. What is the **scope** in JavaScript? Question ○
6. What is the difference between `==` and `===` ? Question ○
7. Describe **closure** in JavaScript. Can you give an example? Question ○
8. What is the '**this** keyword' and how does its context change? Question ○
9. What are **arrow functions** and how do they differ from regular functions? Question ○
10. What are **template literals** in JavaScript? Question ○

JavaScript Functions and Higher-Order Functions

11. What is a **higher-order function** in JavaScript? Question ○
12. Can functions be assigned as values to variables in JavaScript? Question ○
13. How do **functional programming** concepts apply in JavaScript? Question ○
14. What are **IIFEs** (Immediately Invoked Function Expressions)? Question ○

15. How do you create **private variables** in JavaScript? Question ○

JavaScript Objects and Prototypes

16. How do you create an **object** in JavaScript? 🔒 Question ○

17. What are **prototypes** in JavaScript? 🔒 Question ○

18. Explain **prototypal inheritance**. 🔒 Question ○

19. What is the difference between **object literals** and **constructor functions**? 🔒 Question ○

20. How do you add or remove properties from an **object**? 🔒 Question ○

Asynchronous JavaScript

21. What is the **event loop** in JavaScript? 🔒 Question ○

22. Explain how **callbacks** work in JavaScript. 🔒 Question ○

23. What are **promises** and how do they manage **asynchronous code**? 🔒 Question ○

24. Explain **async/await** in JavaScript and how it differs from Promises. 🔒 Question ○

25. What is the **Job Queue** (or **Microtask Queue**)? 🔒 Question ○

DOM Manipulation and Browser APIs

26. How do you select **DOM elements** using JavaScript? 🔒 Question ○

27. Explain **event propagation** in the DOM. 🔒 Question ○

28. How do you prevent a form from submitting using JavaScript? 🔒 Question ○

29. What are **Web APIs** in the context of JavaScript? 🔒 Question ○

30. How can you manipulate the **browser history** using JavaScript? 🔒 Question ○

ES2015+ and Modern JavaScript Features

31. What are the new features introduced in **ES6**? 🔒 Question ○

32. How do you use **destructuring assignments** in ES6? 🔒 Question ○

33. Explain the use of `const` and `let` keywords. 🔒 Question ○

34. What are **default parameters** in JavaScript functions? 🔒 Question ○

35. Explain the concept of **modules** in ES6. 🔒 Question ○

Event Handling

36. How do you handle **events** in JavaScript? 🔒 Question ○

37. What is **event delegation** and why is it useful? 🔒 Question ○

38. How do you add and remove an **event listener** from an element? 🔒 Question ○

39. Can you explain how “**this**” works in event handlers? 🔒 Question ○

40. What is the difference between **event.preventDefault()** and **event.stopPropagation()**? 🔒 Question ○

Web Storage and Security

41. What is the difference between **localStorage**, **sessionStorage**, and **cookies**? 🔒 Question ○

42. Can you explain **Cross-Site Scripting (XSS)** and how to prevent it? 🔒 Question ○

43. What is **Cross-Origin Resource Sharing (CORS)** and how does it work? 🔒 Question ○

44. How does **content security policy (CSP)** help in preventing security attacks? 🔒 Question ○

JavaScript Debugging

45. What tools and techniques do you use for **debugging JavaScript code**? 🔒 Question ○

46. How do you debug a JavaScript application in the browser? 🔒 Question ○

47. Explain the concept and use of **breakpoints**. 🔒 Question ○

48. How do you handle **exceptions** in JavaScript? 🔒 Question ○

Performance and Optimization

49. What techniques can be used to improve **JavaScript performance**? 🔒 Question ○

50. How does **JavaScript minification** and **bundling** contribute to performance? 🔒 Question ○

51. Discuss the importance of **web page loading time** and **performance**. 🔒 Question ○

52. Explain how **lazy loading** works in JavaScript. 🔒 Question ○

JavaScript Testing

53. What are some JavaScript **testing frameworks** you know? 🔒 Question ○

54. How can you write **unit tests** for JavaScript code? 🔒 Question ○
55. What is **Test-Driven Development (TDD)** in JavaScript? 🔒 Question ○
56. Can you explain the difference between **unit** and **integration testing**? 🔒 Question ○

