# Mastering Nodejs

### 1. Introduction to Node.js

* **1.1 What is Node.js?**
* **1.2 History and Evolution**
* **1.3 Node.js Use Cases**
* **1.4 Advantages and Disadvantages**

### 2. Setting Up the Development Environment

* **2.1 Installing Node.js and npm**
* **2.2 Using Node Version Manager (nvm)**
* **2.3 Setting Up a Code Editor (VSCode, WebStorm)**
* **2.4 Command Line Basics**

### 3. Node.js Fundamentals

* **3.1 Understanding the V8 Engine**
* **3.2 JavaScript Essentials for Node.js**
  + Variables, Data Types, and Operators
  + Control Flow and Looping
  + Functions and Scope
  + ES6+ Features (Arrow Functions, Destructuring, etc.)
* **3.3 The Node.js Event Loop**
* **3.4 Global Objects and Globals**
* **3.5 The Node.js Process Object**

### 4. Modules and Packages

* **4.1 The Module System**
  + CommonJS Modules
  + ES6 Modules (import/export)
* **4.2 Creating and Using Modules**
* **4.3 The require() Function**
* **4.4 Using npm Packages**
* **4.5 Publishing Packages to npm**
* **4.6 Package Management**
  + package.json and package-lock.json
  + Semantic Versioning

### 5. File System Operations

* **5.1 Reading and Writing Files**
* **5.2 Working with Directories**
* **5.3 Watching for File Changes**
* **5.4 Streams and Buffers**
* **5.5 Path and FS Modules**

### 6. Asynchronous Programming

* **6.1 Understanding Asynchronous JavaScript**
* **6.2 Callbacks**
* **6.3 Promises**
* **6.4 Async/Await**
* **6.5 Event Emitters**
* **6.6 Error Handling in Async Code**

### 7. Networking and HTTP

* **7.1 Building a Simple HTTP Server**
* **7.2 Handling Requests and Responses**
* **7.3 Working with HTTPS**
* **7.4 Using HTTP Clients (http, https modules)**
* **7.5 Making HTTP Requests**

### 8. Express.js Framework

* **8.1 Introduction to Express.js**
* **8.2 Setting Up an Express Application**
* **8.3 Middleware Functions**
* **8.4 Routing**
* **8.5 Handling Forms and Input**
* **8.6 Serving Static Files**
* **8.7 Templating Engines (EJS, Pug, Handlebars)**
* **8.8 Error Handling in Express**

### 9. Working with Databases

* **9.1 Overview of Databases (SQL vs NoSQL)**
* **9.2 Using MongoDB with Mongoose**
  + Connecting to MongoDB
  + Defining Schemas and Models
  + CRUD Operations
* **9.3 Using PostgreSQL with Sequelize**
  + Setting Up Sequelize
  + Defining Models and Associations
  + Querying Data
* **9.4 Database Connections and Pools**
* **9.5 Transactions**
* **9.6 Indexing and Performance**

### 10. Authentication and Authorization

* **10.1 Understanding Authentication vs Authorization**
* **10.2 Implementing JWT (JSON Web Tokens)**
* **10.3 OAuth and Social Logins**
* **10.4 Sessions and Cookies**
* **10.5 Password Hashing and Security Best Practices**

### 11. RESTful API Development

* **11.1 Principles of REST**
* **11.2 Designing RESTful APIs**
* **11.3 Versioning APIs**
* **11.4 Handling Pagination, Sorting, and Filtering**
* **11.5 API Documentation with Swagger/OpenAPI**
* **11.6 Testing APIs with Postman**

### 12. Testing and Debugging

* **12.1 Importance of Testing**
* **12.2 Unit Testing with Mocha, Chai, Jest**
* **12.3 Integration Testing**
* **12.4 Test-Driven Development (TDD)**
* **12.5 Debugging Techniques**
* **12.6 Using the Node.js Inspector**
* **12.7 Logging (Winston, Morgan)**

