🧪 What is **Jest**?

Jest is a **JavaScript testing framework** used for **unit testing**, **integration testing**, and **API testing**.

- Created by **Facebook**
- Works with **Node.js**, **React**, **TypeScript**, etc.
- Zero configuration — easy to use

🌐 What is Supertest?

Supertest is a library for **testing HTTP endpoints** (like `/signup`, `/login`, etc.)
It works perfectly with **Express** and **Jest**.

🧰 Installation

Run this in your Node.js project:

npm install --save-dev jest supertest

Then, add a Jest script in `package.json`:

```
"scripts": {
  "test": "jest"
}
```

🧪 3. Jest Test File Structure

✅ Test files can be:

- `filename.test.js`
- `filename.spec.js`

✅ Default location:

- Anywhere in the project (Jest automatically detects `__tests__` folders)
-

```
css

src/
  app.js
  routes/
  tests/
    signup.test.js
```

🧩 Example: Testing `/signup` functionality

app.js

```
import express from 'express';
const app = express();

app.use(express.json());

app.post('/signup', (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).json({ message: 'Email and password are required' });
  }

  // Fake DB check
  if (email === 'test@example.com') {
    return res.status(409).json({ message: 'User already exists' });
  }

  return res.status(201).json({ message: 'User registered successfully' });
});

export default app;
```

🧪 signup.test.js

```
import request from 'supertest';
import app from './app.js';

describe('Signup API', () => {

  test('should return 400 if email or password missing', async () => {
    const res = await request(app)
      .post('/signup')
      .send({ email: '' }); // missing password

    expect(res.statusCode).toBe(400);
    expect(res.body.message).toBe('Email and password are required');
  });

  test('should return 409 if user already exists', async () => {
    const res = await request(app)
      .post('/signup')
      .send({ email: 'test@example.com', password: '123456' });

    expect(res.statusCode).toBe(409);
    expect(res.body.message).toBe('User already exists');
  });

  test('should return 201 if signup successful', async () => {
    const res = await request(app)
      .post('/signup')
      .send({ email: 'newuser@example.com', password: 'mypassword' });

    expect(res.statusCode).toBe(201);
    expect(res.body.message).toBe('User registered successfully');
  });

});
```

🧠 How it Works

- `request(app)` → Supertest uses your Express app directly.
- `.post('/signup')` → Sends a POST request to the `/signup` route.
- `.send({...})` → Sends a JSON body.
- `expect()` → Jest assertion to validate the response.
- `test()` or `it()` → defines a single test

▶️ Run Tests

You'll see output like:

```
PASS  ./signup.test.js
 Signup API
   ✓ should return 400 if email or password missing
   ✓ should return 409 if user already exists
   ✓ should return 201 if signup successful
```

⚖️ Common Jest Matchers

⚖️ 5. Common Jest Matchers

Matcher	Description	Example
`toBe()`	Exact equality	`expect(2 + 2).toBe(4)`
`toEqual()`	Deep equality for objects/arrays	`expect({a:1}).toEqual({a:1})`
`toBeTruthy()`	Checks if value is truthy	`expect(true).toBeTruthy()`
`toBeFalsy()`	Checks if value is falsy	`expect(0).toBeFalsy()`
`toContain()`	Array or string contains item	`expect(['a','b']).toContain('a')`
`toHaveLength()`	Checks array/string length	`expect([1,2,3]).toHaveLength(3)`
`toMatch()`	Regex match in string	`expect("hello").toMatch(/ell/)`
`toBeGreaterThan()`	Numeric comparison	`expect(10).toBeGreaterThan(5)`
`toBeCloseTo()`	For floating points	`expect(0.1 + 0.2).toBeCloseTo(0.3)`

🧩 Code Coverage Example

npm test -- --coverage

🧩 Mocking in Jest

Used to **simulate dependencies**, **external APIs**, or **DB calls**.

Example 1 — Mock function

```js
const sendEmail = jest.fn();
sendEmail("hello@example.com");

expect(sendEmail).toHaveBeenCalled();
expect(sendEmail).toHaveBeenCalledWith("hello@example.com");
```

Example 2 — Mocking a module

```js
jest.mock('../services/emailService.js', () => ({
  sendEmail: jest.fn(() => true),
}));
```