RTL with Codevolution YouTube channel With RTL

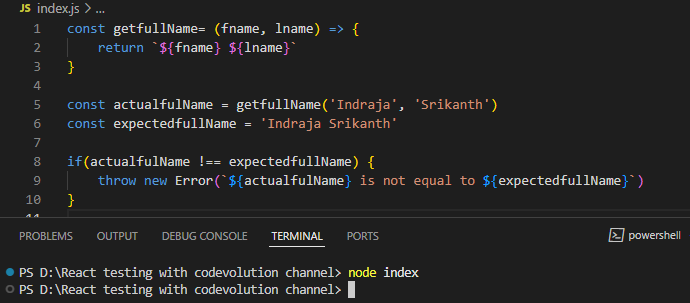
With RTL we are not concerned about the implementation details of a component. Instead we are testing how the components behaves when a user interacts with it.

Demo to understand testing:

Create a folder

Open VS code from that folder

Create a file index.js



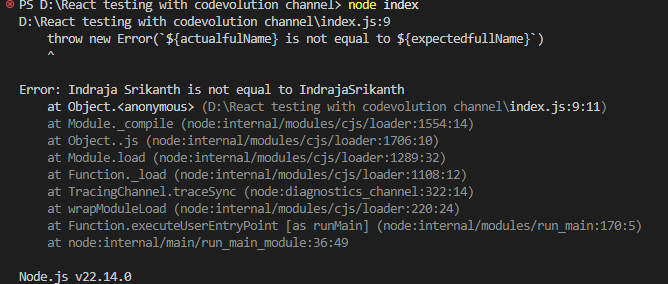
When you run node index you see nothing

but when you change

const expectedfullName = 'Indraja Srikanth' to

const expectedfullName = 'IndrajaSrikanth'

you see error like:



Here, we check the condition and throw error.

Similar way we do testing on UI components

React Project setup for testing:

* **Nodejs** should be installed
* **Git** should be installed
* Create folder (react-testing-codevolution-utube)
* Open folder in VScode
* Run the command: npx create-react-app . --template typescript
  + This command creates a new **React** project with **TypeScript** support in the **current directory**.
  + Here's what each part means:
    - npx - A tool that runs npm packages without installing them globally.
    - create-react-app - The CLI tool used to set up a new React project with a good default setup.
    - . - This tells the command to create the project **in the current folder**.
    - --template typescript - This tells Create React App to use the **TypeScript template** instead of JavaScript.

