

Lesson 06 Demo 01

JSON Response from API

Objective: To develop a React application that demonstrates API calls with React

Tools Required: Node Terminal, React App, and Visual Studio Code

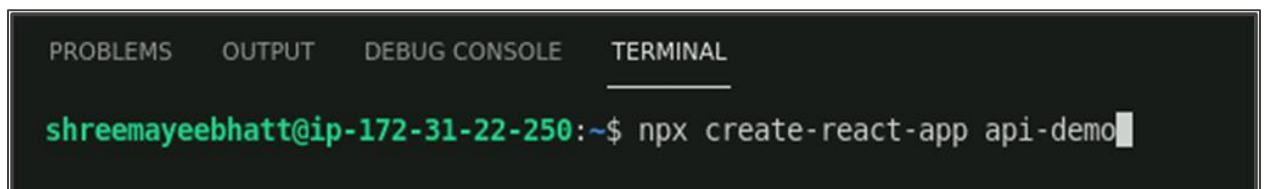
Prerequisites: Knowledge of creating a React app and understanding of the folder structure

Steps to be followed:

1. Create a new **React** app
2. Modify the **src/App.js** by defining a state variable
3. Run the app and view it in a browser

Step 1: Create a new React app

- 1.1 Open the terminal and run the command **npx create-react-app api-demo**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
shreemayeebhatt@ip-172-31-22-250:~$ npx create-react-app api-demo
```

This command will create a new **React** app with the name **api-demo**.

- 1.2 Run the command **cd api-demo** in the terminal to move inside the newly created directory

Step 2: Modify the src/App.js by defining a state variable

- 2.1 Open the React app with the **Visual Studio Code** editor and open **src/App.js** file. Modify the **App.js** file by importing **React**, **useState**, and **useEffect** from **react** library.

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

- 2.2 Define the **App** functional component, and inside it, define a state variable called **data** using the **useState** hook, and initialize it as **null**

```
4 function App() {  
5   const [data, setData] = useState(null);  
6 }
```

- 2.3 Define an asynchronous function called **fetchData** that makes an API call using the **fetch** function and updates the state with the response data

```
async function fetchData() {
```

- 2.4 Use the **useEffect** hook to call the **fetchData** function when the component mounts by passing an empty **dependency array []** as the second argument to **useEffect**

```
async function fetchData() {  
  const response = await fetch('https://jsonplaceholder.typicode.com/todos/1');  
  const jsonData = await response.json();  
  setData(jsonData);  
}  
  
useEffect(() => {  
  fetchData();  
}, []);
```

- 2.5 In the **return** statement, render a **div** and use a conditional rendering to display the **API response** if the data state is not null, and if it is null, render a **p** element with the text **Loading...** and export the **App** component as the default export of the file

```
17   return (  
18     <div className="App">  
19       <h1>API Demo</h1>  
20       {data ? (  
21         <p>API response: {JSON.stringify(data)}</p>  
22       ) : (  
23         <p>Loading...</p>  
24       )}  
25     </div>  
26   );  
27 }  
28  
29 export default App;  
30 |
```

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

```
import React, { useState, useEffect } from 'react';  
import './App.css';
```

```
function App() {  
  const [data, setData] = useState(null);
```

```
  async function fetchData() {  
    const response = await fetch('https://jsonplaceholder.typicode.com/todos/1');  
    const jsonData = await response.json();  
    setData(jsonData);  
  }  
  useEffect(() => {  
    fetchData();  
  }, []);
```

```
  return (  
    <div className="App">  
      <h1>API Demo</h1>
```

```
      {data ? (  
        <p>API response: {JSON.stringify(data)}</p>  
      ) : (  
        <p>Loading...</p>  
      )}
```

```

    <p>Loading...</p>
  })
</div>
);
}

```

export default App;

```

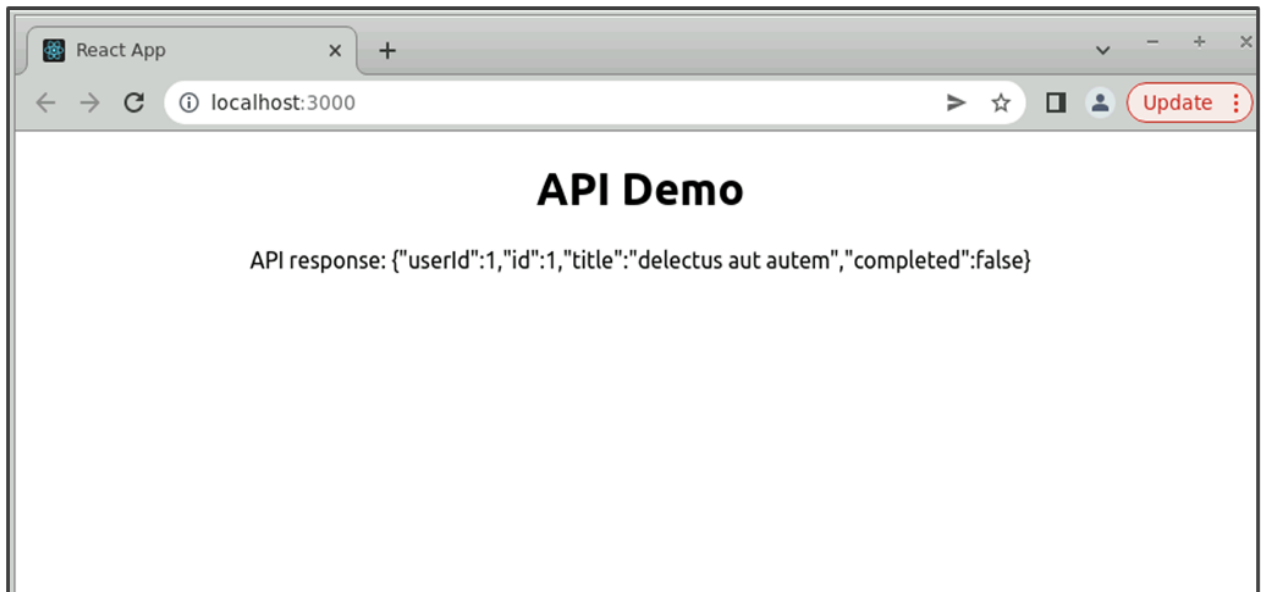
src > JS App.js > ...
1  import React, { useState, useEffect } from 'react';
2  import './App.css';
3
4  function App() {
5    const [data, setData] = useState(null);
6
7    async function fetchData() {
8      const response = await fetch('https://jsonplaceholder.typicode.com/todos/1');
9      const jsonData = await response.json();
10     setData(jsonData);
11   }
12
13   useEffect(() => {
14     fetchData();
15   }, []);
16
17   return (
18     <div className="App">
19       <h1>API Demo</h1>
20       {data ? (
21         <p>API response: {JSON.stringify(data)}</p>
22       ) : (
23         <p>Loading...</p>
24       )}
25     </div>
26   );
27 }
28
29 export default App;
30

```

Step 3: Run the app and view it in a browser

3.1 In the terminal, navigate to the project directory and run the **npm start** command to start the development server

3.2 Open your browser and navigate to <http://localhost:3000>



You should see a simple app that displays the **JSON** response from the **API**, initially showing **Loading...** and then updating with the actual response once it is fetched.

With this, you have successfully created an app to demonstrate the API call in React.