Lesson 04 Demo 06
Fetch Data Using Custom Hook

Objective: To develop a React application that demonstrates fetching data using custom Hooks

Tools required: Node terminal, React App, and Visual Studio Code

Prerequisites: Knowledge of creating a React app and understanding of the folder structure

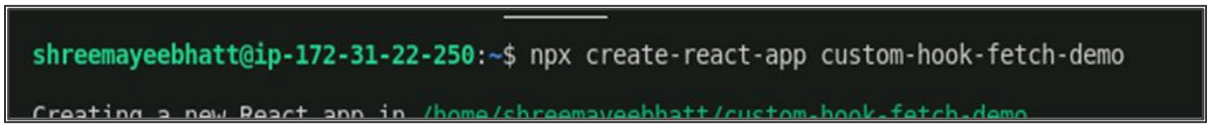
Steps to be followed:

1. Create a new **React** app
2. Create a new file called **useFetch.js** in the **src** directory
3. Import **useFetch** from the **useFetch.js file** in **App.js**
4. Run the app and view it in the browser

Step 1: Create a new React app

1.1 Open your terminal and run the following command:

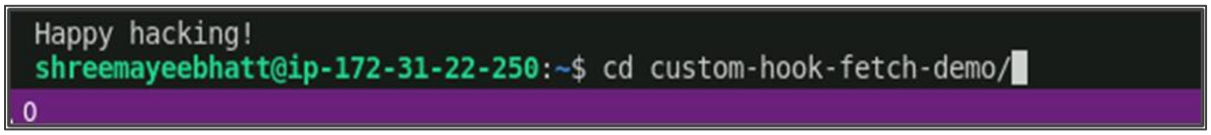
npx create-react-app custom-hook-fetch-demo



```
shreemayeebhatt@ip-172-31-22-250:~$ npx create-react-app custom-hook-fetch-demo
```

This command will create a new **React** app with the name **custom-hook-fetch-demo**

1.2 Run the **cd custom-hook-fetch-demo** command in the terminal

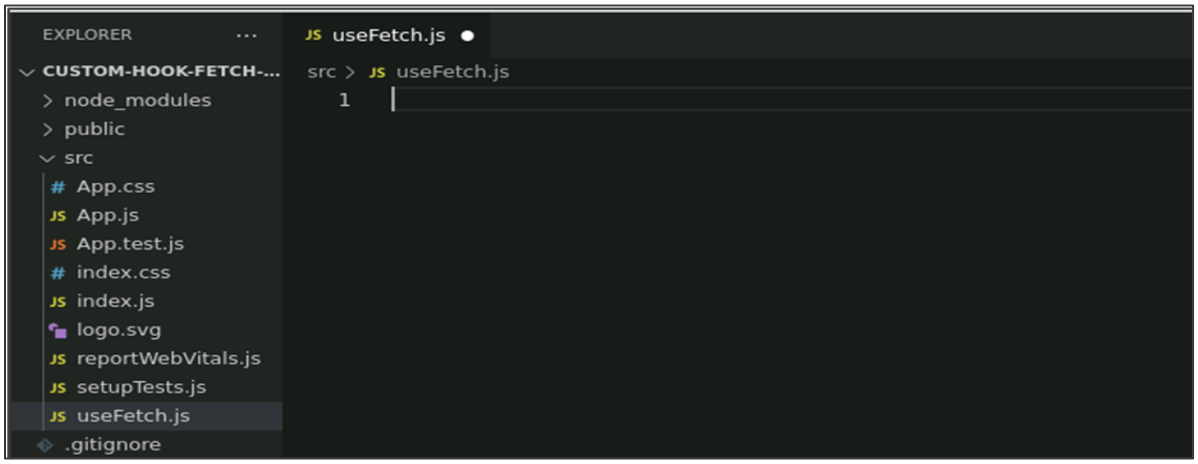


```
Happy hacking!
shreemayeebhatt@ip-172-31-22-250:~$ cd custom-hook-fetch-demo/
```

This will change the current directory to the newly created **React** app directory.

