# Guide2Code - Node.js Roadmap 🌍

## 🟢 Phase 1: Beginner Level

### 📚 Topics to Learn:

1. Introduction to Node.js (What is Node.js? Why use it?)
2. Installing Node.js & npm (Setting up the environment)
3. Understanding the Event Loop (Asynchronous programming)
4. Core Modules in Node.js (fs, path, os, events, http)
5. Handling Files (Reading, Writing, Deleting files)
6. Creating a Basic HTTP Server (Using the http module)
7. Working with npm (Installing and managing packages)
8. Basic Express.js (Setting up a simple server)
9. Handling Requests & Responses (GET, POST, PUT, DELETE)
10. Error Handling & Debugging (try-catch, logging errors)

### 🛠 Beginner Project Ideas:

- **Simple Web Server** – Serve HTML pages using Node.js
- **JSON File Manager** – Read and update JSON files dynamically
- **Basic API with Express** – Create a simple REST API
- **Weather CLI App** – Fetch weather data using an API
- **File Uploader** – Upload and manage files using Node.js

## 🟡 Phase 2: Intermediate Level

### 📚 Topics to Learn:

1. Advanced Express.js (Middleware, Routing, CORS, Body Parser)
2. Database Integration (MongoDB, MySQL, PostgreSQL)
3. Authentication & Authorization (JWT, OAuth, bcrypt)
4. Working with APIs (RESTful APIs, GraphQL Basics)
5. WebSockets & Real-time Communication (Socket.io)
6. File Handling (Multer for file uploads)
7. Caching & Performance Optimization (Redis, Compression)

8. Testing in Node.js (Jest, Mocha, Chai)
9. Background Jobs & Task Scheduling (node-cron, BullMQ)
10. Deploying Node.js Apps (Heroku, Vercel, DigitalOcean)

### 🛠️ Intermediate Project Ideas:

- **User Authentication System** – Implement login and signup with JWT
- **Chat Application** – Real-time messaging with WebSockets
- **URL Shortener** – Create a TinyURL-like service
- **E-commerce API** – Build a backend for an online store
- **Task Scheduler** – Automate tasks with cron jobs

## 🔴 Phase 3: Advanced Level

### 📚 Topics to Learn:

1. Microservices Architecture (Building scalable apps with Node.js)
2. Advanced Database Management (Indexes, Transactions, Replication)
3. Kubernetes & Docker (Containerizing Node.js apps)
4. Serverless Architecture (AWS Lambda, Google Cloud Functions)
5. Advanced GraphQL (Subscriptions, Resolvers, Apollo Federation)
6. Message Queues (RabbitMQ, Kafka, BullMQ)
7. Load Balancing & Scaling (NGINX, PM2, Clustering)
8. Security Best Practices (Rate Limiting, Data Encryption, Helmet.js)
9. CI/CD Pipelines (GitHub Actions, Jenkins, Travis CI)
10. Building a Full-Stack MERN App (MongoDB, Express, React, Node.js)

### 🛠️ Advanced Project Ideas:

- **Microservices-based App** – Build a modular application with Node.js
- **AI-Powered Chatbot** – Integrate OpenAI API for automated responses
- **Stock Market Tracker** – Real-time stock price updates using WebSockets
- **SaaS Application Backend** – Scalable API for multi-tenant applications
- **Streaming Platform Backend** – Handle video uploads and live streaming