Networking with JavaScript

57. How do you make **HTTP requests** in JavaScript? 🔒 Question ○
58. What is the difference between **XMLHttpRequest** and **Fetch API**? 🔒 Question ○
59. What is **AJAX**, and how does it work? 🔒 Question ○
60. How do you use **WebSockets** in a web application? 🔒 Question ○

JavaScript Patterns and Best Practices

61. What is a **design pattern** in JavaScript? 🔒 Question ○
62. Can you explain the **module pattern**? 🔒 Question ○
63. What is the **Singleton pattern** in JavaScript? 🔒 Question ○
64. Explain the **Revealing Module pattern**. 🔒 Question ○
65. What are some **best practices** for coding in JavaScript? 🔒 Question ○

JavaScript Algorithms and Data Structures

66. How do you implement a **stack** and a **queue** in JavaScript? 🔒 Question ○
67. Explain how to sort an **array** in JavaScript. 🔒 Question ○
68. How do you check if a **string** is a **palindrome**? 🔒 Question ○
69. Describe a **recursive function** and provide an example. 🔒 Question ○
70. What is the **time complexity** of JavaScript operations? 🔒 Question ○

JavaScript Libraries and Frameworks

71. What is the difference between a **library** and a **framework** in JavaScript? 🔒 Question ○
72. Explain the **Virtual DOM** in **React**. 🔒 Question ○
73. How does **data binding** work in **Angular**? 🔒 Question ○
74. What is **Vue.js** and what sets it apart from other frameworks? 🔒 Question ○

75. Can you describe the **jQuery library** and its applications? 🔒 Question ○

JavaScript ES2017+ and Beyond

76. What are **async iterators** and **generators**? 🔒 Question ○
77. What is the purpose of the `async` keyword? 🔒 Question ○
78. Can you explain the use of `Object.entries()` and `Object.values()` ? 🔒 Question ○
79. How does JavaScript Handle **big integers** (`BigInt`)? 🔒 Question ○
80. What are **dynamic imports** in JavaScript? 🔒 Question ○

Browser Compatibility and Transpilation

81. How do you ensure your JavaScript code is **cross-browser compatible**? 🔒 Question ○
82. What is **Babel** and how is it used in JavaScript development? 🔒 Question ○
83. What are **polyfills**, and when would you use them? 🔒 Question ○

JavaScript and the DOM

84. What is the **Document Object Model (DOM)**? 🔒 Question ○
85. How do you create, append, or remove an element from the **DOM**? 🔒 Question ○
86. Describe different ways to find or access HTML elements in the **DOM**. 🔒 Question ○
87. Explain the difference between `innerHTML` and `textContent` . 🔒 Question ○
88. How do you handle **DOM events** in a memory-efficient way? 🔒 Question ○

Tooling and Build Systems

89. What is **npm**, and how do you use it? 🔒 Question ○
90. Discuss the role of **Webpack** in modern JavaScript development. 🔒 Question ○
91. What is a **source map**? 🔒 Question ○
92. How do you use **ESLint** for maintaining JavaScript code quality? 🔒 Question ○
93. What is **continuous integration/continuous deployment (CI/CD)** in the context of JS development? 🔒 Question ○

JavaScript and the Web Platform

94. What is the **Window** object and its significance? 🔒 Question ○
95. Explain the **Document** object. 🔒 Question ○
96. What new features does **HTML5** bring to JavaScript development? 🔒 Question ○
97. Discuss the role of JavaScript in **Progressive Web Apps (PWAs)**. 🔒 Question ○

JavaScript and Mobile Development

98. Explain how to use JavaScript for mobile development. 🔒 Question ○
99. What is **React Native** and how does it differ from traditional web apps? 🔒 Question ○
100. How does JavaScript interact with native mobile components? 🔒 Question ○

Unlock interview insights

Get the inside track on what to expect in your next interview. Access a collection of high quality technical interview questions with detailed answers to help you prepare for your next coding interview.

Track progress

Simple interface helps to track your learning progress. Easily navigate through the wide range of questions and focus on key topics you need for your interview success.

Save time

Save countless hours searching for information on hundreds of low-quality sites designed to drive traffic and make money from advertising.

Land a six-figure job at one of the top tech companies

mazon Meta Google Microsoft Open

Ready to nail your next interview?

and get

your

Kickstart your prep

devinterview Blog Terms of use Privacy policy Pricing Contacts