Step 2: Create a new file called useFetch.js in the src directory

2.1 Open your React project **custom-hook-fetch-demo** with the **Visual Studio Code** editor. Create the **useFetch.js** file within the **src** directory.



This file will define the **custom useFetch** Hook.

2.2 Inside the **useFetch.js** file, import the **useState** and **useEffect** Hooks from the React library

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

2.3 Define the **useFetch** function that takes a URL as a parameter

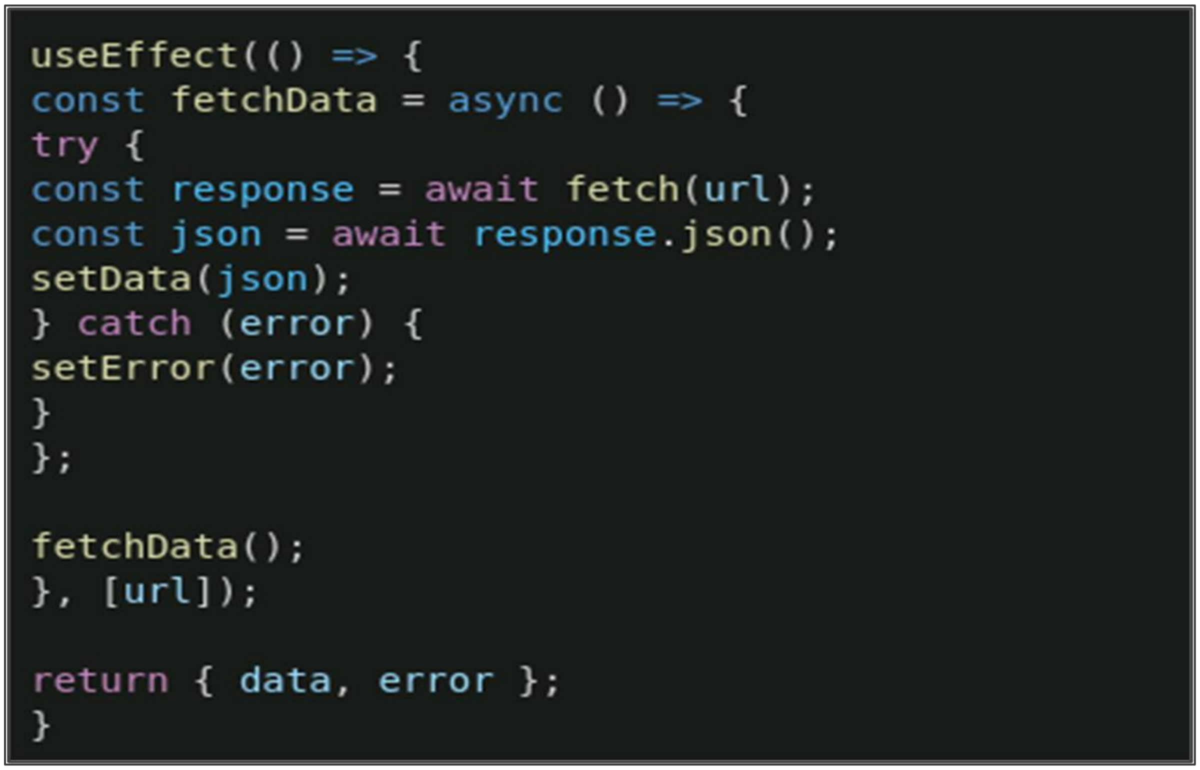
```
function useFetch(url) {
```

2.4 Use the **useState** Hook to create the **data** and **error** state variables

```
const [data, setData] = useState(null);
const [error, setError] = useState(null);
```

2.5 Use the **useEffect** Hook to fetch data from a specified **URL** using the **GET** request and convert that data in JSON format with **.json()**. Finally, handle any potential errors by using the try-catch block.

