Digital-Nurture-4.0-JavaFSE-main

Hands-on in this document

Name:  Thilak Ragav M

Superset ID : 6409405

Week Task : Week - 6

**Objectives**

* Explain the need and Benefits of component life cycle
* Identify various life cycle hook methods
* List the sequence of steps in rendering a component

In this hands-on lab, you will learn how to:

* Implement componentDidMount() hook
* Implementing componentDidCatch() life cycle hook.

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties



Figure 2: Post class

1. Create a new class based component named as **Posts** inside **Posts.js** file



Figure 3: Posts Component

1. Initialize the component with a list of Post in state of the component using the constructor
2. Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>)



Figure 4: loadPosts() method

1. Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts



Figure 5: componentDidMount() hook

1. Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.



Figure 6: render() method

1. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.



Figure 7: componentDidCatch() hook

1. Add the Posts component to App component.
2. Build and Run the application using *npm start* command.

Index.js

import React from 'react';import ReactDOM from 'react-dom/client';import './index.css';import App from './App';import reportWebVitals from './reportWebVitals';const root = ReactDOM.createRoot(document.getElementById('root'));root.render( <React.StrictMode> <App /> </React.StrictMode>);// If you want to start measuring performance in your app, pass a function// to log results (for example: reportWebVitals(console.log))// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitalsreportWebVitals();

App.js

import React from 'react';import Posts from './Posts';function App() { return ( <div className="App"> <Posts /> </div> );}export default App;export default App;

Package.json:

{ "name": "blogapp", "version": "0.1.0", "private": true, "dependencies": { "@testing-library/dom": "^10.4.0", "@testing-library/jest-dom": "^6.6.3", "@testing-library/react": "^16.3.0", "@testing-library/user-event": "^13.5.0", "react": "^19.1.0", "react-dom": "^19.1.0", "react-scripts": "5.0.1", "web-vitals": "^2.1.4" }, "scripts": { "start": "react-scripts start", "build": "react-scripts build", "test": "react-scripts test", "eject": "react-scripts eject" }, "eslintConfig": { "extends": [ "react-app", "react-app/jest" ] }, "browserslist": { "production": [ ">0.2%", "not dead", "not op\_mini all" ], "development": [ "last 1 chrome version", "last 1 firefox version", "last 1 safari version" ] }}

Output:

 