Getting Started with useQuery of React Query

Dhrubo Dh · Follow

3 min read · Mar 25, 2021

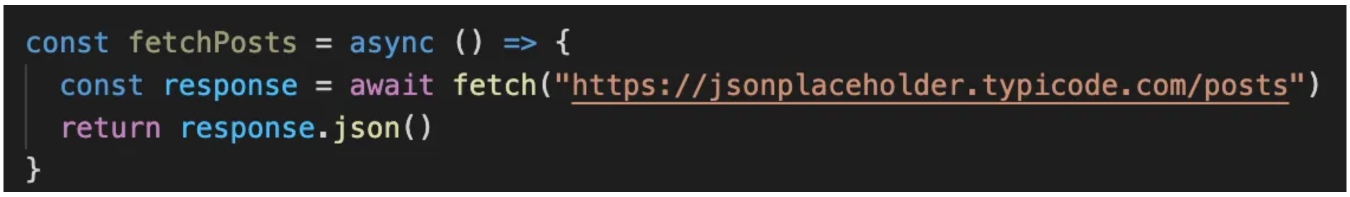
React Query is a ReactJS library that provides state management ability for any kind of asynchronous data. React Query makes fetching, caching, synchronizing, and updating server states very easy.

useQuer is one of the hooks provided by React Query to fetch data. Under the hood, these hooks manage lots of other things like caching data after the initial fetch, re-fetching data in the background, etc. In this post, I will explain my first attempt and steps that I took to fetch data with useQuery hook using a demo.

First, I created a brand new React app using create react app and installed React Query.

```
npm i react-query
```

I used https://jsonplaceholder.typicode.com/ to fetch some mock data. In my App.js I cleared out the boilerplate code and wrote a fetch method that uses async-await and fetches some posts data from JSONPlaceholder.



```
const fetchPosts = async () => {
  const response = await fetch("https://jsonplaceholder.typicode.com/posts")
  return response.json()
}
```

FetchPosts function

Then, I had to import useQuery from react-query. This gives me the ability to manage fetching data using useQuery hooks and useQuery requires a key and a function that fetches and optional option objects. Similar to this:

```
const response = useQuery("string", fetchThat);
```

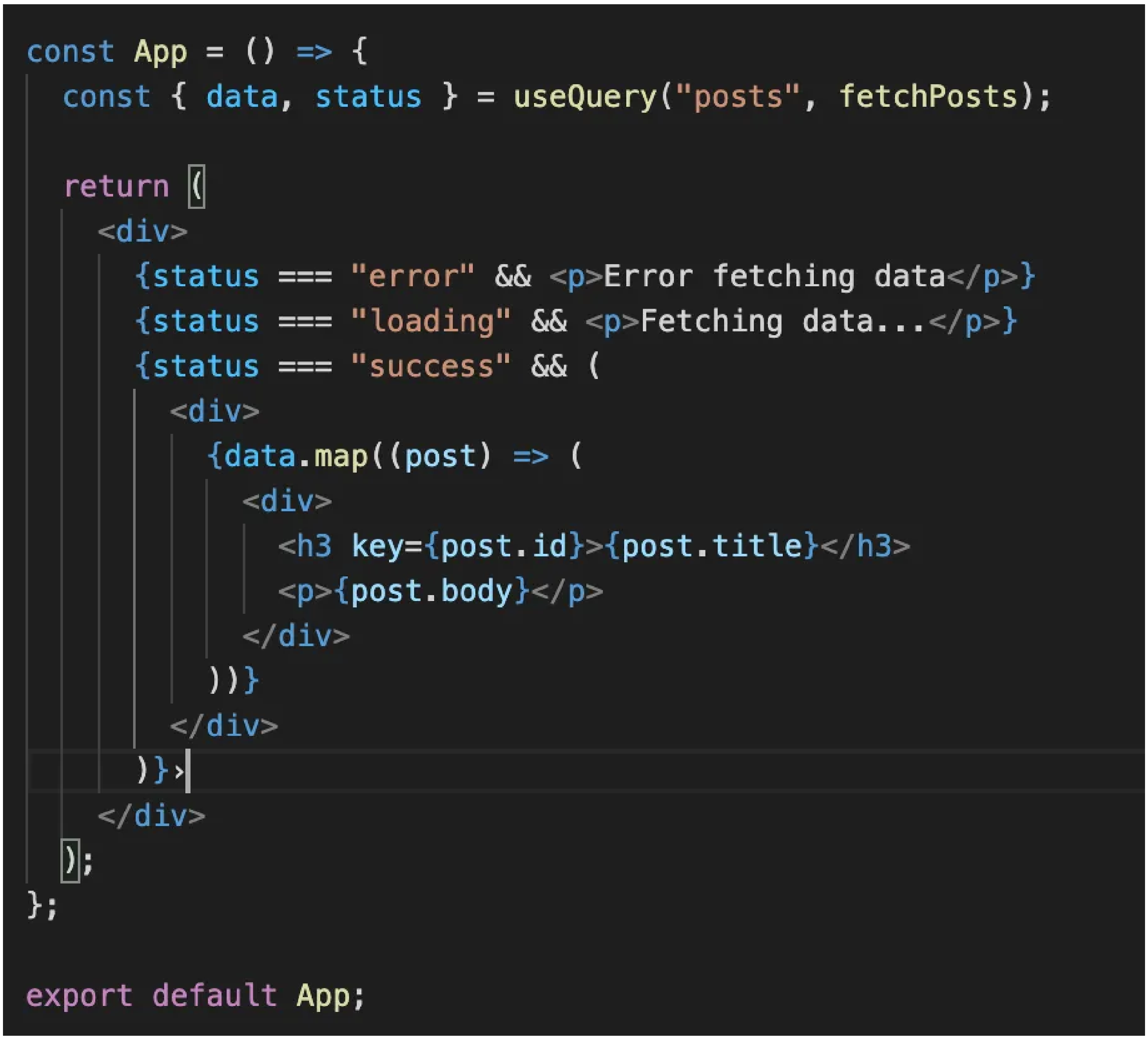
Here I am using a string as a key for the first argument, alternatively, I could use an array as well, If an Array is used, each item will be serialized into a stable query key. The second one is a function to fetch the data from the backend or a remote source. The response return from useQuery has the following properties:

```
data,
error,
failureCount,
isError,
isFetchedAfterMount,
isFetching,
isIdle,
isLoading,
isPreviousData,
isStale,
isSuccess,
refetch,
remove,
status,
```

