Zensar Technologies

Few Node.js Interview Questions

Basic Questions

1. What is Node.js and what are its main features?
   - Explain that Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine, and its main features include asynchronous, event-driven architecture, non-blocking I/O, and single-threaded event loop.

2. What is the role of the event loop in Node.js?
   - Describe how the event loop handles asynchronous operations and manages callbacks, allowing Node.js to perform non-blocking I/O operations.

3. How does Node.js handle asynchronous operations?
   - Discuss the use of callbacks, Promises, and `async/await` for managing asynchronous code in Node.js.

4. What is npm and what is it used for?
   - Explain that npm (Node Package Manager) is used to manage packages and dependencies for Node.js projects.

5. How do you create a basic HTTP server in Node.js?
   - Provide a simple example of creating an HTTP server using the `http` module.

6. What are modules in Node.js and how are they used?
   - Describe how Node.js modules encapsulate functionality and how `require` or `import` is used to include them in other files.

7. What is the purpose of the `package.json` file?
   - Explain that `package.json` manages project metadata, dependencies, scripts, and other configurations.

8. How do you handle errors in Node.js?
   - Discuss techniques for error handling including try-catch blocks, error-first callbacks, and error events.

Intermediate Questions

1. What are streams in Node.js and how do they work?
   - Describe how streams handle reading and writing data efficiently by processing it in chunks.

2. What is the difference between `process.nextTick()` and `setImmediate()`?
   - Explain how `process.nextTick()` queues a callback to be executed after the current operation completes, while `setImmediate()` queues a callback to be executed on the next iteration of the event loop.

3. How do you use the `fs` module for file operations in Node.js?
   - Describe methods for reading, writing, and manipulating files using the `fs` module.

4. What is middleware in the context of Express.js?

- Explain that middleware functions process requests before they reach the route handlers, and can be used for tasks like authentication, logging, and error handling.

5. How does Node.js manage concurrency?
- Discuss how Node.js uses its event-driven model and non-blocking I/O to handle multiple operations concurrently.

6. What are the differences between `require()` and `import`?
- Explain the differences in module systems, where `require()` is used in CommonJS and `import` in ES Modules.

7. How do you manage environment variables in Node.js?
- Describe the use of `.env` files and libraries like `dotenv` for managing environment variables.

8. What is a callback hell and how can it be avoided?
- Explain the problem of deeply nested callbacks and how to avoid it using Promises, `async/await`, or modularization.

Advanced Questions

1. What is clustering in Node.js and how does it improve performance?
- Describe how clustering allows for running multiple instances of Node.js on different CPU cores to handle more load and improve performance.

2. How do you implement authentication and authorization in a Node.js application?
- Discuss using libraries like `passport` for authentication and strategies for managing user roles and permissions.

3. What are the benefits and challenges of using microservices with Node.js?
- Explain the advantages of modularization, scalability, and the challenges of inter-service communication, data consistency, and deployment.

4. How do you handle memory leaks in Node.js applications?
- Describe techniques for identifying and addressing memory leaks, such as using tools like `heapdump` and `node-inspect`.

5. What are the key differences between `child_process` and `worker_threads` modules?
- Explain how `child_process` is used for spawning child processes and `worker_threads` is used for multi-threading within a single process.

6. How does Node.js interact with databases?
- Discuss different database interaction methods, including using drivers and ORMs (e.g., Mongoose for MongoDB, Sequelize for SQL databases).

7. What are the key security considerations when building Node.js applications?
- Describe practices such as input validation, securing dependencies, preventing code injection, and using HTTPS.

8. How do you implement caching in Node.js applications?
- Discuss strategies for caching, such as using in-memory caches (e.g., `memory-cache`), external caches (e.g., Redis), and HTTP caching mechanisms.

Scenario-Based Questions

1. How would you design a RESTful API using Express.js?
   - Describe the steps to set up routes, handle HTTP methods, and structure controllers and middleware.

2. Explain how you would handle file uploads in a Node.js application.
   - Discuss using middleware like `multer` for handling multipart form data and storing files.

3. How would you optimize a Node.js application for high performance?
   - Describe techniques such as profiling, load balancing, caching, and using asynchronous patterns effectively.

4. How would you troubleshoot a Node.js application that is running slowly?
   - Explain methods for diagnosing performance issues, including profiling, monitoring, and identifying bottlenecks.