1. What is Node.js and Express?

- **Node.js:** A JavaScript runtime built on Chrome's V8 engine, enabling server-side JavaScript execution. It allows developers to build scalable, high-performance server-side applications.
- **Express:** A lightweight web application framework built on top of Node.js. It simplifies building web applications and APIs by providing robust features such as routing, middleware, and request/response handling.

---

2. Why do we use Express instead of Node.js?

- Node.js provides basic server capabilities but lacks advanced features for web development.
- Express simplifies tasks like:
    - Routing for different endpoints.
    - Handling middleware to process requests and responses.
    - Parsing request bodies.
    - Managing error handling.
    - Building RESTful APIs efficiently.

---

3. What is JSON?

- **JSON (JavaScript Object Notation):** A lightweight data-interchange format. It's easy for humans to read and write and easy for machines to parse and generate.
- Example:

json

```
{
  "name": "John",
  "age": 30,
  "isDeveloper": true
}
```

---

4. What are the HTTP Methods? Explain each.

HTTP methods define the action to perform on a resource:

- **GET:** Retrieve data from a server.
- **POST:** Send data to the server to create a new resource.
- **PUT:** Update an existing resource or create one if it doesn't exist.
- **PATCH:** Partially update an existing resource.
- **DELETE:** Remove a resource from the server.

---

5. What are the Status Codes?

Status codes indicate the result of an HTTP request:

- **1xx (Informational):** Request received, processing continues (e.g., 101 Switching Protocols).
- **2xx (Success):** Request succeeded (e.g., 200 OK, 201 Created).
- **3xx (Redirection):** Further action needed (e.g., 301 Moved Permanently, 302 Found).
- **4xx (Client Error):** Client-side error (e.g., 400 Bad Request, 401 Unauthorized, 404 Not Found).
- **5xx (Server Error):** Server-side error (e.g., 500 Internal Server Error, 503 Service Unavailable).

---

6. What is an API?

- **API (Application Programming Interface):** A set of protocols and tools that allow two applications to communicate. In web development, APIs enable interaction between a client (frontend) and a server (backend).

---

7. How to Create an API?

Example using Node.js and Express:

1. **Install Node.js and Express:**

npm init -y

npm install express

2. **Create an Express Server:**

```
const express = require('express');
const app = express();

// Middleware to parse JSON
app.use(express.json());

// Define an API endpoint
app.get('/api/users', (req, res) => {
  res.json([{ id: 1, name: 'John' }]);
});

// Start the server
app.listen(3000, () => console.log('Server running on port 3000'));
```

3. Test it using Postman or a browser.

---

8. How do we get and use JWT?

- **JWT (JSON Web Token):** A secure way to transmit information between parties as a JSON object. Used for authentication and authorization.
- **Steps:**
    1. Install JWT library:

```
npm install jsonwebtoken
```

    2. Generate a Token:

```
const jwt = require('jsonwebtoken');

const token = jwt.sign({ userId: 1 }, 'secretKey', { expiresIn: '1h' });
console.log(token);
```

    3. Verify and Use JWT:

```
app.get('/secure-route', (req, res) => {
  const token = req.headers.authorization?.split(' ')[1];

  if (!token) return res.status(401).send('Unauthorized');

  jwt.verify(token, 'secretKey', (err, decoded) => {
    if (err) return res.status(403).send('Forbidden');
    res.send(`Welcome user ${decoded.userId}`);
  });
});
```

---

9. What is Request and Response?

- **Request:** The data sent by the client to the server, including headers, query parameters, body, and method (e.g., GET, POST).
- **Response:** The data sent by the server back to the client. It includes status codes, headers, and the response body.

---

1. What is Node.js?

Node.js is a server-side JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to build scalable network applications using JavaScript.

---

2. What is the difference between Node.js and JavaScript?

- **Node.js**: A runtime environment for executing JavaScript on the server-side.
- **JavaScript**: A programming language primarily used for client-side development in web browsers.

---

3. Is Node.js single-threaded?

Yes, Node.js operates on a single-threaded event loop architecture but can handle multiple concurrent operations using asynchronous I/O.

---

4. What kind of API function is supported by Node.js?

Node.js supports both:

- **Asynchronous, non-blocking APIs** (default)
- **Synchronous APIs** (less common, blocking)

---

5. What is a module in Node.js?

A module is a reusable piece of code in Node.js that can be exported and imported. Examples include built-in modules like fs or custom modules.

