Unit Testing React Applications

React Unit Testing

- Testing the Smallest Unit of the Program is called Unit Testing.
- Every developer has to prove that Code written by the Developer is Working.
- Smallest Unit in React - Component/Hook
- You need to Write Unit Test for the Component for the following
    - Variables
    - Constrcutor with Dependencies
    - Functions
    - Component Template(HTML)

- JEST and React-testing-library are Unit Testing Library for React Applications.

- You need to write TestCase for Componet/Hook
- You need to write Multiple Unit Test for Componet
- You need to write Multiple Unit Test for Hook

- **When Unit Test will be Successed?**

Ans: When the Expected Value and Actual Value are same then Test is sucess.

- **When Unit Test will be failed?**

Ans: When the Expected Value and Actual Value are not same then Test is fail

- Every Unit Test should run Independently without sharing resources.
    - setUp() Method => Initializing/Creating Resources
    - tearDown() Method => Release the Resources

- Angular Provides the following methods for Resource Initialization and Resource Cleanup.
    - beforeEach()
    - afterEach()
    - beforeAll()
    - afterAll()

- beforeEach() method will be called Before executing each Unit Test
- afterEach() method will be called After executing each Unit Test
- beforeAll() method will be called Before executing all the Unit Tests of TestCase
- afterAll() method will be called After executing all the Unit Tests of TestCase

Unit Testing Demo - Working Steps:

1. Create the new project called **unit-testing-demo**

```
create-react-app unit-testing-demo
```

2. Delete the following 5 files from **public** folder
   a) favicon.ico
   b) logo192.png
   c) logo512.png
   d) manifest.json
   e) robots.txt

3. Delete the following 5 files from **src** folder
   a) App.css
   b) index.css
   c) logo.svg
   d) reportWebVitals.js

4. Update index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>JLC BookStore App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="myroot"></div>
  </body>
</html>
```

5. Update index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { App } from './App';

ReactDOM.render(<App />, document.getElementById('myroot'));
```

6. Update App.js

```
import React, { Component } from 'react';

export class App extends Component {
render() {
return (
<div>
<h1> Welcome to Unit Testing React Apps </h1>
<h3> Learn React from Srinivas Dande </h3>
</div>
);
}
}
```

7. Run the Project

   npm start

8. Open http://localhost:3000

9. Stop the Server.

10. Update **App.test.js**

11. Run the Tests

    npm run test

12. Wrire the **Hello Component - Hello.js**

13. Write the Unit Tests for **Hello component - Hello.test.js**

14. Run the Tests

    npm run test

15. Wrire the **CourseList Component - CourseList.js**

16. Write the Unit Tests for **CourseList Component – CourseList.test.js**

17. Run the Tests

    npm run test

18. Total of 10 Unit Tests from 3 Test Cases will run and show the Report.

unit-testing-demo - Files Required

1. index.html	2. index.js
3. App.js	4. App.test.js
5. Hello.js	6. Hello.test.js
7. CourseList.js	8. CourseList.test.js
9. setupTests.js	

1. index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>JLC BookStore App</title>
</head>
<body>
<noscript>You need to enable JavaScript to run this app.</noscript>
<div id="myroot"></div>
</body>  </html>
```

2. index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { App } from './App';

ReactDOM.render(<App />, document.getElementById('myroot'));
```

3. App.js

```
import React, { Component } from 'react';
import { CourseList } from './CourseList';
import { Hello } from './Hello';

export class App extends Component {
render() {
return (
<div>
<h1> Welcome to Unit Testing React Apps </h1>
<h3> Learn React from Srinivas Dande </h3>
<Hello />
<CourseList />
</div>
);
}  }
```

4. App.test.js

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

describe("App Component - TestCase", () => {

beforeEach(() => {
console.log("beforeEach");
});

afterEach(() => {
console.log("afterEach");
});

beforeAll(() => {
console.log("beforeAll of App Component");
});

afterAll(() => {
console.log("afterAll  of App Component");
});

test('renders Welcome to Unit Testing', () => {
render(<App />);
const textElement = screen.getByText(/Welcome to Unit Testing/i);
expect(textElement).toBeInTheDocument();
});

test('renders Learn React', () => {
render(<App />);
const textElement = screen.getByText(/Learn React/i);
//expect(textElement).not.toBeInTheDocument();
expect(textElement).toBeInTheDocument();

});