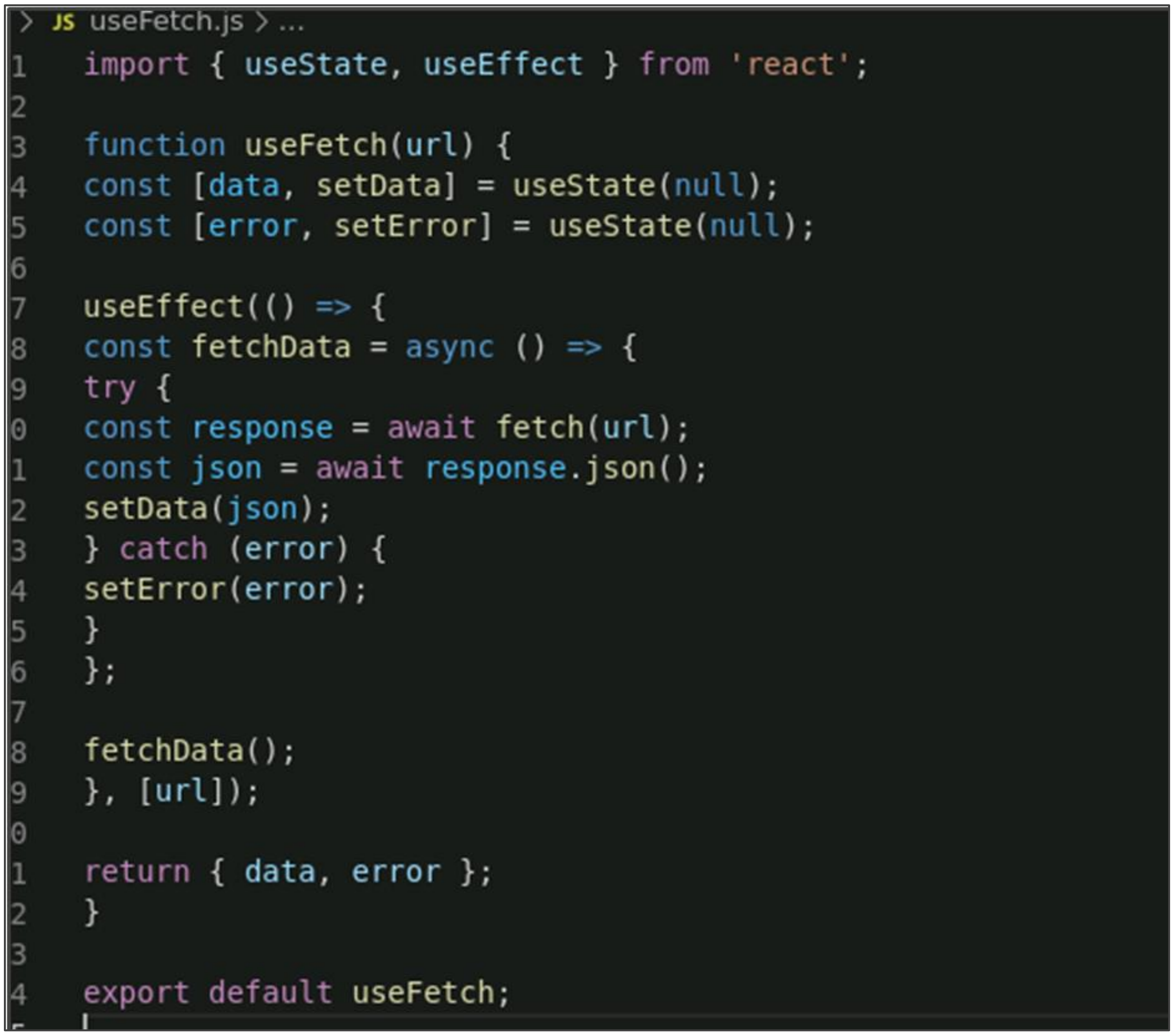
```
useEffect(() => {
const fetchData = async () => {
try {
const response = await fetch(url);
const json = await response.json();
setData(json);
} catch (error) {
setError(error);
} };
fetchData();
}, [url]);
return { data, error };
}
```



```
useEffect(() => {
const fetchData = async () => {
try {
const response = await fetch(url);
const json = await response.json();
setData(json);
} catch (error) {
setError(error);
}
};

fetchData();
}, [url]);

return { data, error };
}
```

