**Stop using useEffect! How you can fetch data without useEffect ? React Query Data Fetching technique**

[[Atihar Hossan Mahir](https://ah-mahir.medium.com/?source=post_page-----2fb4dd3e0332--------------------------------)](https://ah-mahir.medium.com/?source=post_page-----2fb4dd3e0332--------------------------------)

[Atihar Hossan Mahir](https://ah-mahir.medium.com/?source=post_page-----2fb4dd3e0332--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F149a621bcf9a&operation=register&redirect=https%3A%2F%2Fah-mahir.medium.com%2Fstop-using-useeffect-2fb4dd3e0332&user=Atihar+Hossan+Mahir&userId=149a621bcf9a&source=post_page-149a621bcf9a----2fb4dd3e0332---------------------post_header-----------)

3 min read

·

Jul 22

109

2

The use of **useEffect**in React applications is not inherently bad. It is a crucial part of the React Hooks system that is meant to deal with side effects (like data fetching, subscriptions, timers, manual DOM manipulations, etc.) in functional components.

**However, there are scenarios or ways of using useEffect that could potentially lead to problems:**

**Incorrect Dependency Array:** One of the most common mistakes with useEffect is not correctly specifying dependencies. If the dependencies of your effect are wrong, it may run more often than needed, causing performance issues.

**Cleanup Issues:**useEffect provides a way to run cleanup code by returning a function from within it. If you forget to cleanup, it can lead to issues such as memory leaks, especially when dealing with subscriptions or event listeners.

**Async Effects:**React requires useEffect to return a cleanup function or nothing. Returning a Promise (e.g. when using an async function directly as your effect function) will lead to a console warning. This often confuses people new to hooks.

**Order of Execution:**useEffect runs after the render is committed to the screen. Therefore, if you need to run some code before the browser updates the screen, useEffect might not be the right choice.

**Overfetching:** If not managed well, data fetching within useEffect can lead to overfetching. Libraries like react-query and swr are designed to deal with caching and avoid unnecessary fetches.

Remember, these are not problems with useEffect itself, but more about how it is used. useEffect is a powerful tool when understood and used correctly, but it also requires a good understanding of its nuances to avoid potential pitfalls.

**Fetch data using React-Query**

react-query is a powerful data fetching and state management library in React. It abstracts the fetching, caching, synchronization, and updating of server state in your react applications.

Here is an example of how you can use react-query to fetch data instead of useEffect:

Firstly, make sure to install react-query:

npm install react-query  
# or  
yarn add react-query

Then start using it (example)

import React from 'react';  
import { useQuery } from 'react-query';  
  
// The function that fetches the data  
async function fetchData() {  
 const res = await fetch('https://api.example.com/data');  
 if (!res.ok) {  
 throw new Error('Network response was not ok');  
 }  
 return res.json();  
}  
  
function MyComponent() {  
 const { data, status } = useQuery('myData', fetchData);  
  
 if (status === 'loading') {  
 return <div>Loading...</div>;  
 }  
  
 if (status === 'error') {  
 return <div>Error fetching data</div>;  
 }  
  
 return (  
 <div>  
 {data && data.map((item, index) => (  
 <div key={index}>  
 {/\* Render your data here \*/}  
 </div>  
 ))}  
 </div>  
 );  
}  
  
export default MyComponent;

In this example, **useQuery**fetches the data from the API. It automatically handles loading and error states, and provides them along with the fetched data in its return object.

Note that **useQuery**takes two arguments: a unique key for the data (which react-query uses for caching) and a function that fetches the data.

The advantage of using react-query over useEffect is that it provides many out-of-the-box features such as caching, automatic **refetching**, background updates, pagination, and more. It also improves the performance and user experience of your applications.

Let me know if you have already used the react query based API calling and what’s your thought on this?