---

6. What is npm and its advantages?

npm (Node Package Manager) is a package manager for Node.js.
Advantages:

- Manages dependencies
- Easy sharing of packages
- Access to a large ecosystem of open-source libraries

---

7. What is middleware?

Middleware is a function that processes requests and responses in a Node.js application, typically used in frameworks like Express.js for handling tasks like authentication and logging.

---

8. How does Node.js handle concurrency despite being single-threaded?

Node.js uses the **event loop** and **non-blocking I/O operations** to handle multiple tasks concurrently.

---

9. What is the control flow in Node.js?

Control flow refers to the execution order of asynchronous code using callbacks, promises, or async/await.

---

10. What do you mean by the event loop in Node.js?

The event loop is a mechanism in Node.js that processes asynchronous tasks, I/O operations, and timers in a single-threaded environment.

---

11. What are the main disadvantages of Node.js?

- Single-threaded nature can be limiting for CPU-intensive tasks.
- Callback hell in deeply nested callbacks.
- Less mature ecosystem compared to other languages for some use cases.

---

12. What is REPL in Node.js?

REPL (Read-Eval-Print Loop) is an interactive shell in Node.js for running JavaScript code interactively.

---

13. How to import a module in Node.js?

Use require() to import modules:

javascript

Copy code

const fs = require('fs');

---

14. What is the difference between Node.js and AJAX?

- **Node.js**: Server-side JavaScript runtime.
- **AJAX**: Client-side technique for asynchronous HTTP requests.

---

15. What is package.json in Node.js?

A file that defines project metadata, scripts, and dependencies for a Node.js project.

---

16. What is the most popular Node.js framework used these days?

Express.js is the most popular framework for Node.js.

---

17. What are promises in Node.js?

Promises are objects representing eventual completion or failure of asynchronous operations, used to avoid callback hell.

---

18. What is event-driven programming in Node.js?

A programming paradigm where actions are driven by events, such as user interactions or server requests.

---

19. What is buffer in Node.js?

A buffer is a temporary storage area for handling binary data in streams.

---

20. What are streams in Node.js?

Streams are objects for handling data reading/writing sequentially. Types:

- Readable
- Writable
- Duplex
- Transform

---

21. Explain crypto module in Node.js.

The crypto module provides cryptographic functionalities, such as hashing and encryption.

---

22. What is callback hell?

A situation where multiple nested callbacks make code difficult to read and maintain.

---

23. Explain the use of the timers module in Node.js.

The timers module provides functions like setTimeout, setInterval, and setImmediate to schedule code execution.

---

24. What is the difference between setImmediate() and process.nextTick() methods?

- setImmediate(): Executes in the next iteration of the event loop.
- process.nextTick(): Executes before the next event loop iteration.

---

25. What is the difference between setTimeout and setImmediate() methods?

- setTimeout: Executes after a specified delay.
- setImmediate: Executes immediately after the I/O phase.

---

26. What is the difference between spawn() and fork() methods?

- spawn(): Starts a new process.
- fork(): Creates a child process that shares the parent's memory.

---

27. Explain the use of the passport module in Node.js.

The passport module is used for authentication, supporting strategies like OAuth, JWT, and local login.

---

28. What is a fork in Node.js?

A fork creates a child process for parallel task execution.

---

29. What are the three methods to avoid callback hell?

1. Using promises
2. Using async/await
3. Modularizing code

---

30. What is body-parser in Node.js?

Body-parser is middleware for parsing incoming request bodies.

---

31. What is CORS in Node.js?

CORS (Cross-Origin Resource Sharing) enables sharing resources between different origins.

---

32. Explain the tls module in Node.js.

The tls module provides APIs for secure communication using TLS/SSL.

---

33. What is a cluster in Node.js?

The cluster module allows creating multiple processes to handle workload on multi-core systems.

---

34. How to manage sessions in Node.js?

Sessions are managed using libraries like express-session, storing session data in memory or a database.

---

35. Explain the types of streams in Node.js.

1. Readable
2. Writable
3. Duplex
4. Transform

---

36. How can we implement authentication and authorization in Node.js?

Using libraries like Passport.js or JWT for authentication and middleware for role-based authorization.

---

37. Explain the packages used for file uploading in Node.js.

Packages like multer or formidable are used for handling file uploads.

---

38. How to handle database connections in Node.js?

Use libraries like mongoose for MongoDB or pg for PostgreSQL to manage database connections.

---

39. How to read command-line arguments in Node.js?

Using process.argv:

```
const args = process.argv.slice(2);
```

---

40. What are child processes in Node.js?

Child processes are spawned using the child_process module to run tasks concurrently.