Top Node.js Interview Questions and Answers

Q: What is Node.js?

A: Node.js is an open-source, cross-platform runtime environment built on Chrome's V8 JavaScript engine. It allows JavaScript to be used for server-side scripting, enabling developers to build scalable and high-performance applications.

Q: What is the difference between Node.js and JavaScript?

A: JavaScript is a programming language primarily used for client-side scripting in browsers, while Node.js is a runtime environment that enables JavaScript to run on the server.

Q: What are the key features of Node.js?

A: - Non-blocking and event-driven I/O

- Built-in asynchronous programming support

- Single-threaded but highly scalable

- Large ecosystem via npm (Node Package Manager)

Q: Explain the event loop in Node.js.

A: The event loop is the mechanism that allows Node.js to perform non-blocking I/O operations. It continuously checks the call stack and event queue for tasks to execute, handling them asynchronously.

Q: What is the difference between setImmediate() and process.nextTick()?

A: - process.nextTick(): Executes a callback immediately after the current operation finishes, before the event loop continues.

- setImmediate(): Schedules a callback to be executed in the next iteration of the event loop.

Q: What are streams in Node.js?

A: Streams are objects that enable reading or writing data in chunks, improving performance for

large files. There are four types of streams:

- Readable: Read data (e.g., fs.createReadStream).

- Writable: Write data (e.g., fs.createWriteStream).

- Duplex: Both read and write (e.g., network sockets).

- Transform: Modify or transform data (e.g., zlib.createGzip).

Q: What is middleware in Node.js?

A: Middleware is a function in Express.js that processes requests and responses. It can:

- Execute code.

- Modify requests and responses.

- End the request-response cycle.

- Call the next middleware in the stack.

Q: How does Node.js handle concurrency?

A: Node.js uses a single-threaded event loop for non-blocking operations. For heavy I/O tasks or CPU-intensive operations, it offloads them to worker threads or the thread pool, using the libuv library for handling concurrency.

Q: What is clustering in Node.js?

A: Clustering allows you to create multiple child processes (workers) sharing the same server port to leverage multi-core CPUs and improve application performance. Example: using the cluster module.

Q: What is the purpose of buffer in Node.js?

A: The Buffer class is used to handle binary data directly in Node.js, especially when dealing with file systems, streams, and networking.

Q: Explain how to secure a Node.js application.

A: - Use HTTPS.

- Sanitize and validate user inputs.

- Implement authentication and authorization (e.g., JWT, OAuth).
- Prevent SQL and NoSQL injection.
- Use Helmet for setting HTTP headers.
- Avoid storing sensitive information in plain text.

Q: How does Node.js handle file uploads?

A: Node.js can handle file uploads using packages like multer or formidable, which parse incoming form-data and save the files to the server or storage.

Q: Write an example of a simple HTTP server in Node.js.

A: ```javascript

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

const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Welcome to the homepage!');
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Page not found');
  }
});

server.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Q: Explain the difference between synchronous and asynchronous programming in Node.js.

A: - Synchronous: Tasks are executed one after another, blocking further execution until the current task completes.

- Asynchronous: Tasks can run in parallel, allowing the next task to start without waiting for the previous one to complete.