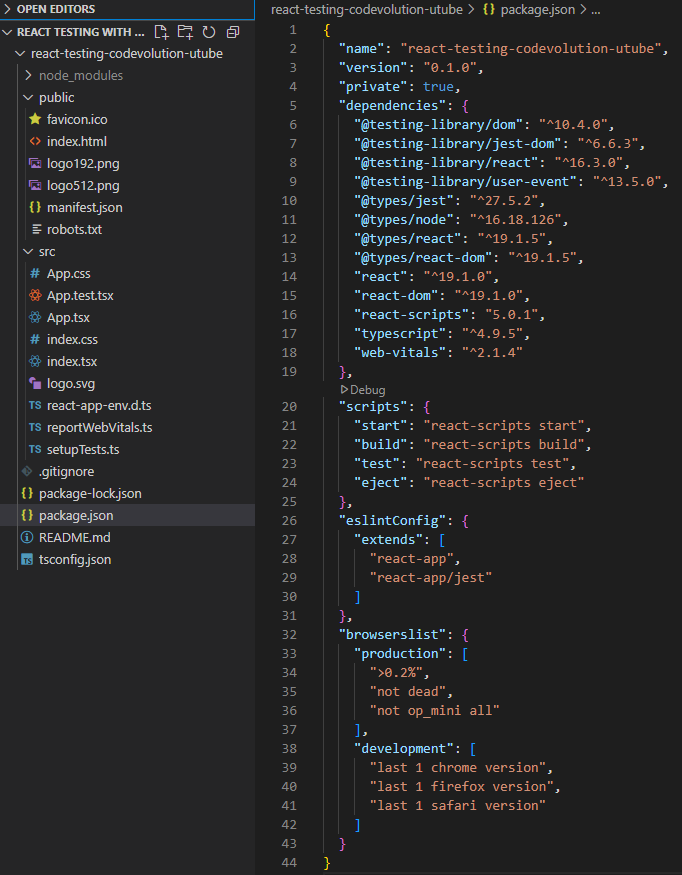
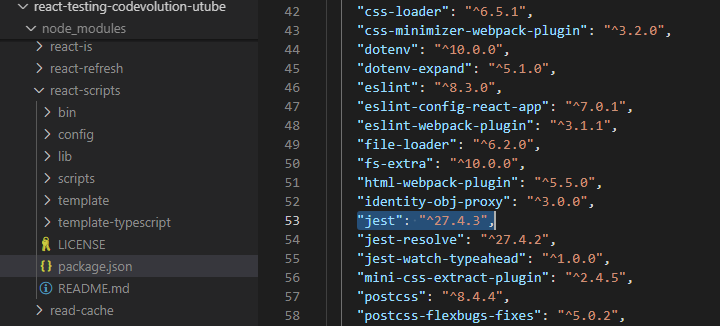
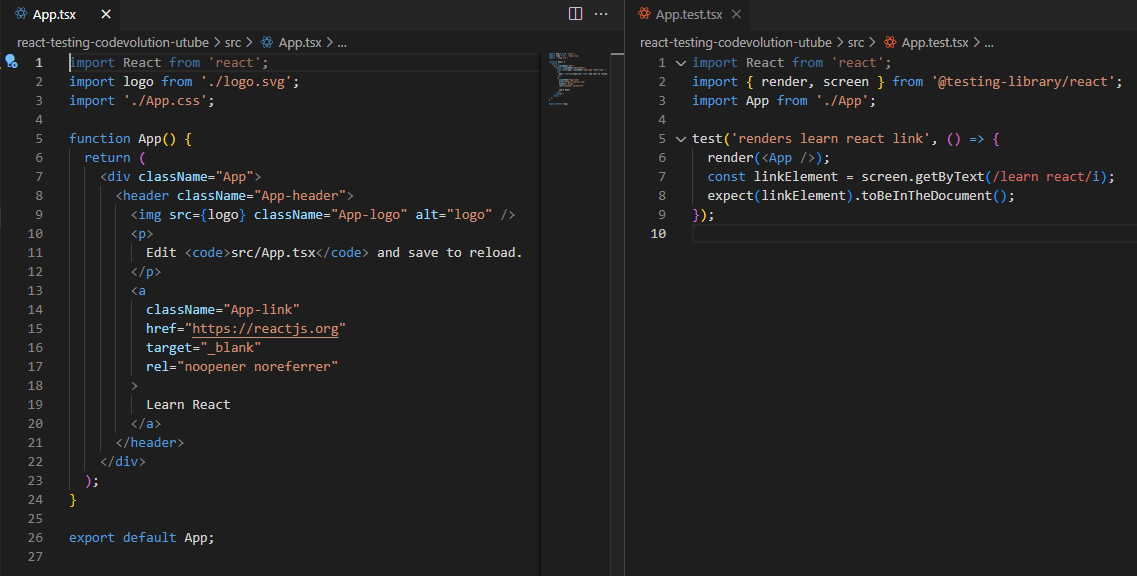


Fig: project structure and packages installed after running above command

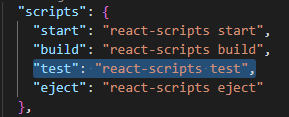
* **Jest** and **React testing library** package already installed and configured with **npx create-react-app** command, you can see **jest** in   
  node\_module folder -> react-scripts -> package.json



* Run npm start command to run the project
* In project structure you can see a default test file created for App.tsx

