Perfect 👍 let's cover the **10th topic: Testing in Node.js**. Since you're already an experienced Laravel developer, I'll highlight **similarities/differences with PHPUnit/Pest**, and add **interview-relevant points**.

---

# 10. Testing in Node.js

---

## 📌 Why Testing in Node.js?

- Testing ensures **stability, reliability, and maintainability** of your application.
- In interviews, companies want to know:
  - Which testing frameworks you use?
  - How do you test APIs in Node.js?
  - Do you use **unit tests, integration tests, and end-to-end (E2E) tests**?

---

## 📌 Types of Tests in Node.js

1. **Unit Testing** – Testing small isolated functions/modules.
   *Example: testing a utility function that hashes passwords.*
2. **Integration Testing** – Testing how different modules work together.
   *Example: API route calling a DB and returning JSON.*
3. **End-to-End (E2E) Testing** – Simulating real-world usage.
   *Example: Testing a login API → sending request with email & password → checking JWT response.*

---

## 📌 Popular Testing Frameworks

- **Mocha** → Flexible, widely used.
- **Jest** → Zero-config, comes with assertions, spies, coverage.
- **Chai** → Assertion library (used with Mocha).
- **Supertest** → For testing HTTP endpoints (Express APIs).
- **Sinon.js** → For mocking, spying, stubbing functions.

👉 For interviews:

- Be ready to explain **Jest + Supertest combo** (most common for REST API testing).

---

## 📌 Example: Unit Test with Jest

```
// math.js
function add(a, b) {
  return a + b;
```

```
}
module.exports = add;

// math.test.js
const add = require('./math');

test('adds two numbers', () => {
  expect(add(2, 3)).toBe(5);
});
```

👉 Run: `npx jest`

---

## 📌 Example: API Testing with Supertest + Jest

```
// app.js
const express = require('express');
const app = express();
app.get('/hello', (req, res) => {
  res.status(200).json({ message: 'Hello World' });
});
module.exports = app;

// app.test.js
const request = require('supertest');
const app = require('./app');

describe('GET /hello', () => {
  it('should return Hello World', async () => {
    const res = await request(app).get('/hello');
    expect(res.statusCode).toBe(200);
    expect(res.body.message).toBe('Hello World');
  });
});
```

👉 Run: `npx jest`

---

## 📌 Coverage Reports

- Jest can generate code coverage reports easily:

  ```
  npx jest --coverage
  ```

- Shows % of statements, branches, functions, and lines tested.

- Interviewers may ask: **"How do you ensure test coverage?"** → Mention using `jest --coverage` or similar tools.

---

## 📌 Best Practices

- Keep **unit tests isolated** (don't hit DB, mock external services).
- Use **integration tests** for API + DB flow.
- Use **E2E tests** for user journeys (login, checkout, etc.).
- Automate tests in **CI/CD pipelines** (GitHub Actions, GitLab, Jenkins).

---

## 📌 Interview Questions

1. **Which testing frameworks have you used in Node.js?**
   → Expected: Jest, Mocha, Supertest.

2. **How do you test APIs in Node.js?**
   → Use Supertest with Jest/Mocha to simulate HTTP requests.

3. **Difference between unit, integration, and E2E testing?**

4. **How do you mock database calls in tests?**
   → Use libraries like `sinon`, or Jest's `jest.fn()` mocks.

5. **How do you measure test coverage?**
   → With Jest/Mocha + Istanbul (built-in with Jest).

---

☑ **Key Takeaway**:

- Use **Jest (unit/integration) + Supertest (API)**.
- Focus on **coverage, mocking, and CI/CD automation**.
- Interviewers love candidates who talk about **test pyramid (unit > integration > E2E)**.

---

👉 Next up would be **11. Event-Driven Programming (Events & EventEmitter in Node.js)**.

Do you want me to **continue sequentially** (like we're doing), or do you want me to **prioritize interview-heavy topics first** (e.g., clustering, scaling, performance, async vs sync pitfalls)?