13- Time for the Test (Developer Testing)

(Test case)

Different types of Testing a developer can do ⇒

① Manual Testing
- Just Testing whatever we develop. Just checking the function.

② write the test cases
- writing the code that can test our Application automatically

Even you write a single line of code, that can introduce a bug anywhere in the code. A small change can create a big impact, becoz they are inter-related.

3 Types of Testing

- most preferred
- developer are Responsib.

1) Unit Testing → (you test your React Component in isolation)
2) Integration Testing → (Testing the integration of components) (When multiple component work together)
3) End-to-End Testing - e2e testing → (Testing our Application as soon as user landing on our page till user exits from Application. Testing the whole flow of App)
   ↳ (Selenium) tool for testing

libraries used for testing:

① React Testing library → uses jest behind the scenes
built on top of
Dom Testing library

Javascript testing Framework

installation

① npm i -D @testing-library/react @testing-library/dom
② npm i -D jest
③ npm i babel-jest @babel/core @babel/present-env
④ create babel.config.js for babel configuration.

(③ dependency for jest)

```
module.exports = {
    presets: [["@babel/preset-env", { target: {
        node: "current" } }]],
};
```

⑤ Configure Parcel Config file to disable default babel transpilation

Parcel has bydefault babel Configuration, by for jest or eslint there is Separate another babel Configuration, inorder to avoid collision of both of these Configuration. you have make parcel.rc file for parcel configuration

(& now both will not conflict) and babel configuration can be used for jest

How to run testcases ⇒ "npm run test"

(--save-dev) = (-D)
Both are Same

⑥ jest Configuration

"npx jest --init"
↳ Create a jest.config.js

It will ask few questions ① Would you like to use Typescript ⇒ No
② Choose the test environment for testing
⇒ jsdom (browser-like)

Test Cases does not run on Browser or Server.
- It requires environment for running.
i.e, jsdom
↳ library which parses & interacts with assembled HTML (just like browser)

③ Do you want to add coverage report ⇒ Yes
④ Which provider should be used to instrument code for coverage? ⇒ babel
⑤ Automatically clear mock calls, instances, content and results before every test ⇒ Yes

⑦ Install jsdom library
" npm install -D jest-environment-jsdom"

Q) Where do it write test cases?
⇒ test cases files are present in the folder(--tests--)
- whatever files are present inside this folder jest will track them.
- Any file with the extension .js & .ts in the (--tests--) Folder will be considered for testing.
eg:- testing files :- _.test.js
_.test.ts
_.spec.js
_.Spec.ts

Two underscore in the start and Two underscore in the end are considered as dunder
• It is the name convection for testing (Reserved name)

Q) How to write test cases. ⇒ Write a fn known as & this fn takes 2 arg.
- String
- Callback fn

first arg :- description of test case.
Second arg :- Implementation of test case

```
test("", () => {
  const res = sum(3,4);
  expect(result).toBe(7);
})
```
↳ assertion

- you can write "it" instead of "test"
both are Same. both have same meaning
- you can write any either 'it' or 'test'.

Testing Cases for React [Unit Testing]

- Testing only one Component. Independent of your App.
- You are taking out one Component and you're testing.
- like Isolation. Testing in a Isolated env

- whenever you are testing a UI Component inside React
  - you will have to Render that Component on to the JS DOM
    - using `render(<componentName/>);` method.

```
import { render } from "@testing-library/react"
```

Eg:- We trying to test our Contact Us Component

```
import { render, screen } from "@testing-library/react";
import Contact from "../contact";

test("Should load contact us component", () => {
  render(<Contact/>);
  const heading = screen.getByRole("heading");
  expect(heading).toBeInTheDocument();
});
```

⇩

toBeInTheDocument comes from a library known as
npm i -D @testing-library/jest-dom

- We cannot use jsx inside our testcase.
- JSX isn't enabled for our test cases.
- To enable, you need to add.
  @babel/preset-react

⇩

It makes jsx work inside our test cases

After installing above library add its configuration in babel.config.js file.

```
module.exports = {
presets: [
  ["@babel/preset-react",
    { runtime : "automatic"}
  ],
],};
```

Everytime in your test you will

- Render ⟶ `render(<component/>)`
- Quering ⟶ `const btn = screen.getByRole("button")`
- Assertion ⟶ `expect(btn).toBeInTheDocument();`

- Sometimes test file increases, Suppose in your file there are 20 test cases, It becomes difficult to make such huge amount of test case.
- We can create small few grp of test cases. different grp of test cases

How you can create different grp's using,

```
describe("", () => {
  - test1("", () => {})
  - test2("", () => {})
  - test3("", () => {})
})
```

your describe block can have multiple test cases.

- jsdom understands react code, jsx code, js code, it does not know redux code.
- Our test case will throw error incase we try to render component with redux library

Hence, we need to provide the store using
<Provider store={appStore}></Provider>

Be careful to configure these before testing

- Also, we have to provide context of React-Router-dom library.
  By provider Router to Component i.e. <BrowserRouter/>

* Inorder to test the ~~click~~ onClick event in React, there is something called as FireEvent
  ↳ it comes from @testing-library/react

eg:-

```
render(<Component/>)
- const loginButton = screen.getByRole("button", {name: "Login"})
  fireEvent.click(loginButton);
                  ↳ Click event
- const logoutButton = screen.getByRole("button", {name: "Logout"})
  fireEvent.click(logoutButton);
```

* Passing Props inside our Component and do unit Testing.
  First create a mockData
  ↳ get the mock data
  ↳ create Sparate file .json
  ↳ Pass the MOCK_DATA as props in the Component

npm run watch-test ; will execute the test case automatically on making a change in code.

- You don't to run the test case everytime manually it will start automatically.

package.json
```
"scripts": {
  "watch-test": "jest --watch"
}
```

Whenever you have a async fn or state update
=> You need to wrap component in act() → Returns a Promise
↳ comes from react-dom/tests-utils

```
it(" ", async() => {
  await act(async() => render(<Body/>));
```

test.js

screen.getByTestId("search-input");

```
<input
  data-testid="search-input" />
```

if any of the method such as getByRole, getBylabel, doesn't work, then getByTestId will definetly work.

(helper fn)

- beforeAll(() => {}) — If you want to do something beforeAll test cases you will do it inside this fn
- beforeEach(() => {}) — If you want to do something beforeEach test case you will do it insid this beforeEach fn
- Similarly we have AfterAll() & afterEach()