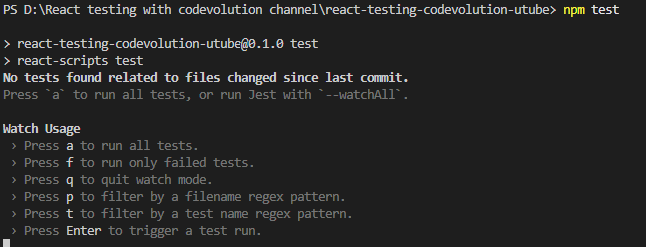


* **Jest** runs the test files

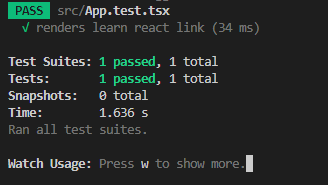


This `**test**` script defined here will internally call **jest**. So, when you run npm test, it actually runs: react-scripts test

* Run command npm test which will run all test files in project. And in a Create React App (CRA) project, **Jest** runs in **watch mode** by default.  
  **Watch mode** is a feature (used by tools like **Jest**, which powers npm test in React apps) that **automatically re-runs tests whenever your code changes**.

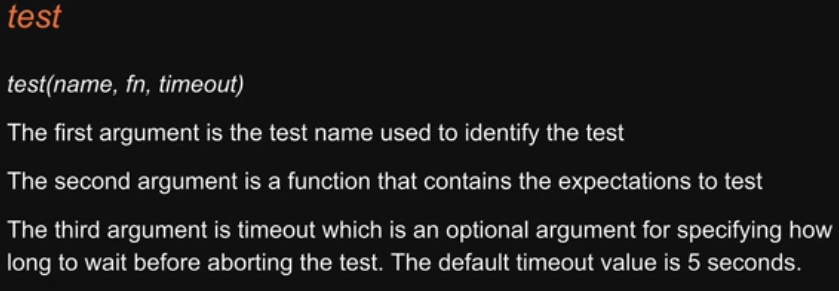


* Press `a` then we see



Here **jest** was able to find one test i.e., **renders learn react link** from a test suite/file i.e., **App.test.tsx** and run that test

Structure of jest’s test() in App.test.tsx:



**RTL** come into usage in 2nd argument.

Starting line would be **render()** from RTL which renders a React component into a virtual DOM for testing

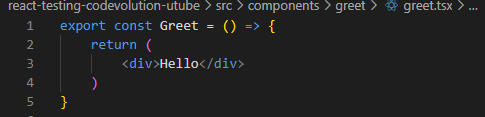
**screen** – object that provides methods to query elements (getByText, getByRole, etc.)

**test()**, **expect()**, and matcher method **toBeInTheDocument()** does not need to import as they comes from **jest** which are made globally available by **create-react-app** CLI.

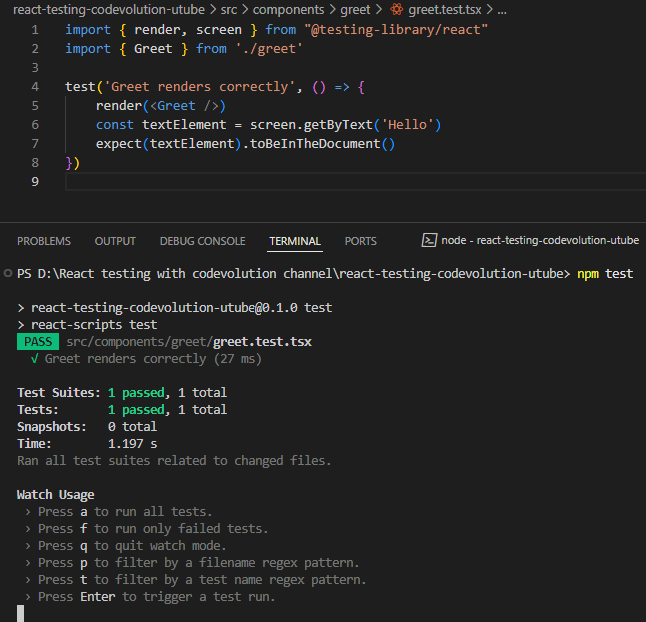
**Hence**, we make **use of Jest + RTL** in testing React components.

