JEST is a test runner that finds tests, runs the tests, determines whether the tests passes or failed & reports it back in human readable manner.

React Testing Library (RTL) 🡪 It is a testing utility that provides virtual DOM for testing React Components.

RTL provides a virtual DOM which we can use to interact with & verify the behaviour of react component. RTL is in fact a family of packages which helps testing UI components.

The CORE library is called “DOM Testing library” & RTL is simply a wrapper around the core library to test application in an easier way.

**TYPES OF TESTS**

1. Unit Tests
2. Integration Tests
3. End to End Tests

UNIT Tests focuses on testing the individual building blocks of an application such as class, function or a component.   
Each unit or building block is tested in isolation, independent of other units.  
Dependencies are mocked.

Integration Tests focuses on testing a combination of units & ensuring they work together.

End to End Tests focuses on testing the entire application flow & ensuring it works as designed from start to finish. It involves real UI, real backend DB & real services.

**Test Function**

Test (name, function, timeout)

* “name” is used to identify the test.
* “functions” contains the expectations to test.
* “timeout” specifies how long to wait before aborting the test. Default time is 5 sec.

**Explanation of default Test**

Import {render, screen} from ‘@testing-library/react’;  
Import ‘App’ from ‘./App’;  
Test(‘renders learn react link’, () => {

1. Render(<App/>);
2. Const linkElement = screen.getBYText(/learn react/i); //i means case sensitive.
3. Expect(linkElement.toBeInDocument());

});

1. Creates a virtual DOM of App Component.
2. Screen is imported from React Testing Library which contains the object to query virtual DOM.
3. Expect & Test are automatically imported from JEST.

Note:

We can also use “it” instead of “test”.  
Test.only 🡪 fit  
Test.skip 🡪xit

**WATCH MODE**

Watch Mode is an option that we can pass to JEST asking to watch files that have changed since the last commit and execute tests related only to those changed files.  
It is optimization designed to make your tests run fast regardless of how may tests you have.

test.only(“test name”, function) 🡪 will run only this test.  
test.skip(“test name”, function) 🡪 will skip this test.

**GROUPING TEST WITH JEST USING “describe”**

Syntax 🡪 describe(“name”, function)

1. “name” is group name.
2. “function” contains expectations to test.

describe(“component name”, () => {

test(“testName1”, () => {function})  
test(“testName2”, () => {function})

});

We can also use describe.only & describe.skip.  
We can nest a “describe” within a “describe”.  
We can have multiple describe in same file.  
1 test file is 1 suite.

**File Name Conventions**

Files with following suffix are allowed.

1. .test.js
2. .test.jsx
3. .spec.js
4. .spec.tsx
5. .js or .tsx files in \_\_test\_\_ folder.

**CODE COVERAGE**

It helps us to understand how much our S/W code is tested.

1. **Statement Coverage:** how many statements have been executed.
2. **Branch Coverage:** how many branches of control statements have been executed.
3. **Function Coverage:** how many functions defined have been called.
4. **Line Coverage:** how many lines of code have been tested.

Following Script has to be added in package.json.

"jest": {  
 "collectCoverage": true,  
 "collectCoverageFrom": ["src/\*\*/\*.js", "src/\*\*/\*.jsx"],  
 "coverageReporters": ["lcov", "text-summary"]  
 }

**ASSERTIONS**