59 NodeJS interview questions to ask your applicants

Questions

1. What is NodeJS and how does it differ from traditional web servers?
2. Can you explain the event-driven architecture of NodeJS?
3. What are some advantages of using NodeJS for web development?
4. How does NodeJS handle asynchronous programming?
5. What is the purpose of the package.json file in a NodeJS application?
6. Can you describe what middleware is in the context of Express.js?
7. What is npm and how does it relate to NodeJS?
8. How do you manage dependencies in a NodeJS project?
9. What is the role of the event loop in NodeJS?
10. Can you explain how to create a simple HTTP server in NodeJS?
11. What are streams in NodeJS and why are they useful?
12. How do you handle errors in a NodeJS application?
13. Can you differentiate between process.nextTick() and setImmediate()?
14. What are environment variables, and how do you use them in NodeJS?
15. How can you improve the performance of a NodeJS application?
16. What are some common security concerns when developing with NodeJS?
17. Can you explain the concept of the 'Single Threaded Event Loop' in NodeJS?
18. How do you handle file operations in NodeJS?
19. What is the purpose of the 'require' function in NodeJS?
20. Can you explain what a callback function is and how it's used in NodeJS?
21. What are Promises in NodeJS and why are they useful?
22. How do you manage different environments in a NodeJS application?
23. What is your approach to logging in a NodeJS application?
24. Can you explain what an event emitter is in NodeJS?
25. How do you handle version management in NodeJS applications?
26. Can you explain the role of the 'cluster' module in NodeJS and when you would use it?
27. Describe how you would implement and use a custom middleware in an Express.js application.
28. How do you manage application configuration in a NodeJS project?
29. What is the difference between local and global npm packages?
30. Can you explain the concept of 'callback hell' and how to avoid it?
31. Describe how you would use the 'fs' module to read and write files asynchronously.
32. What are some best practices for structuring a large-scale NodeJS application?
33. Can you explain how you would use WebSockets in a NodeJS application for real-time communication?
34. How do you handle session management and authentication in a NodeJS application?
35. What is your approach to testing NodeJS applications?
36. Describe a scenario where you had to debug a NodeJS application and how you resolved the issue.
37. Can you explain the difference between synchronous and asynchronous operations in NodeJS?
38. How does NodeJS handle concurrent requests without using threads?
39. What are the advantages and potential pitfalls of using callbacks in NodeJS?
40. How do Promises improve upon callbacks in handling asynchronous operations?
41. Can you explain the concept of 'async/await' and how it relates to Promises?
42. How would you handle multiple asynchronous operations that depend on each other in NodeJS?
43. How does Node.js handle multiple client requests simultaneously without using threads?
44. Can you explain the role of the EventEmitter class in Node.js?
45. What is the difference between 'emit' and 'on' methods in Node.js?
46. How does the event loop work in Node.js, and why is it important?
47. Can you describe a scenario where you would use custom events in a Node.js application?
48. What are the advantages of using an event-driven architecture in Node.js?
49. How do you handle errors in an event-driven Node.js application?
50. Can you explain the concept of 'non-blocking I/O' in the context of Node.js?
51. How does Node.js achieve high performance with its single-threaded model?
52. What is the purpose of the 'process' object in Node.js and how is it related to events?
53. You're tasked with building a high-traffic e-commerce site using Node.js. How would you ensure the application remains responsive during peak loads?
54. Your team is experiencing frequent memory leaks in a Node.js application. Walk me through your process for identifying and resolving these issues.
55. You need to integrate a third-party API that occasionally times out. How would you implement a robust error handling and retry mechanism?
56. Your Node.js application needs to process large CSV files. How would you design this to be memory-efficient and scalable?
57. You're working on a real-time collaborative text editor. How would you implement this using Node.js and WebSockets?
58. Your team is transitioning from callbacks to Promises in a large Node.js codebase. What strategy would you use to manage this transition smoothly?
59. You're tasked with optimizing database queries in a Node.js application. What steps would you take to improve performance?