I just used "data" and "status" where data is the data that has been fetched and status is the state which is: "loading", "error", "success" or "idle". Using destructuring, I extracted data and status as below:

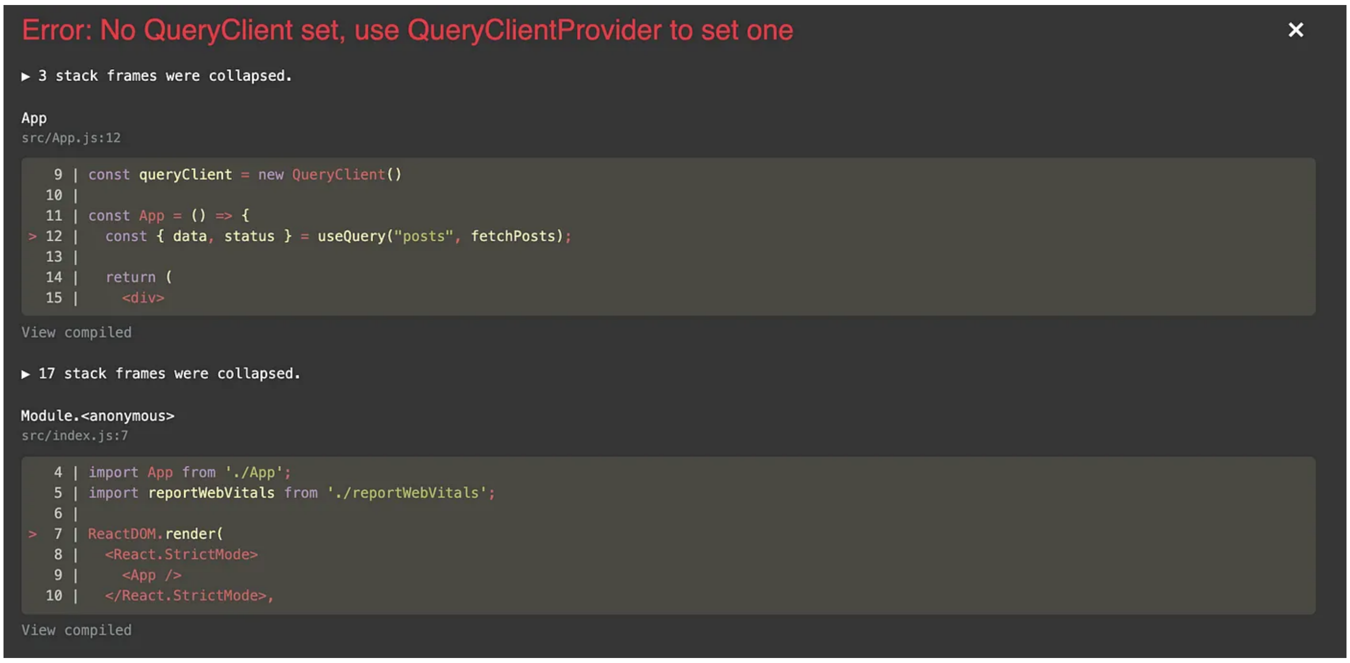
```
const { data, status } = useQuery("posts", fetchUsers);
```

Now I had the data, and was ready to display it:



