Node JS

Friday, February 14, 2025

11:50 AM

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine that allows you to run JavaScript code server-side. It’s designed to build scalable network applications due to its non-blocking, event-driven architecture.

#### Getting Started with Node.js

1. \*\*Installation\*\*:

- Download and install Node.js from the [official website](https://nodejs.org/).

- Verify installation via terminal or command prompt:

```bash

node -v

npm -v

```

2. \*\*Creating a Simple Server\*\*:

- Create a `server.js` file:

```javascript

const http = require('http');

const hostname = '127.0.0.1';

const port = 3000;

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

res.statusCode = 200;

res.setHeader('Content-Type', 'text/plain');

res.end('Hello World\n');

});

server.listen(port, hostname, () => {

console.log(`Server running at http://${hostname}:${port}/`);

});

```

- Run your server:

```bash

node server.js

```

3. \*\*Understanding npm\*\*:

- `npm` is the Node.js package manager for installing and managing packages.

- Initialize a project:

```bash

npm init

```

- Install a package, e.g., Express.js:

```bash

npm install express --save

```

4. \*\*Building a Simple Express Application\*\*:

- Express is a minimal web application framework for Node.js.

- Create an `app.js` file:

```javascript

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Hello Express!');

});

app.listen(port, () => {

console.log(`App listening at [http://localhost:${port}`](http://localhost:$%7bport%7d%60));

});

```

- Start your application:

```bash

node app.js

```

5. \*\*Asynchronous Programming\*\*:

- Node.js handles operations asynchronously using callbacks, promises, or async/await.

- Example with promises and async/await:

```javascript

const fetchData = () => {

return new Promise((resolve, reject) => {

setTimeout(() => resolve('Data received'), 1000);

});

};

const getData = async () => {

const data = await fetchData();

console.log(data);

};

getData();

```

### Important Node.js Interview Questions

1. \*\*What is Node.js, and why do you use it?\*\*

- Node.js is a runtime environment for executing JavaScript on the server, designed for building scalable applications with non-blocking I/O.

2. \*\*How does Node.js handle asynchronous operations?\*\*

- Through event-driven architecture using callbacks, Promises, and async/await for managing non-blocking operations.

3. \*\*Explain the Node.js event loop.\*\*

- The event loop is a core component of Node.js that waits for and dispatches events or messages in programming. It allows Node.js to perform non-blocking I/O operations.

4. \*\*What is npm, and how do you use it?\*\*

- npm (Node Package Manager) is a tool to manage libraries and dependencies in Node.js projects. It's used to install, publish, and manage node modules.

5. \*\*What are streams in Node.js?\*\*

- Streams are objects that allow you to read data from a source or write data to a destination in a continuous manner. They’re ideal for handling large volumes of data.

6. \*\*Describe middleware in Express.js.\*\*

- Middleware functions in Express.js are functions that have access to the request object, response object, and the next middleware function. They're used for tasks like logging, parsing request bodies, and implementing authentication.

7. \*\*How can you create RESTful APIs in Node.js using Express?\*\*

- By defining routes that respond to HTTP requests and using HTTP verbs (GET, POST, PUT, DELETE) to handle CRUD operations.

8. \*\*How do you handle errors in Node.js?\*\*

- Using try-catch blocks, error events, and error-handling middleware in Express.js to manage runtime errors gracefully.

9. \*\*What is the purpose of package.json?\*\*

- A package.json file holds metadata relevant to a Node.js project and contains scripts, dependencies, and project configuration.

10. \*\*What is the difference between process.nextTick() and setImmediate()?\*\*

- `process.nextTick()` executes a callback function on the current operation's completion before the event loop continues, whereas `setImmediate()` schedules a callback to execute after the current poll phase.

These topics and questions highlight key concepts and skills for working with Node.js, providing a foundation for both learning and discussing this technology in technical interviews.