First test:

Create a functional component **Greet** in project



Create a test file for this **Greet** component and run **npm test**



If we pass text which is not present in **Greet** component JSX like:



Then test will fail with a message

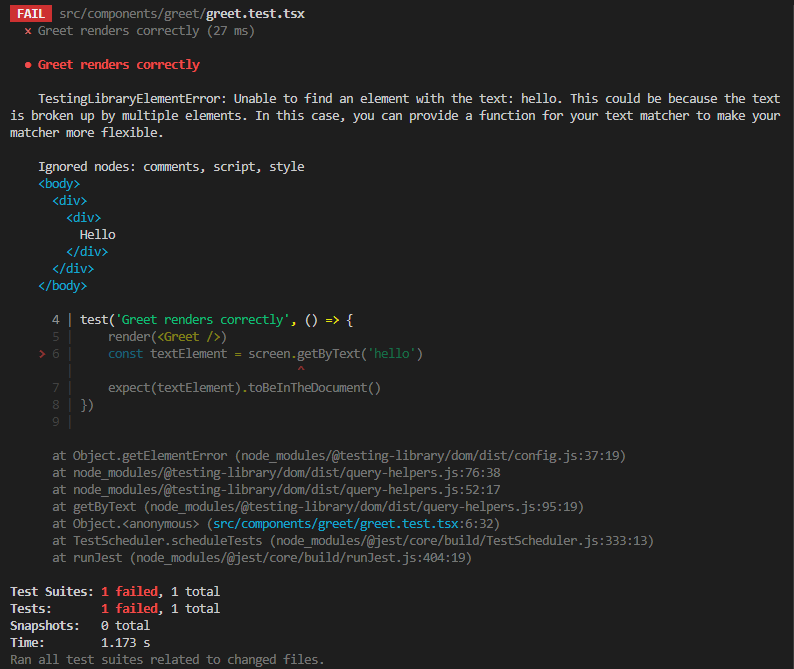
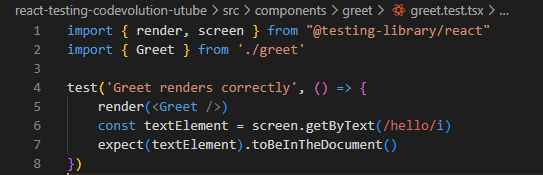


Fig: Unable to find text `hello` with lower-case `h`

We can fix this by using regex /hello/i



Test Driven Development(TDD):

TDD is a software development process where you write tests before writing the software code.

Once the tests have been written, you then write the code to ensure the tests pass.

It has 3 phases (often called Red-Green-Refactor):

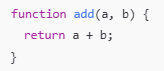
Red- write a failing test

You write a **test** for the functionality **before** writing the code. The test will **fail** because the feature hasn’t been implemented yet. This confirms the test is working and the feature doesn’t already exist.



Green – write the minimum code to pass the test:

Write **just enough code** to make the test **pass**. It doesn't have to be perfect — just enough to go green ✅.



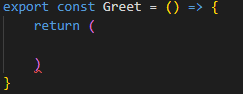
Refactor – Improve the code

Clean up or optimize the code **without changing its behavior**.

All tests should still pass after refactoring.

Let’s implement TDD:

