**Lesson 07 Demo 01**

**Creating a Simple Debugger React Application**

**Objective:** To create a simple React application that consumes a REST API and debug it with React developer tools in the browser using breakpoints

**Tools required:** Node.js and React.js

**Prerequisites:** None

Steps to be followed:

1. Create and set up a React project
2. Create user-defined components: GitHubUser.js and Posts.js
3. Add the components to the App.js file
4. Test the application

**Step 1: Create and set up a React project**

1. Open a terminal window and run the following command to create a React application:

**npx create-react-app simple-debugger-react-app**

A screenshot of a computer

Description automatically generated

1. Open a terminal window inside the React project **simple-debugger-react-app**, and run the following command to install the required dependencies:

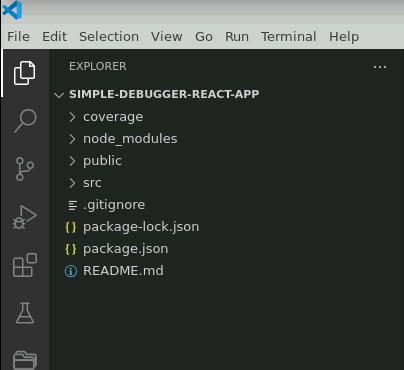
**npm install**

A screenshot of a computer

Description automatically generated

**Note:** This command helps to install all the required dependencies mentioned in the **package.json** file on the local machine in the form of a **node\_module** folder.

1. Open the **SIMPLE-DEBUGGER-REACT-APP** folder in the VS Code editor

****

1. Open the **package.json** file and view the external dependencies

**A screen shot of a computer

Description automatically generated**

**Step 2: Create user-defined components: GitHubUser.js and Posts.js**

1. Create a user-defined component **GitHubUser.js** with the following code:

**import React, { useState, useEffect } from 'react';**

**const GitHubUser = () => {**

**const [user, setUser] = useState(null);**

**const [loading, setLoading] = useState(true);**

**useEffect(() => {**

**const fetchUser = async () => {**

**try {**

**const response = await fetch('https://api.github.com/users/octocat');**

**const userData = await response.json();**

**setUser(userData);**

**setLoading(false);**

**} catch (error) {**

**console.error('Error fetching user:', error);**

**setLoading(false);**

**}**

**};**

**fetchUser();**

**}, []);**

**if (loading) {**

**return <div>Loading...</div>;**

**}**

**if (!user) {**

**return <div>User not found</div>;**

**}**

**return (**

**<div>**

**<h1>{user.login}</h1>**

**<img src={user.avatar\_url} alt={user.login} style={{ width: '100px', height: '100px' }} />**

**<p>Followers: {user.followers}</p>**

**</div>**

**);**

**};**

**export default GitHubUser;**

A computer screen shot of a program code

Description automatically generated

**Note**: **GitHubUser.js** file is responsible for interacting with the default GitHub account details on the current machine with your GitHub follower details. This API is consumed using the fetch() JavaScript function. You also have the option to use Axios third-party modules.

1. Create a **Posts.js** file that is responsible for calling fake REST API to consume the fake post details using fetch() functions with the following code:

