Testing (Jest, Mocha, Jasmine)

Here's a detailed explanation of **JavaScript Testing** using **Jest, Mocha, and Jasmine**:

✅ Why Testing Is Important

Testing ensures your code:

- Works as expected
- Remains bug-free after changes (regressions)
- Is easier to maintain and refactor

Types of tests:

- **Unit tests**: Test individual functions/components
- **Integration tests**: Test interactions between modules
- **End-to-end (E2E) tests**: Test the entire app workflow (e.g., using Cypress)

🧪 Common JavaScript Testing Libraries

Library	Type	Use Case
Jest	Test runner + assertion	Best for React and full JS apps
Mocha	Test runner	Flexible, often paired with Chai
Jasmine	Test runner + assertion	Simple and standalone testing framework

---

🔶 Jest

Jest is a full-featured testing framework by **Facebook**.

✅ Features:

- Zero config
- Built-in assertions

- Code coverage
- Snapshot testing
- Mocks and spies

📦 Install:

```
npm install --save-dev jest
```

🧪 Example:

```
// sum.js
function sum(a, b) {
  return a + b;
}
module.exports = sum;

// sum.test.js
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

Run tests:

```
npx jest
```

🌟 Jest is ideal for:

- **React apps**
- **Fast unit testing**
- Snapshot testing (e.g., React component output)

🟤 Mocha (with Chai)

Mocha is a lightweight JavaScript **test runner**, often used with **Chai** for assertions.

📦 Install:

```
npm install --save-dev mocha chai
```

🧪 Example:

```
// sum.js
function sum(a, b) {
  return a + b;
}
module.exports = sum;

// test/sum.test.js
const sum = require('../sum');
const { expect } = require('chai');

describe('sum()', () => {
  it('should return 3 when 1 + 2', () => {
    expect(sum(1, 2)).to.equal(3);
  });
});
```

Run:

```
npx mocha
```

🧠 Mocha offers:

- More **flexibility** than Jest
- Customizable setup
- Better for **Node.js applications**

You can add:

- `chai` (assertions)
- `sinon` (spies/mocks)
- `nyc` (code coverage)

🟡 Jasmine

Jasmine is a **BDD** (Behavior Driven Development) testing framework. It includes a test runner and assertion library.

📦 Install:

```
npm install --save-dev jasmine
npx jasmine init
```

🧪 Example:

```
// sum.js
function sum(a, b) {
  return a + b;
}
module.exports = sum;

// sum.spec.js
const sum = require('./sum');

describe("sum", () => {
  it("adds 1 and 2 to be 3", () => {
    expect(sum(1, 2)).toBe(3);
  });
});
```

Run:

```
npx jasmine
```

🧠 Jasmine is:

- Used by Angular CLI
- Lightweight and no external dependencies

✅ Comparison Table

Feature	Jest	Mocha + Chai	Jasmine
Test Runner	✅	✅	✅
Assertion Library	✅ (built-in)	❌ (needs Chai)	✅ (built-in)
Snapshot Testing	✅	❌	❌
Spies & Mocks	✅ (built-in)	❌ (use Sinon)	✅ (built-in)
Configuration	Zero-config	Customizable	Simple setup
Ideal For	React/JS	Node.js	Angular

🧪 Bonus: Test Terminology

- `describe()` — Groups test cases
- `it()` or `test()` — Defines individual tests
- `expect()` — Defines assertions
- `beforeEach` / `afterEach` — Setup/teardown hooks

🧠 Summary

Tool	Best For	Strengths
Jest	React, JS apps	Easy setup, fast, snapshot tests
Mocha	Node.js, backend services	Flexible, plugin-rich
Jasmine	Angular, simple projects	BDD style, standalone

Let me know if you want:

- Jest setup in a project

- Testing React components
- Coverage reports
- Mocking APIs in tests