**Creating a Sample Service in React JS**

Table of Contents

* [Setting Up the Project](https://www.crsinfosolutions.com/creating-a-sample-service-in-react-js/#setting-up-project)
* [Creating the Service](https://www.crsinfosolutions.com/creating-a-sample-service-in-react-js/#creating-service)
* [Creating the View](https://www.crsinfosolutions.com/creating-a-sample-service-in-react-js/#creating-the-view)
* [Running the Application](https://www.crsinfosolutions.com/creating-a-sample-service-in-react-js/#run-application)
* [FAQs](https://www.crsinfosolutions.com/creating-a-sample-service-in-react-js/#faqs)

React JS has revolutionized the way we build web applications. It offers a component-based approach, making it easier to manage and maintain large applications. In this blog post, we’ll create a simple service in React and connect it to a view. This service will demonstrate how to manage state and pass data between components.

**Setting Up the Project**

Before we dive into coding, ensure that you have Node.js installed on your machine. Then, create a new React app by running the following command:

**npx create-react-app react-service-example**

**cd react-service-example**

**Creating the Service**

Let’s start by creating a simple service. In React, a service is typically a set of functions that handle data fetching, processing, or any other logic that isn’t directly related to rendering the UI.

1. **Create a New File for the Service:**  
   Inside your project, create a new file named SampleService.js in the src directory.
2. **Writing the Service Logic:**  
   In SampleService.js , we’ll create a function that simulates fetching data from an API.

const fetchData = async () => {

try {

// Simulate an API call

const response = await new Promise(resolve => {

setTimeout(() => resolve({ data: 'Sample Data' }), 1000);

});

return response.data;

} catch (error) {

console.error("Error fetching data:", error);

}

};

export { fetchData };

**Explore:**[**How Can You Pass Props to Children Components in React?**](https://www.crsinfosolutions.com/how-can-you-pass-props-to-children-components-in-react/)

**Creating the View**

Now, let’s create a view that uses this service.

1. **Modify App.js:**  
   Open App.js and make the following changes:

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

import './App.css';

import { fetchData } from './SampleService';

function App() {

const [data, setData] = useState('');

useEffect(() => {

const loadData = async () => {

const result = await fetchData();

setData(result);

};

loadData();

}, []);

return (

<div className="App">

<header className="App-header">

<p>

Fetched Data: {data}

</p>

</header>

</div>

);

}

export default App;

In this code, we import the fetchData function from our service. We then use the useEffect hook to call this function when the component mounts. The fetched data is stored in the data state variable and displayed in the view.

**Explore:**[**Lifecycle Methods in React**](https://www.crsinfosolutions.com/understanding-lifecycle-methods-in-react-react-js-tutorial-5/)

**Running the Application**

To see your service and view in action, run the following command in the terminal:

npm start

This will start the development server and open your new React application in the browser.

CRS Info Solutions stands out for its exceptional [React.js training in Hyderabad](https://www.crsinfosolutions.com/react-js-training-hyderabad/), tailored specifically for students. Their program focuses on practical, hands-on learning, ensuring that students not only understand [React.js training Bangalore](https://www.crsinfosolutions.com/react-js-training-in-bangalore/) concepts but also apply them effectively in real-world scenarios. This approach has established CRS Info Solutions as a go-to destination for aspiring React.js developers in the region.