Alright 🚀 let's dive into the **first topic**. Since you already have backend experience (Laravel), I'll keep explanations **developer-oriented**, highlighting **why Node.js is different** and **what matters in interviews**.

---

1. Introduction to Node.js

📌 What is Node.js?

- **Node.js** is a **JavaScript runtime environment** built on **Google Chrome's V8 engine**.
- It allows you to run JavaScript **outside the browser** → mainly on the **server-side**.
- Key feature: **Non-blocking, event-driven architecture**, making it lightweight and efficient for **I/O-heavy applications** (APIs, chat apps, streaming services).

📌 Why use Node.js?

- **Single Language Full-Stack**: Same language (**JavaScript**) for frontend & backend.
- **Asynchronous by default**: Handles **thousands of concurrent connections** efficiently without blocking.
- **Huge ecosystem**: npm (Node Package Manager) has **1M+ packages**.
- **Real-time support**: Excellent for real-time apps (chat, live streaming).

📌 Node.js vs Traditional Backend (e.g., Laravel, Django, Spring)

Feature	Node.js	Laravel (PHP)
Concurrency Model	Single-threaded, non-blocking I/O	Multi-threaded, blocking (unless async added)
Performance	High for I/O-heavy tasks	Good for CPU+DB-heavy, but can block
Language	JavaScript everywhere	PHP backend only
Scalability	Easy horizontal scaling (clusters, microservices)	Scaling requires more infra & threads
Community/Ecosystem	npm, huge JS ecosystem	Composer ecosystem

👉 **Interview Angle**:

- If asked **why Node.js over PHP/Laravel?** → highlight **event-driven architecture, single-threaded performance for I/O, scalability, and same language (JS) across stack**.

📌 Event-Driven & Non-Blocking I/O

- Traditional servers (like Laravel/PHP-FPM) → **block per request** → need **multiple threads** for concurrency.
- Node.js → **event loop** handles requests asynchronously in a **single thread**.
- **Non-blocking I/O** = while Node waits for DB/FS/Network, it continues processing other requests.

Example (Pseudo-code):

```
// Blocking (Laravel/PHP style)
$data = db.query("SELECT * FROM users");
echo $data;

// Non-blocking (Node.js style)
db.query("SELECT * FROM users", (err, data) => {
  if (err) throw err;
  console.log(data);
});
console.log("Query sent"); // Executes without waiting
```

📌 Where Node.js Shines

☑ Real-time apps (chat, notifications, gaming)
☑ APIs handling many concurrent requests
☑ Streaming apps (Netflix, Spotify)
☑ Microservices

❌ Not best for CPU-intensive apps (e.g., video encoding, AI training) → because **event loop gets blocked**.

📌 Key Takeaways

- Node.js = **JavaScript on server** using V8.
- Runs on **event-driven, non-blocking I/O model** → handles concurrency well.
- Perfect for **APIs, real-time apps, microservices**.
- Different from Laravel/PHP → instead of threads, Node uses **event loop**.

👉 Next up would be **2. Node.js Architecture** (event loop, libuv, background threads, async/await).

Do you want me to explain **Node.js Architecture** in a **deep-dive with diagrams and interview Q&A**, or keep it **short bullet-style notes**?