1. **What is Node.js?**

**Answer:** Node.js is a **JavaScript runtime** built on Chrome’s V8 engine that allows JavaScript to run on the server. It uses an **event-driven, non-blocking I/O model**, making it lightweight and efficient for real-time applications.

2. **How does Node.js work?**

**Answer:** Node.js runs on a **single-threaded event loop** and uses **asynchronous callbacks** to handle multiple operations without blocking the main thread. It delegates I/O operations to the system kernel and continues executing other tasks.

3. **What is the event loop in Node.js?**

**Answer:** The **event loop** is a mechanism that allows Node.js to perform non-blocking operations by offloading tasks to the system kernel when possible. It continuously checks the call stack and processes events from the **callback queue**.

4. **What is non-blocking I/O?**

**Answer:** Non-blocking I/O means Node.js doesn’t wait for operations like file reading or network requests to complete. Instead, it continues executing other code and uses **callbacks** to handle results when ready.

5. **How is Node.js different from traditional servers (Java/PHP)?**

**Answer:**

* **Node.js** is **single-threaded**, **event-driven**, and **non-blocking**.
* **Java/PHP** are **multi-threaded** and **blocking** (each request creates a new thread).
* Node.js is better for **real-time apps (chat, gaming)**, while Java/PHP are better for **CPU-heavy tasks**.

6. **What is npm?**

**Answer:** npm (**Node Package Manager**) is the default package manager for Node.js. It helps install, manage, and share reusable JavaScript libraries (packages).

7. **What are global objects in Node.js?**

**Answer:** Global objects (e.g., process, console, global, \_\_dirname, \_\_filename) are available in all modules without requiring them.

8. **What is**module.exports**?**

**Answer:** module.exports is used to **export functions, objects, or variables** from a module so they can be used in other files using require().

9. **How do you include external modules?**

**Answer:** Using require():

javascript

Copy

const fs = require('fs'); // Includes the 'fs' (file system) module

10. **What is**package.json**?**

**Answer:** A package.json file stores **project metadata**, dependencies, scripts, and configurations. It’s created using npm init.

**Asynchronous Programming**

11. **What is a callback function?**

**Answer:** A callback is a function passed as an argument to another function and executed after a task completes.

javascript

Copy

fs.readFile('file.txt', (err, data) => {

if (err) throw err;

console.log(data);

});

12. **What is callback hell? How to avoid it?**

**Answer:** Callback hell is **nested callbacks** making code unreadable. Avoid it using:

* **Promises**
* **Async/Await**
* **Modularization** (breaking into smaller functions)

13. **What are Promises?**

**Answer:** Promises represent **future completion** (or failure) of an async operation. They have 3 states:

* **Pending** → **Fulfilled** or **Rejected**

javascript

Copy

const promise = new Promise((resolve, reject) => {

if (success) resolve(data);

else reject(error);

});

14. **What is**async/await**?**

**Answer:** async/await is syntactic sugar over Promises, making async code look synchronous.

javascript

Copy

async function fetchData() {

const data = await someAsyncFunction();

console.log(data);

}

15. **Difference between**setTimeout**and**setImmediate**?**

**Answer:**

* setTimeout(callback, delay) runs after a **minimum delay**.
* setImmediate(callback) runs **after I/O events** in the next event loop cycle.

**Node.js Modules & File System**

16. **What are core modules in Node.js?**

**Answer:** Built-in modules like fs, http, path, events, stream, etc.

17. **How to read a file in Node.js?**

**Answer:** Using fs.readFile():

javascript

Copy

fs.readFile('file.txt', 'utf8', (err, data) => {

if (err) throw err;

console.log(data);

});

18. **What is**Buffer**in Node.js?**

**Answer:** A Buffer is a **temporary storage** for binary data (e.g., reading files, network streams).

19. **What is**stream**in Node.js?**

**Answer:** Streams allow **reading/writing data in chunks** (useful for large files). Types:

* **Readable** (e.g., fs.createReadStream)
* **Writable** (e.g., fs.createWriteStream)

20. **Difference between**require**and**import**?**

**Answer:**

* require → **CommonJS** (Node.js default)
* import → **ES Modules** (used with "type": "module" in package.json)

**HTTP & Web Servers**

21. **How to create an HTTP server in Node.js?**

**Answer:** Using the http module:

javascript

Copy

const http = require('http');

http.createServer((req, res) => {

res.writeHead(200, {'Content-Type': 'text/plain'});

res.end('Hello World!');

}).listen(3000);

22. **What is Express.js?**

**Answer:** Express is a **minimalist web framework** for Node.js that simplifies routing, middleware, and HTTP handling.

23. **What is middleware in Express?**

**Answer:** Middleware are functions that **process requests before they reach routes**. Example:

javascript

Copy

app.use((req, res, next) => {

console.log('Middleware running!');

next();

});

24. **How to handle GET/POST requests in Express?**

**Answer:**

javascript

Copy

app.get('/', (req, res) => res.send('GET Request'));

app.post('/', (req, res) => res.send('POST Request'));

**Error Handling & Debugging**

25. **How to handle errors in Node.js?**

**Answer:**

* **Try-catch** (for synchronous code)
* **Error-first callbacks** ((err, data) => {...})
* **Promises**.catch()
* **Express error middleware** (app.use((err, req, res, next) => {...}))

26. **What is**process.nextTick()**?**

**Answer:** It schedules a callback to run **immediately after the current operation** (before the event loop continues).

27. **How to debug Node.js apps?**

**Answer:**

* console.log()
* **Node.js Inspector** (node --inspect app.js)
* **Debugger** (debugger keyword)

**Database & Security**

28. **How to connect Node.js to a database?**

**Answer:** Using libraries like:

* **MongoDB** → mongoose
* **MySQL** → mysql2
* **PostgreSQL** → pg

29. **What is JWT?**

**Answer:** **JSON Web Tokens (JWT)** are used for **authentication** by generating encrypted tokens.

30. **What is CORS? How to enable it in Express?**

**Answer:** **Cross-Origin Resource Sharing (CORS)** allows requests from different domains. Enable it using:

javascript

Copy

const cors = require('cors');

app.use(cors());