

Topics and Subtopics from the Repository

1. Introduction to Node.js

- What is Node.js?
- Installing Node.js
- Node.js REPL

2. Node.js Basics

- Modules (require and exports)
- Global Objects
- Process Object

3. NPM (Node Package Manager)

- Installing Packages
- Package.json
- NPM Scripts

4. File System Module

- Reading Files
- Writing Files
- Directories and Paths

5. HTTP Module

- Creating a Simple Server
- Handling Requests and Responses
- Routing Basics

6. Express.js Framework

- Installing Express
- Basic Routing
- Middleware

7. Advanced Express

- Static Files
- Error Handling
- Template Engines (EJS/Pug)

8. Databases with Node.js

- MongoDB Setup
- Mongoose ODM
- CRUD Operations

9. Authentication

- User Registration
- Login and Sessions
- JWT Tokens

10. REST APIs

- Building RESTful Endpoints
- API Testing
- CORS Handling

11. Real-Time Applications

- WebSockets with Socket.io
- Chat Applications
- Event Handling

12. Testing and Debugging

- Unit Testing with Mocha
- Assertions with Chai
- Debugging Tools

13. Deployment and Production

- Environment Variables
- Hosting on Heroku/AWS
- Process Management (PM2)

Suggestions for Coding Practicals

To build practical knowledge, focus on hands-on coding exercises tied to each topic. Start with basics and progress to projects. Use the repo's examples as a base, then extend them. Here's a list of suggested practicals (one or more per topic) with brief descriptions. Aim to code each in a dedicated folder, test thoroughly, and version control with Git.

1. **Introduction to Node.js:** Create a "Hello World" script that logs system info using `process` and runs in REPL.
2. **Node.js Basics:** Build a custom module that exports functions for math operations (e.g., add, multiply) and import it in another file.
3. **NPM:** Initialize a project with `npm init`, install a package like `lodash`, and write a script that uses it for array manipulation.
4. **File System Module:** Write a program that reads a JSON file, modifies data, and writes it back to a new file.
5. **HTTP Module:** Create a basic HTTP server that serves static HTML pages and handles GET/POST requests for a simple form.

6. **Express.js Framework:** Build a mini-blog app with routes for home, about, and contact pages using Express.
 7. **Advanced Express:** Add middleware for logging requests and use EJS to render dynamic templates for a user profile page.
 8. **Databases with Node.js:** Set up a MongoDB connection, create a schema for "users," and perform CRUD (create, read, update, delete) operations via a Node.js script.
 9. **Authentication:** Implement user registration and login with bcrypt for password hashing and sessions for persistence.
 10. **REST APIs:** Develop a REST API for a "tasks" app with endpoints for listing, adding, updating, and deleting tasks, using Postman for testing.
 11. **Real-Time Applications:** Build a simple chat app using Socket.io where users can send messages in real-time.
 12. **Testing and Debugging:** Write unit tests for a calculator module using Mocha and Chai, including edge cases.
 13. **Deployment and Production:** Deploy your Express app to Heroku, configure environment variables, and set up PM2 for process management.
-