Note: Refer to the following code to configure the **useFetch.js** file:

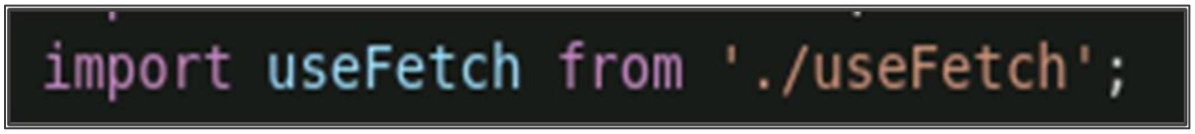
```
import { useState, useEffect } from 'react';

function useFetch(url) {
const [data, setData] = useState(null);
const [error, setError] = useState(null);

useEffect(() => {
const fetchData = async () => {
try {
const response = await fetch(url);
const json = await response.json();

setData(json);
} catch (error) {
setError(error);
}
};

fetchData();
}, [url]);

return { data, error };
}

export default useFetch;
```



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

function useFetch(url) {
const [data, setData] = useState(null);
const [error, setError] = useState(null);

useEffect(() => {
const fetchData = async () => {
try {
const response = await fetch(url);
const json = await response.json();
setData(json);
} catch (error) {
setError(error);
}
};

fetchData();
}, [url]);

return { data, error };
}

export default useFetch;
```

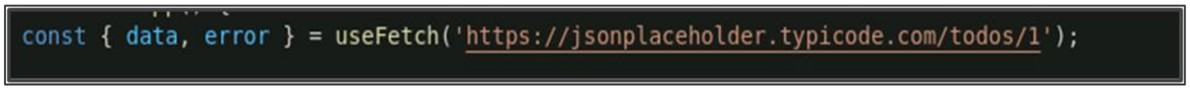
Step 3: Import useFetch from the useFetch.js file in App.js

3.1 Open the **App.js** file in the **Visual Studio Code** editor and import the **useFetch** Hook from the **useFetch.js** file



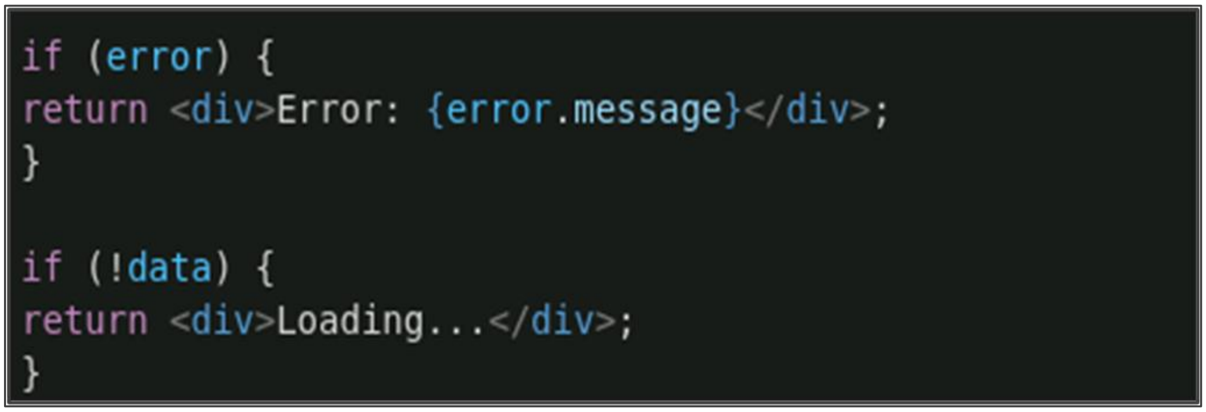
```
import useFetch from './useFetch';
```

3.2 Inside the App function component, call the **useFetch** Hook and assign the returned values to variables



