Node.js Interview Questions and Answers

1. What is Node.js?

Node.js is a runtime environment that allows you to execute JavaScript on the server-side. It is built on Google Chrome's V8 JavaScript engine.

2. How is Node.js different from traditional server-side programming languages?

Node.js uses an event-driven, non-blocking I/O model, making it efficient for handling multiple concurrent requests.

3. What is the purpose of the package.json file?

It contains metadata about the project, such as name, version, dependencies, and scripts, and is essential for managing the project.

4. Explain Event-Driven Programming in Node.js.

In Node.js, operations are triggered through events. For example, when an event occurs (e.g., data received), the corresponding callback function is executed.

5. What is the role of the fs module?

The fs module provides an API for interacting with the file system, allowing you to read, write, and manage files and directories.

6. What is middleware in Node.js?

Middleware functions in Node.js are functions that process requests and responses in a pipeline-like structure, often used in Express.js.

7. What is the difference between require and import?

require: CommonJS module system, used in Node.js.

import: ES6 module system, requires the "type": "module" in package.json.

8. How does Node.js handle concurrency?

Node.js handles concurrency using a single-threaded event loop and non-blocking I/O operations.

9. What is the process object in Node.js?

The process object is a global object in Node.js that provides information about the current Node.js process, including environment variables, arguments, etc.

10. What are streams in Node.js?

Streams are instances of EventEmitter that allow reading or writing data continuously. They are used to handle large amounts of data efficiently.

11. What are the types of streams in Node.js?

1. Readable

2. Writable

3. Duplex

4. Transform

12. Explain the role of the EventEmitter in Node.js.

EventEmitter allows you to create, listen to, and handle custom events in Node.js applications.

13. What is the difference between setImmediate and process.nextTick?

process.nextTick: Executes code at the end of the current operation, before I/O.

setImmediate: Executes code in the next iteration of the event loop.

14. What is clustering in Node.js?

Clustering allows you to create multiple instances of a Node.js application to utilize multi-core CPUs.

15. How can you handle errors in Node.js?

By using try-catch blocks, event listeners (e.g., .on('error')), and global handlers

like process.on('uncaughtException').

16. What is the difference between synchronous and asynchronous code in Node.js?

Synchronous: Blocks the execution of subsequent code.

Asynchronous: Executes subsequent code without waiting for the operation to complete.

17. What is npm?

npm (Node Package Manager) is the default package manager for Node.js, used for managing libraries and dependencies.

18. What is the purpose of async/await in Node.js?

async/await simplifies writing asynchronous code by allowing you to write it in a synchronous-like manner.

19. What is the difference between readFile and createReadStream in the fs module?

readFile: Reads the entire file into memory.

createReadStream: Reads the file in chunks, suitable for large files.

20. How does Node.js handle uncaught exceptions?

Using process.on('uncaughtException', callback), but it is better to handle errors locally.

21. What is CORS, and how do you handle it in Node.js?

CORS (Cross-Origin Resource Sharing) allows or restricts requests from different origins. It is handled using middleware like cors in Express.js.

22. What is the difference between res.send() and res.json() in Express.js?

res.send(): Sends data of any type.

res.json(): Sends JSON-formatted data.

23. How can you secure a Node.js application?

1. Use HTTPS.

2. Validate user input.

3. Use environment variables for sensitive information.

4. Implement rate limiting.

5. Sanitize data to prevent SQL injection.

24. What are child processes in Node.js?

Child processes allow you to execute external commands or create new Node.js processes, using the child_process module.

25. What is the difference between app.use() and app.get() in Express.js?

app.use(): Mounts middleware functions.

app.get(): Defines a route handler for GET requests.