

Top 50+ Node.js Interview Questions and Answers for 2023

Topics : [Node.js Interview Questions](#)

Written on [November 21, 2023](#)

Basics of Node.js:

1. What is Node.js?

- Node.js is a server-side JavaScript runtime environment built on the V8 JavaScript engine.

2. How does Node.js differ from traditional JavaScript?

- Node.js is not a language; it allows JavaScript to be executed server-side, outside the browser.

3. Explain the event-driven architecture of Node.js.

- Node.js operates on a non-blocking, event-driven model where events trigger asynchronous callbacks.

4. What is an EventEmitter in Node.js?

- EventEmitter is a class in Node.js that allows objects to emit and listen for events.

Node.js Modules:

5. What is a module in Node.js?

- A module is a reusable piece of code that encapsulates functionality.

6. How can you include external libraries in Node.js?

- Use `require('module')` to include external libraries.

npm (Node Package Manager):

7. What is npm?

- npm is the Node Package Manager used for package management and distribution.

8. How do you install packages using npm?

- Use `npm install package-name` to install packages locally and `npm install -g package-name` for global installation.

9. Explain the purpose of package.json.

- `package.json` is a file containing metadata about a Node.js project, including dependencies.

Callbacks and Promises:

10. What is a callback function in Node.js?

- A callback is a function passed as an argument to another function, to be executed later.

11. Explain the concept of callback hell.

- Callback hell refers to the nesting of multiple callbacks, making the code hard to read and maintain.

12. What are Promises in Node.js?

- Promises are objects representing the eventual completion or failure of an asynchronous operation.

13. How do you handle errors in Promises?

- Use `.catch()` or `.then(null, errorHandler)` to handle errors in Promises.

File System:

14. How do you read and write files in Node.js?

- Use `fs.readFile` for reading and `fs.writeFile` for writing files.

15. Explain the difference between fs.readFileSync and fs.readFile.

- `fs.readFileSync` is synchronous, blocking execution, while `fs.readFile` is asynchronous.

16. What is the purpose of the fs.createReadStream method?

- It creates a readable stream to efficiently read large files.

Express.js:

17. What is Express.js?

- Express.js is a web application framework for Node.js, simplifying the process of building robust web applications.

18. How do you install Express.js?

- Use `npm install express` to install Express.js.

19. Explain the routing in Express.js.

- Routing in Express.js defines how the application responds to client requests.

20. What is middleware in Express.js?

- Middleware functions are functions that have access to the request, response, and the next middleware function in the application's request-response cycle.

RESTful APIs:

21. What is RESTful architecture?

- RESTful architecture is a style of designing networked applications using simple HTTP methods.

22. How do you create a RESTful API using Express.js?

- Use Express.js to define routes and handlers for HTTP methods.

23. Explain the HTTP methods used in RESTful services.

- GET (read), POST (create), PUT (update), DELETE (delete).

Asynchronous Programming:

24. What is the event loop in Node.js?

- The event loop is a core concept in Node.js for handling asynchronous operations.

25. How does Node.js handle asynchronous code?

- Through callbacks, Promises, and `async/await`.

26. What is the purpose of the `setImmediate` function?

- `setImmediate` is used to execute a script once the current event loop cycle completes.

Streams:

27. What are streams in Node.js?

- Streams provide an efficient way to read or write data in chunks.

28. Explain the difference between readable and writable streams.

- Readable streams allow reading, while writable streams allow writing.

29. How do you pipe streams in Node.js?

- Use the pipe method to connect the output of one stream to the input of another.

WebSocket:

30. What is WebSocket?

- WebSocket is a communication protocol providing full-duplex communication channels over a single TCP connection.

31. How do you implement WebSocket in Node.js?

- Use the ws library or the socket.io library for WebSocket implementation.

MongoDB and Mongoose:

32. What is MongoDB?

- MongoDB is a NoSQL database.

33. How do you connect to a MongoDB database using Node.js?

- Use the mongodb driver or an ODM like Mongoose.

34. What is Mongoose?

- Mongoose is an ODM (Object-Document Mapper) for MongoDB and Node.js.

35. Explain the schema in Mongoose.

- A schema defines the structure of documents in a collection.

Testing in Node.js:

36. What testing frameworks are commonly used in Node.js?

- Mocha, Jasmine, Jest.

37. How do you write unit tests in Node.js?

- Use testing frameworks like Mocha and assertions libraries like Chai.

Security:

38. How can you prevent common security vulnerabilities in Node.js applications?

- Validate input, use parameterized queries, sanitize user inputs, and keep dependencies updated.

Debugging:

39. How do you debug a Node.js application?

- Use the debugger statement or tools like node-inspector and built-in debugging in VSCode.

Performance Optimization:

40. **What techniques can you use to optimize the performance of a Node.js application?**
- Caching, load balancing, minimizing blocking code, and using a reverse proxy.

Child Processes:

41. **Explain the use of the child_process module in Node.js.**
- It allows running external processes, enabling interaction with the operating system.

Global Objects in Node.js:

42. **What is the global object in Node.js?**

- The global object represents the global namespace in Node.js.

43. **Explain the purpose of __dirname and __filename.**

- __dirname is the name of the directory containing the currently executing script, and __filename is the file name of the current module.

Error Handling:

44. **How do you handle errors in Node.js?**

- Use try-catch blocks, callback error-first pattern, and Promise .catch().

Deployment:

45. **What are some common deployment strategies for Node.js applications?**

- Using containers (Docker), cloud platforms (AWS, Azure, Heroku), and continuous integration.

Scalability:

46. **How can you scale a Node.js application?**

- Horizontal scaling with load balancing, using a reverse proxy, and optimizing code.

Middleware:

47. **Explain the concept of middleware in Express.js.**

- Middleware functions are functions that have access to the request, response, and the next middleware function in the application's request-response cycle.

Cross-Origin Resource Sharing (CORS):

48. **What is CORS, and how can you handle it in Express.js?**

- CORS (Cross-Origin Resource Sharing) is a security feature implemented by browsers. In Express.js, you can use the cors middleware to handle it.

Template Engines:

49. **What is a template engine, and which ones are commonly used with Node.js?**

- A template engine processes templates to produce HTML. Common ones for Node.js are EJS, Handlebars, and Pug.

WebSockets:

50. **How do WebSockets differ from traditional HTTP communication?**

- WebSockets provide full-duplex communication, allowing real-time data transfer, unlike the request-response nature of HTTP.

© Copyright **Aryatechno**. All Rights Reserved. Written tutorials and materials by [Aryatechno](#)

ARYATECHNO