```
const { data, error } = useFetch('https://jsonplaceholder.typicode.com/todos/1');
```

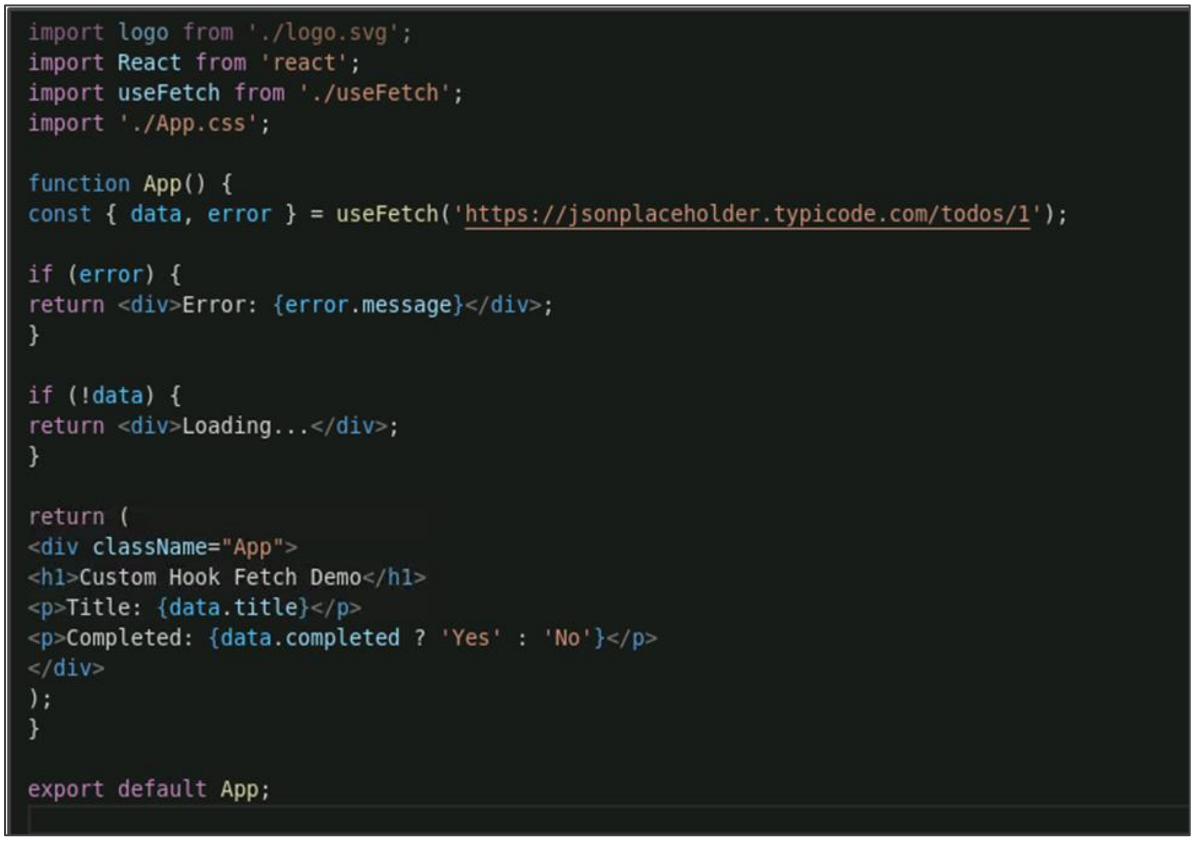
3.3 Use conditional rendering to handle the loading state and error state, and display the fetched data