### 13. Real-time Applications

* **13.1 Introduction to WebSockets**
* **13.2 Using <Socket.io>**
* **13.3 Building a Real-time Chat Application**
* **13.4 Broadcasting and Rooms**
* **13.5 Event Handling**

### 14. Performance and Optimization

* **14.1 Profiling Node.js Applications**
* **14.2 Memory Management and Garbage Collection**
* **14.3 Event Loop Monitoring**
* **14.4 Clustering and Worker Threads**
* **14.5 Caching Strategies (Redis, Memory Cache)**
* **14.6 Best Practices for Performance**

### 15. Security in Node.js

* **15.1 Common Security Threats (XSS, CSRF, SQL Injection)**
* **15.2 Secure Coding Practices**
* **15.3 Data Validation and Sanitization**
* **15.4 Using Helmet.js**
* **15.5 SSL/TLS Certificates**
* **15.6 Rate Limiting**
* **15.7 Vulnerability Scanning with npm audit**

### 16. Deployment and Scaling

* **16.1 Preparing for Production**
* **16.2 Environment Variables and Configurations**
* **16.3 Using Process Managers (PM2, Forever)**
* **16.4 Containerization with Docker**
* **16.5 Continuous Integration/Continuous Deployment (CI/CD)**
* **16.6 Deploying to Cloud Platforms (AWS, Heroku, DigitalOcean)**
* **16.7 Load Balancing**
* **16.8 Horizontal vs. Vertical Scaling**

### 17. Microservices and Serverless Architecture

* **17.1 Introduction to Microservices**
* **17.2 Breaking Monolith into Microservices**
* **17.3 Communication Between Services (REST, gRPC, Message Queues)**
* **17.4 Serverless Concepts**
* **17.5 Building Serverless Functions with AWS Lambda, Azure Functions**
* **17.6 Advantages and Challenges**

### 18. GraphQL with Node.js

* **18.1 Introduction to GraphQL**
* **18.2 Setting Up a GraphQL Server**
* **18.3 Defining Schemas and Resolvers**
* **18.4 Querying and Mutations**
* **18.5 Integrating with Databases**
* **18.6 Using Apollo Server**

### 19. Advanced Topics

* **19.1 Child Processes and Worker Threads**
* **19.2 Native Add-ons with Node.js**
* **19.3 Streamlining Build Processes (Webpack, Babel)**
* **19.4 Event-Driven Architecture**
* **19.5 Internationalization and Localization**
* **19.6 Message Queues (RabbitMQ, Kafka)**

### 20. Best Practices and Design Patterns

* **20.1 Coding Standards and Style Guides**
* **20.2 Design Patterns in Node.js**
* **20.3 Error Handling Strategies**
* **20.4 Refactoring Techniques**
* **20.5 Code Reviews and Pair Programming**

### 21. Contributing to the Node.js Community

* **21.1 Open Source Contribution**
* **21.2 Understanding Node.js Internals**
* **21.3 Node.js Release and LTS Cycles**
* **21.4 Participating in Node.js Working Groups**

### 22. Project: Building a Full-stack Application

* **22.1 Project Planning and Requirements Gathering**
* **22.2 Setting Up the Project Structure**
* **22.3 Implementing Backend APIs**
* **22.4 Integrating with Frontend Frameworks (React, Angular, Vue)**
* **22.5 Testing and Deployment**
* **22.6 Continuous Improvement and Maintenance**

### 23. Resources for Continuous Learning

* **23.1 Recommended Books and Documentation**
* **23.2 Online Courses and Tutorials**
* **23.3 Community Forums and Meetups**
* **23.4 Staying Up-to-date with Node.js Releases**

————————

This comprehensive table of contents is designed to guide you from a beginner to a hero in Node.js, covering all essential topics and advanced concepts up to October 2023.

#software/tools/webdev/nodejs