Perfect 👍 Since you already have **5+ years in Laravel/PHP**, I'll structure these **Node.js notes** as if you're an **experienced developer transitioning to Node**, with a focus on **core fundamentals, advanced concepts, and interview-focused topics**.

Let's start with the **Index (Table of Contents)**.

---

1. Introduction to Node.js

- What is Node.js?
- Event-driven & non-blocking I/O
- Difference from traditional backends

---

2. Node.js Architecture

- Event loop (single-threaded)
- libuv & background threads
- Callbacks, Promises, Async/Await in event loop

---

3. Core Modules in Node.js

- `fs` (File System)
- `http` & `https`
- `path`, `os`, `events`
- `stream`, `buffer`

---

4. Package Management

- npm & yarn basics
- package.json & package-lock.json
- Semantic Versioning (semver)

---

5. Asynchronous Programming

- Callbacks & callback hell
- Promises
- Async/Await
- Error handling

---

6. Express.js Framework

- Introduction to Express
- Routing & Middleware
- Request & Response cycle

- Error handling in Express

---

7. Database Integration

- Connecting Node with MySQL & PostgreSQL
- MongoDB with Mongoose
- ORM/ODM vs Query Builders

---

8. RESTful APIs with Express

- REST principles
- CRUD with Express
- API versioning & documentation (Swagger/Postman)

---

9. Authentication & Security in Node.js

- JWT authentication
- OAuth & session-based auth
- Preventing vulnerabilities (XSS, CSRF, SQL injection)

---

10. Testing in Node.js

- Unit testing with Jest/Mocha
- Supertest for API testing
- Best practices in testing

---

11. Event-Driven Programming

- EventEmitter in Node
- Custom events
- Real-time apps with Socket.io

---

12. Streams in Node.js

- Types of streams (Readable, Writable, Duplex, Transform)
- `pipe()` method
- Use cases (file handling, network data, large payloads)

---

13. Node.js Clustering & Scaling

- Clustering (multi-core utilization)
- Worker Threads vs Clusters
- Load balancing

---

14. Worker Threads & Child Processes

- Worker Threads API
- Child processes (`spawn`, `exec`, `fork`)
- When to use each

---

15. Node.js Security Best Practice

- Secure headers with `helmet`
- Input validation & sanitization
- Avoid eval & unsafe modules
- Rate limiting & brute-force protection
- Environment variable management

---

16. Performance Optimization in Node.js

- Optimize async code
- Caching (Redis, in-memory)
- Clustering & load balancing
- Streams for large data
- Gzip & compression
- DB query optimization
- Monitoring & profiling

---

17. Microservices & Message Queues

- Microservices architecture in Node.js
- RabbitMQ basics
- Kafka basics
- Redis Pub/Sub

---

18. GraphQL with Node.js

- Intro to GraphQL
- Apollo Server setup
- Queries, Mutations & Subscriptions

---

19. Serverless with Node.js

- Why Serverless? (pros & cons)
- AWS Lambda basics
- Google Cloud Functions
- Azure Functions

- Serverless frameworks (Serverless Framework, SAM)
- Use cases (APIs, event-driven, cron jobs)

---

20. Error Handling & Debugging ⚡ (missing earlier)

- try/catch vs async error handling
- Centralized error handling in Express
- Logging with Winston/Morgan
- Debugging with Node Inspector

---

21. Deployment & DevOps ⚡ (missing earlier)

- PM2 for process management
- Dockerizing Node.js apps
- Nginx as reverse proxy
- CI/CD pipelines for Node.js projects

---

22. Interview-Focused Questions

- Process vs Thread in Node
- How does the event loop work?
- Streams in Node.js
- Scaling Node.js apps
- JWT vs Sessions
- Callback hell & solutions
- Common performance bottlenecks
- Clustering vs Worker Threads
- Security practices
- Difference between Serverless & Containers

---

23. One-Page Summary

- Visual mind-map + quick bullet-point review
- Event loop diagram
- Core patterns & best practices
- Interview shortcuts

☑ Now this is your **final cleaned-up index** with **subtopics** included.
Do you want me to now continue with the **missing Topic 20 (Error Handling & Debugging)** in the same detailed format as the others?