## 🔰 React Testing with Jest — Phase 1: Foundation (Absolute Basics)

### ✅ Objective

To build a solid understanding of the **fundamentals of testing in React** using **Jest** and **React Testing Library (RTL)**. This phase focuses on the absolute basics, enabling you to test React components confidently.

### 📘 1. What is Testing and Why Do We Test?

#### ✅ Definition:

Testing is the process of writing code to automatically verify that your application behaves as expected.

#### 🎯 Why Do We Test?

* To **ensure correctness** of components and logic.
* To **catch bugs early** in development.
* To **prevent regressions** when refactoring code.
* To help write **better, more modular code**.
* To **automate verification** instead of manual testing.

#### ✅ Analogy:

Think of testing like a safety net for a trapeze artist — it prevents catastrophic falls (bugs) when you make changes.

### 📘 2. Types of Testing

| **Type** | **Scope** | **Examples** | **Tooling** |
| --- | --- | --- | --- |
| **Unit** | Tests a single function/component | Button, utils, pure functions | Jest |
| **Integration** | Tests multiple units together | Form with input + submit | Jest + React Testing Library |
| **E2E** | Tests user flow end-to-end | Login flow, cart to checkout | Cypress, Playwright |

### 📘 3. Setting Up React Testing Library and Jest

#### ✅ Install Dependencies

For most modern React apps (especially if using Vite, CRA, or Next.js), testing support is built-in or easy to add.

npm install --save-dev jest @testing-library/react @testing-library/jest-dom @testing-library/user-event

#### ✅ Configure Jest (if not auto-configured)

In package.json:

"scripts": {

"test": "jest"

}

#### ✅ Optional: Create jest.config.js

module.exports = {

testEnvironment: 'jsdom',

setupFilesAfterEnv: ['@testing-library/jest-dom/extend-expect'],

};

### 📘 4. Your First Test Case: Button Click

#### 🔹 Component (Counter.jsx):

import { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p data-testid="count-display">Count: {count}</p>

<button onClick={() => setCount(count + 1)}>Increment</button>

</div>

);

}

export default Counter;

#### 🔹 Test File (Counter.test.js):

import { render, screen, fireEvent } from '@testing-library/react';

import Counter from './Counter';

test('increments count on click', () => {

render(<Counter />);

const button = screen.getByText('Increment');

fireEvent.click(button);

expect(screen.getByTestId('count-display')).toHaveTextContent('Count: 1');

});

#### ✅ Breakdown:

* render(<Component />) renders the component in a virtual DOM.
* screen.getByText() finds the button.
* fireEvent.click() simulates a user click.
* expect(...).toHaveTextContent() asserts the result.

### 📘 5. Understanding render(), screen, and fireEvent

| **Method** | **Purpose** |
| --- | --- |
| render() | Renders a React component in virtual DOM |
| screen | Global helper to access queries (getBy, etc.) |
| fireEvent() | Simulates DOM events like click, change |

#### 🔹 Example:

render(<MyForm />);

const input = screen.getByPlaceholderText('Enter name');

fireEvent.change(input, { target: { value: 'Likan' } });

expect(input.value).toBe('Likan');

### 📘 6. Role of describe, test, it, expect

#### ✅ Syntax:

describe('Group of related tests', () => {

test('Individual test case', () => {

expect(true).toBe(true);

});

it('Another test case (alias of test)', () => {

expect(1 + 1).toBe(2);

});

});

| **Function** | **Purpose** |
| --- | --- |
| describe | Groups related tests |
| test | Defines a test case |
| it | Alias for test, can be used interchangeably |
| expect | Makes an assertion about some value |

#### 🔹 Chained Matchers with expect()

expect(button).toBeInTheDocument();

expect(input).toHaveAttribute('placeholder', 'Enter name');

### ✅ Summary of Phase 1

* ✔️ Learned **why** we test.
* ✔️ Understood **types** of testing.
* ✔️ Set up **Jest** and **React Testing Library**.
* ✔️ Wrote **first test case** (click button).
* ✔️ Explored **screen**, **render**, **fireEvent**.
* ✔️ Practiced test grouping and assertions.