

Cheat Sheet: In-depth Understanding of Advanced React Functionality

Hooks and form management	Description	Code example
useState()	useState() hook can manage states of the React function component where you can declare any data type, for example, boolean, object, array, string.	<pre>import React, { useState, useEffect } from 'react'; function SideEffect() { const [empId, setEmpId] = useState(100); return (<div> <p>{empId}</p> </div>); } export default SideEffect;</pre>
useEffect()	useEffect is a React hook that allows you to perform side effects in functional components. A side effect refers to any operation that you need to execute as soon as the page loads without calling those operations/functionalities separately, such as fetching data from an API.	<pre>import React, { useState, useEffect } from 'react'; function SideEffect() { const [foods, setFoods] = useState([]); useEffect(() => { fetch('https://api.npoint.io/d542b9ad99f501ab3dbf') .then(response => response.json()) .then(data => { console.log(data); setFoods(data); }) .catch(error => console.error('Error fetching users:', error)); }, []); // Empty dependency array means this effect runs only once when the component mounts return (<div> <h1>Food List</h1> {foods.map((food)=>{ return (<h1>{food.name}</h1> <p>food.description</p> <p>food.price</p> <p>food.category</p> <p>food.ingredients</p>) })} </div>); } export default SideEffect;</pre>
Custom hook	You can use custom hooks in any other component. In this code snippet, there is one function component known as UseToggle, which serves as a custom hook, and another function component ToggleButton, which will use this custom hook.	<pre>//ToggleButton import { useState } from 'react'; import UseToggle from './UseToggle'; function ToggleButton() { const [isToggled, toggle] = UseToggle(false); return (<div> <h1>Toggle Button</h1> <button onClick={toggle}> {isToggled ? 'ON' : 'OFF'} </button> </div>); } export default ToggleButton; //UseToggle.jsx import { useState } from "react"; function UseToggle(initialValue = false) { const [value, setValue] = useState(initialValue); const toggle = () => { setValue(!value); }; return [value, toggle]; } export default UseToggle</pre>
fetch api method	Fetch method can fetch data using API.	<pre>const apiUrl = 'https://jsonplaceholder.typicode.com/posts'; fetch(apiUrl) .then(response => response.json()) .then(data => { console.log(data); }) .catch(error => { console.error('There was a problem with the fetch operation:', error); });</pre>
axios api method	Axios method can fetch data using API.	<pre>import axios from 'axios'; const apiUrl = 'https://jsonplaceholder.typicode.com/posts'; axios.get(apiUrl) .then(response => { console.log(response.data); })</pre>

		<pre>.catch(error => { console.error('There was a problem with the fetch operation:', error); });</pre>
onChange	<p>The <code>onChange</code> event attribute is often used in HTML and React to track when the value of an input field changes, like a text input. The <code>onChange</code> event occurs when a user writes something into an input field. This attribute lets you record and handle the changes.</p>	<pre>import React, { useState } from 'react'; function FormData() { const [empName, setEmpName] = useState(''); const handleChange = event => { setEmpName(event.target.value); }; const handleSubmit = event => { event.preventDefault(); console.log('Form submitted:', empName); }; return (<div> <h2>My Form</h2> <form onSubmit={handleSubmit}> <label> Input: <input type="text" value={empName} onChange={handleChange} /> </label> <button type="submit">Submit</button> </form> </div>); }; export default FormData;</pre>
Redux toolkit	Redux toolkit can be installed using <code>npm</code>	<pre>npm install @reduxjs/toolkit.</pre>



Skills Network