Install:

Link for installation: <https://medium.com/@feralamillo/create-react-app-typescript-testing-with-jest-and-enzyme-869fdba1bd3>

npm i -D ts-jest jest-fetch-mock enzyme enzyme-adapter-react-16 enzyme-to-json @types/enzyme @types/enzyme-adapter-react-16 --save-exact

To config:

Put this code in package.json file above debug part

"jest": {

    "collectCoverageFrom": [

      "src/\*\*/\*.{js,jsx,ts,tsx}",

      "!src/\*\*/\*.d.ts",

      "!src/index.tsx",

      "!src/serviceWorker.ts"

    ],

    "coveragePathIgnorePatterns": [

      "./src/\*/\*.types.{ts,tsx}",

      "./src/index.tsx",

      "./src/serviceWorker.ts"

    ],

    "coverageReporters": [

      "json",

      "lcov",

      "text-summary",

      "clover"

    ],

    "coverageThreshold": {

      "global": {

        "statements": 95,

        "branches": 95,

        "lines": 95,

        "functions": 95

      }

    },

    "snapshotSerializers": [

      "enzyme-to-json/serializer"

    ],

    "transform": {

      "^.+\\.(js|jsx|ts|tsx)$": "<rootDir>/node\_modules/ts-jest"

    },

    "transformIgnorePatterns": [

      "[/\\\\]node\_modules[/\\\\].+\\.(js|jsx|ts|tsx)$",

      "^.+\\.module\\.(css|sass|scss)$"

    ],

    "moduleNameMapper": {

      "^react-native$": "react-native-web",

      "^.+\\.module\\.(css|sass|scss)$": "identity-obj-proxy",

      "src/(.\*)$": "<rootDir>/src/$1"

    }

  }

OR step 2 of above article

setupTest.js:

place this code in setupTest.js under src folder:

/\* eslint-disable import/no-extraneous-dependencies \*/

import Enzyme from 'enzyme';

import ReactSixteenAdapter from 'enzyme-adapter-react-16';

Enzyme.configure({ adapter: new ReactSixteenAdapter() });

Replace scripts in package.json:

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "test:coverage": "react-scripts test --coverage --runInBand --watchAll=false",

    "eject": "react-scripts eject",

    "lint": "eslint --ext .js,.jsx,.ts,.tsx src --color",

    "format": "prettier --write src/\*\*/\*.{ts,tsx,scss,css,json}",

    "isready": "npm run format && npm run lint && npm run test:coverage && npm run build"

  },

For test: create folder tests in src folder, and in tests folder, move your app.test.tsx file and paste this code or step 4 of article

import React from 'react';

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

import App from '../App';

test('renders learn react link', () => {

  render(<App />);

  const linkElement = screen.getByText(/learn react/i);

  expect(linkElement).toBeInTheDocument();

});

Now to run the test, use

npm run test

First test:

Create componets folder, place App.spec.tsx file their to run for test

In that write,

import React from 'react';

import { shallow } from 'enzyme';

import App from './App';

describe('App Component renders', () => {

    it('should render a div', ()=>{

        let container = shallow(<App />);

        expect(container.find('div').length).toEqual(1);

    });

})

Descrite is used for description of the test, in that we use it for test, and passed name to it, in arrow function we store object return by shallow, then from that container object, we expect to find the div and its length is 1, so we need to find 1 div in App.tsx in components App folder

Code for app.tsx:

import React from 'react';

const App = () => <div className='contaner'/>

export default App;

See, it contains 1 div only, so it will passed the test, one thing to remember, we first write test in App.spec.tsx, and than write code in App.tsx as per our test to pass.

Second test:

describe('App Component renders', () => {

    let container = shallow(<App />);

    it('should render a div', ()=>{

        expect(container.find('div').length).toEqual(1);

    });

    it("should render the Timer componet", () => {

        expect(container.containsAllMatchingElements(<Timer />)).toEqual(true);

    });

});

As shallow will render App, so we place it outside so it will not run everytime. The second test only checks whether <Timer /> tag is their in the App.tsx, it will not check what inside the Timer as shallow not check the content of child components, it will only check App component and find whether <Timer /> tag is present it or not, and we are saying that it is present that’s why providing it with true.

To check what inside Timer component or <Timer /> tag, we need to write another test case in Timer.spec.tsx in which we pass <Timer/> tag to shallow so that it can check its content

Timer folder:

Timer.spec.tsx ->test

import React from 'react';

import {shallow} from 'enzyme';

import Timer from './Timer';

describe('App Component renders', () => {

    let container = shallow(<Timer />);

    it('should render a div', ()=>{

        expect(container.find('div').length).toBeGreaterThanOrEqual(1);

    });

});

In Timer component, as it checks inside timer content, so their can be more thatn 1 div

Another test in timer.spec.tsx:

    it('should render instances of TimerButton component', () => {

        expect(container.find("TimerButton").length).toEqual(3)

    });

It checks whether we have 3 TimerButton components in Timer.tsx or not.