**import React, { useState, useEffect } from 'react';**

**const Posts = () => {**

**const [posts, setPosts] = useState([]);**

**const [loading, setLoading] = useState(true);**

**const [error, setError] = useState(null);**

**useEffect(() => {**

**const fetchPosts = async () => {**

**try {**

**const response = await fetch('https://jsonplaceholder.typicode.com/posts');**

**//debugger;**

**if (!response.ok) {**

**throw new Error('Network response was not ok');**

**}**

**const data = await response.json();**

**setPosts(data);**

**setLoading(false);**

**} catch (error) {**

**setError(error.message);**

**setLoading(false);**

**}**

**};**

**fetchPosts();**

**}, []);**

**if (loading) {**

**return <div>Loading...</div>;**

**}**

**if (error) {**

**return <div>Error: {error}</div>;**

**}**

**return (**

**<div>**

**<h1>Posts</h1>**

**<ul>**

**{posts.map((post) => (**

**<li key={post.id}>**

**<h3>{post.title}</h3>**

**<p>{post.body}</p>**

**</li>**

**))}**

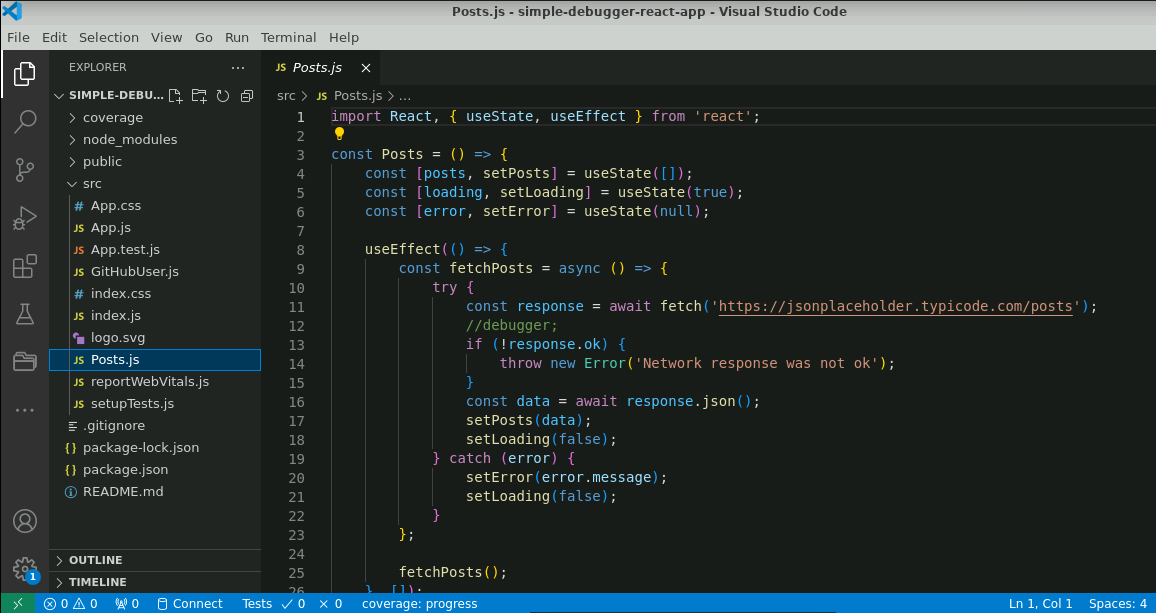
**</ul>**

**</div>**

**);**

**};**

**export default Posts;**



**Step 3:** **Add the components to the App.js file**

1. Add the components **GitHubUser.js** and **Posts.js** in the **App.js** file, as shown in the code below:

**import logo from './logo.svg';**

**import './App.css';**

**import GitHubUser from './GitHubUser';**

**import Posts from './Posts';**

**function App() {**

**return (**

**<div className="App">**

**<GitHubUser></GitHubUser>**

**<hr/>**

**<Posts></Posts>**

**</div>**

**);**

**}**

**export default App;**

A screen shot of a computer

Description automatically generated

**Step 4: Test the application**

1. Run the application using the **npm start** command

A screenshot of a computer

Description automatically generated

1. Open the browser and navigate to **localhost:3000**

A screenshot of a computer

Description automatically generated

1. Search for and click on the **React Developer Tools** extension to download **react-devtools** plugin, as shown in the screenshot below:

A screenshot of a browser

Description automatically generated

1. Click on the **Add to Chrome** button

A screenshot of a computer

Description automatically generated

1. Click on the **Add extension** button and then restart the browser

A screenshot of a computer

Description automatically generated

1. Restart the application with the **npm start** command

A screenshot of a computer

Description automatically generated

1. Open the debugger option in the Chrome browser using **ctrl + shift + I**,then click on **Components** as shown below:

A computer screen shot of a computer screen

Description automatically generated

1. On the **Components** tab, click on the **GitHubUser** to check the component details:

A screenshot of a computer

Description automatically generated

1. Click on the **Posts** to check the component details:

A screenshot of a computer

Description automatically generated

1. Open the source code of **GitHubUser.js** or **Posts.js** and use the JavaScript **debugger** keyword to set a breakpoint, as shown below:

A screen shot of a computer program

Description automatically generated

1. Check and debug the application in the browser window as shown below:

A screenshot of a computer

Description automatically generated

**Note:** Click on the **debugger** and check the code in the browser to debug.

With this, you have successfully created a simple React application that consumes a REST API and debugged it with React developer tools in the browser using breakpoints.