```
if (error) {
return <div>Error: {error.message}</div>;
}

if (!data) {
return <div>Loading...</div>;
}
```

Note: Refer to the following code to configure the **App.js** file:

```
import React from 'react';
import useFetch from './useFetch';
import './App.css';

function App() {
const { data, error } = useFetch('https://jsonplaceholder.typicode.com/todos/1');

if (error) {
return <div>Error: {error.message}</div>;
}

if (!data) {
return <div>Loading...</div>;
}
return (

<div className="App">
<h1>Custom Hook Fetch Demo</h1>
<p>Title: {data.title}</p>
<p>Completed: {data.completed ? 'Yes' : 'No'}</p>
</div>
);
}

export default App;
```

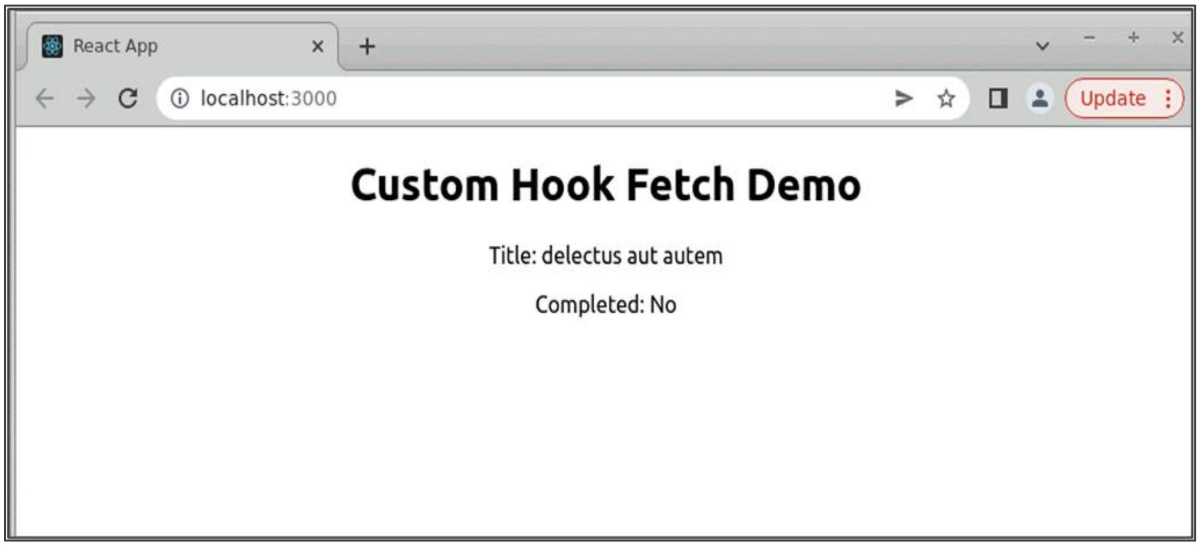


```
import logo from './logo.svg';
import React from 'react';
import useFetch from './useFetch';
import './App.css';

function App() {
const { data, error } = useFetch('https://jsonplaceholder.typicode.com/todos/1');

if (error) {
return <div>Error: {error.message}</div>;
}

if (!data) {
return <div>Loading...</div>;
}

return (
<div className="App">
<h1>Custom Hook Fetch Demo</h1>
<p>Title: {data.title}</p>
<p>Completed: {data.completed ? 'Yes' : 'No'}</p>
</div>
);
}

export default App;
```

Step 4: Run the app and view it in the browser

4.1 In the terminal, run the **npm start** command to start the app

4.2 Open your browser and navigate to **http://localhost:3000**



```
Custom Hook Fetch Demo

Title: delectus aut autem

Completed: No
```

The app should be running, and you should see a simple app that fetches and displays data from the **JSONPlaceholder API** using the custom **useFetch** Hook.