? Node.js Interview Questions and Answers (1–50)

1. What is Node.js?

Answer: Node.js is a runtime environment that allows JavaScript to be run on the server side, built on Chrome's V8 engine.

2. What are the features of Node.js?

Answer:

- Asynchronous and Event-Driven
- Fast execution
- Single-threaded
- Non-blocking I/O
- Open-source

3. What is the V8 engine?

Answer: Google's open-source JavaScript engine that compiles JS into machine code.

4. Is Node.js single-threaded or multi-threaded?

Answer: It uses a single-threaded event loop but handles concurrent requests using background threads.

5. What is NPM?

Answer: Node Package Manager — the default package manager for Node.js.

6. What is a module in Node.js?

Answer: A reusable block of code whose existence does not impact other code.

7. What are the types of modules in Node.js?

Answer:

- Core modules (built-in)
- Local modules
- Third-party modules (via NPM)

8. How do you import a module in Node.js?

Answer:

```
const fs = require('fs');
```

9. What is `package.json`?

Answer: A file that holds metadata about the project and its dependencies.

10. How do you create a `package.json` file?

Answer:

```
npm init
```

11. What is the difference between `dependencies` and `devDependencies`?

Answer:

- `dependencies`: Required for production
- `devDependencies`: Used only during development

12. What is the `require()` function?

Answer: It is used to import modules.

13. What is the `exports` object?

Answer: Used to expose functions or variables from a module.

14. What is the purpose of `fs` module?

Answer: It provides file system operations such as read/write files.

15. How do you read a file using Node.js?

Answer:

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

16. What is the `http` module?

Answer: Used to create server-side HTTP servers and clients.

17. How do you create a simple server in Node.js?

Answer:

```
const http = require('http');
http.createServer((req, res) => {
  res.end('Hello World');
}).listen(3000);
```

18. What is middleware?

Answer: A function that has access to `req`, `res`, and `next()` in Express.js.

19. What is Express.js?

Answer: A minimal and flexible web application framework for Node.js.

20. How do you install Express?

Answer:

```
npm install express
```

21. How do you create a route in Express?

Answer:

```
app.get('/', (req, res) => res.send('Home'));
```

22. What is routing in Express.js?

Answer: Defining endpoints for your application.

23. What is the use of `next()` in Express middleware?

Answer: Passes control to the next middleware function.

24. What is the difference between `app.use()` and `app.get()`?

Answer:

- `use()`: Middleware for all routes
- `get()`: Handles HTTP GET requests

25. What is the purpose of `body-parser`?

Answer: Parses incoming request bodies in middleware (now built into Express).

26. How do you handle form data in Node.js?

Answer: Using `body-parser` or built-in middleware in Express.

27. What is an event loop?

Answer: A loop that handles asynchronous callbacks in a single thread.

28. What is the purpose of `events` module?

Answer: Provides event-driven programming with `EventEmitter`.

29. How do you emit and listen to events?

Answer:

```
emitter.on('event', () => {...});
emitter.emit('event');
```

30. What is `process` object?

Answer: Provides information and control over the current Node.js process.

31. What is the difference between `process.exit()` and `process.kill()`?

Answer:

- `exit()`: Ends the process
- `kill()`: Sends a signal to terminate a process

32. How do you handle exceptions in Node.js?

Answer: Using `try...catch` or `process.on('uncaughtException').`

33. What is callback hell?

Answer: Nested callbacks that become hard to read and maintain.

34. What are Promises in Node.js?

Answer: Objects representing the eventual completion (or failure) of an asynchronous operation.

35. What is async/await?

Answer: Syntactic sugar over Promises for writing cleaner asynchronous code.

36. What is the use of `util.promisify()`?

Answer: Converts callback-based functions to return Promises.

37. What is a stream in Node.js?

Answer: An abstraction for working with streaming data (like reading/writing files).

38. What are the types of streams?

Answer:

- Readable
- Writable

- Duplex
- Transform

39. How do you use `readable` stream?

Answer:

```
const stream = fs.createReadStream('file.txt');
stream.on('data', chunk => { ... });
```

40. How do you handle file uploads in Node.js?

Answer: Use `multer` or `busboy` middleware in Express.

41. How do you connect Node.js with MongoDB?

Answer: Using Mongoose or native MongoDB driver.

42. How do you secure a Node.js application?

Answer:

- Use Helmet
- Sanitize inputs
- Avoid eval()
- Use HTTPS
- Handle errors properly

43. What is CORS?

Answer: Cross-Origin Resource Sharing — a mechanism to allow/disallow resources to be requested from another domain.

44. How do you enable CORS in Node.js?

Answer:

```
npm install cors
```

```
app.use(cors());
```

45. How do you manage environment variables?

Answer: Using `.env` files and the `dotenv` package.

46. How do you deploy a Node.js app?

Answer: Using services like Heroku, Vercel, or traditional VPS.

47. What is clustering in Node.js?

Answer: Running multiple instances of Node.js processes to handle more requests.

48. What is PM2?

Answer: A process manager for Node.js applications with features like clustering and logging.

49. What is the difference between blocking and non-blocking code?

Answer: Blocking code blocks the thread; non-blocking allows other operations to continue.

50. What are some popular Node.js frameworks?

Answer:

- Express.js
- Koa.js
- Nest.js
- Fastify