* Let’s remove code from **Greet** component



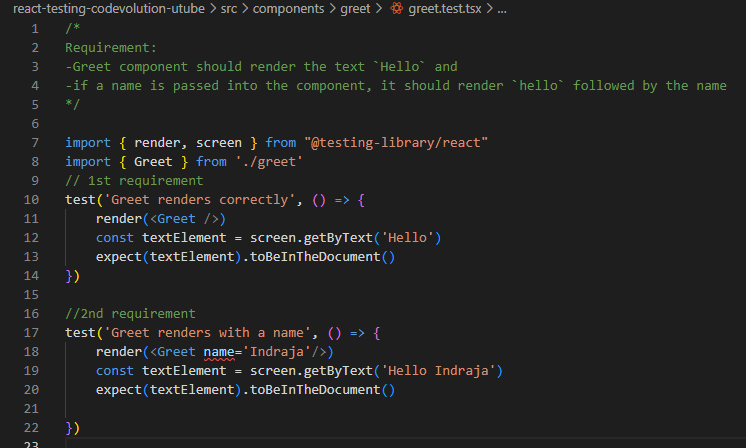
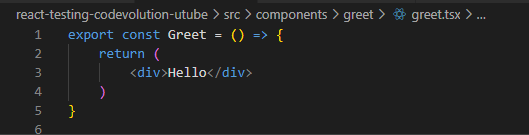
* Project Requirement ni base cheskoni test case rasthamu instead of component code

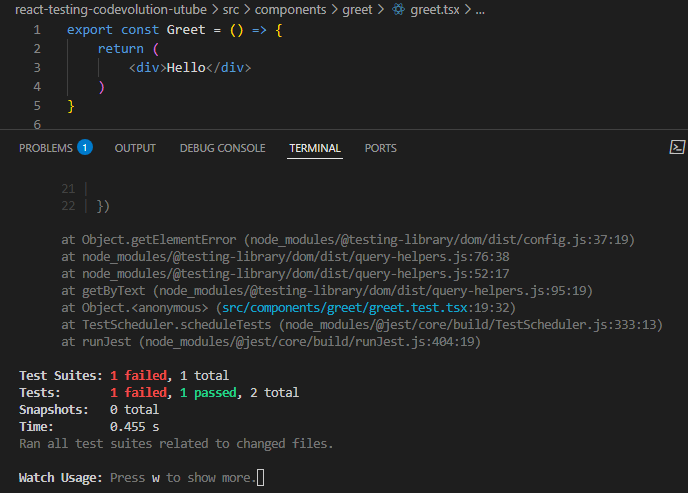
Fig: Requirement ni base cheskoni test case rasthamu instead of component code

* This is **Red phase** : When you run npm test, the test will fail as you did not implement the code in **Greet** component.
* Implement the code in **Greet** component



* You see 1 failed and 1 passed





* To make the 2nd test pass we need to pass `**name**` prop to **Greet** component.  
  Since we build React project support TypeScript, let define typescript variable in Greet component.

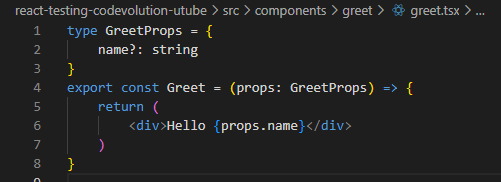
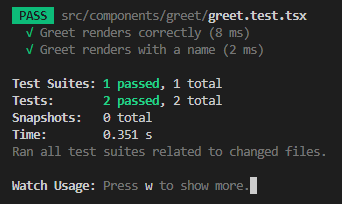


Fig: we specified `**props**` as type **GreetProps** to **Greet** component

* Now you we see all test pass  
  **Green phase:**



Understanding Jest Watch mode:

When you're running: npm test in a **Create React App (CRA) project**, Jest runs in **watch mode** by default.

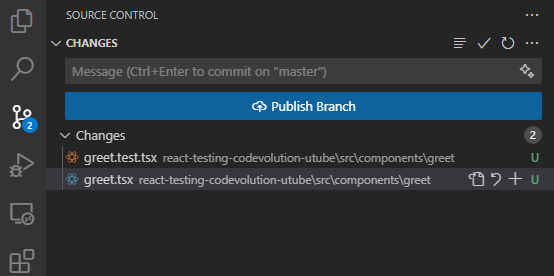
**This means:**

✅ It **watches** your files for changes.

🔁 If you edit a file, it **automatically re-runs the related tests**.

⏱️ Saves time during development — no need to manually rerun tests every time.

Example:



When we run `npm test`, **jest** will run tests only from these two files, even though we have **App.test.tsx** file in project **jest** does not run it.

Hence press `a` from watch mode to run all test files from the project.