});
```

5. Hello.js

```
import { useState } from 'react';

export const Hello = () => {

const [showMessage, setShowMessage] = useState(false);

const showMessageHandler = () => {
setShowMessage(!showMessage);
}

return (
<div>
<h2> Hello Guys!!! </h2>
{!showMessage ?
<h3> Learning Unit Testing with React</h3>
: <h3> Hai, How is Testing </h3>
}

<button onClick={showMessageHandler}> Click Here to Change </button>
</div>
);
}
```

6. Hello.test.js

```
import { render, screen } from '@testing-library/react';
import userEvent from '@testing-library/user-event';

import { Hello } from './Hello';

describe("Hello Component - TestCase", () => {

beforeEach(() => {
console.log("beforeEach");
});

afterEach(() => {
console.log("afterEach");
});

beforeAll(() => {
console.log("beforeAll of Hello Component");
});
afterAll(() => {
console.log("afterAll  of Hello Component");
});

test("displays Hello Guys -  Test1", () => {
render(<Hello />);
const textElement = screen.getByText("Hello Guys", { exact: false });
expect(textElement).toBeInTheDocument();
});

test("displays Hello Guys!!! - Test2", () => {
render(<Hello />);
const textElement = screen.getByText("Hello Guys!!!");
expect(textElement).toBeInTheDocument();
});

test("displays Unit Testing with React -  if button is not clicked - test3", () => {
render(<Hello />);
const textElement = screen.getByText("Unit Testing with React", { exact: false });
expect(textElement).toBeInTheDocument();
});

test("displays How is Testing -  if button is clicked - test4", () => {
render(<Hello />);

const buttonElement = screen.getByRole("button");
userEvent.click(buttonElement);

const textElement = screen.getByText("How is Testing", { exact: false });
expect(textElement).toBeInTheDocument();
});

test("does not displays Unit Testing with React -  if button is clicked - test5", () => {
render(<Hello />);

const buttonElement = screen.getByRole("button");
userEvent.click(buttonElement);

const textElement = screen.queryByText("Unit Testing with React", { exact: false });
// expect(textElement).not.toBeNull();
expect(textElement).toBeNull();
});
});
```

7. CourseList.js

```
import React, { useEffect, useState } from "react";
import axios from 'axios';

export const CourseList = () => {

const [courses, setCourses] = useState([]);
const [message, setMessage] = useState("");

useEffect(() => {
axios.get("https://coursecube.com/jlc-rest-api/mini-courses")
.then(response => {
setCourses(response.data);
})
.catch(error => {
setMessage(error.message);
})

}, []);

const mycourseList = courses.map(
mycourse => {
return (
<li key={mycourse.courseId}> {mycourse.courseName} </li>
);
}
);

return (
<div className="container">
<ul>
{mycourseList}
</ul>
<h3> {message} </h3>
</div >
);
}
```

8. CourseList.test.js

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

import { CourseList } from './CourseList';

describe("CourseList Component - TestCase", () => {

beforeEach(() => {
console.log("beforeEach");
});

afterEach(() => {
console.log("afterEach");
});

beforeAll(() => {
console.log("beforeAll of CourseList Component");
});

afterAll(() => {
console.log("afterAll  of CourseList Component");
});

test("displays Course List if Request Success -  Test1", async () => {

render(<CourseList />);

const listElements = await screen.findAllByRole("listitem");
//expect(listElements).not.toHaveLength(0);
expect(listElements).toHaveLength(9);

});

test("displays Course List if Request Success -  Mock Test1", async () => {

const courses = [
{ courseId: 101, courseName: "Java Full Stack Course" },
{ courseId: 102, courseName: "Spring Boot Course" }
];

const response = { data: courses };
axios.get = jest.fn();
axios.get.mockResolvedValue(response);
render(<CourseList />);

const listElements = await screen.findAllByRole("listitem");
// expect(listElements).not.toHaveLength(0);
expect(listElements).toHaveLength(2);

});

test("displays Course List if Request Failed -  Mock Test2", async () => {

const error = { message: "Network Error" };
axios.get = jest.fn();
axios.get.mockRejectedValue(error);

render(<CourseList />);

const listElements = await screen.findByText("Network Error");
//expect(listElements).not.toBeInTheDocument();
expect(listElements).toBeInTheDocument();

});

});
```

9. setupTests.js

```
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom

import '@testing-library/jest-dom';
```