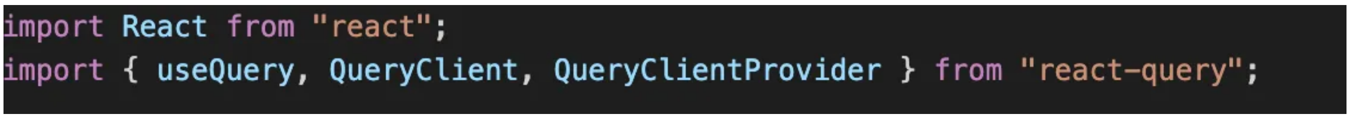
```
const App = () => {
  const { data, status } = useQuery("posts", fetchPosts);

  return (
    <div>
      {status === "error" && <p>Error fetching data</p>}
      {status === "loading" && <p>Fetching data...</p>}
      {status === "success" && (
        <div>
          {data.map((post) => (
            <div>
              <h3 key={post.id}>{post.title}</h3>
              <p>{post.body}</p>
            </div>
          ))}
        </div>
      )}
    </div>
  );
};

export default App;
```

This should've been working in the browser ut it was giving me this error in the browser:

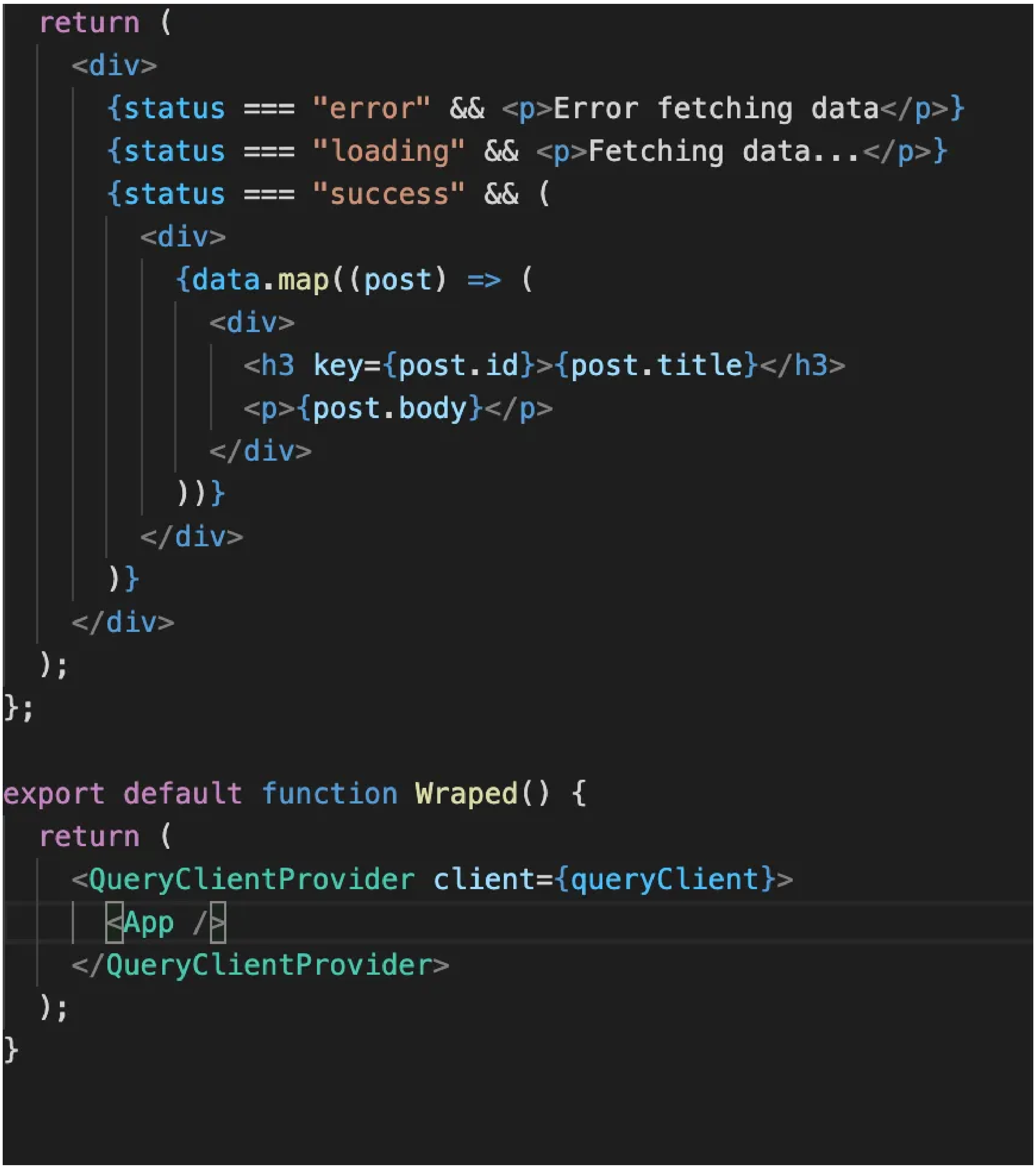


```
Error: No QueryClient set, use QueryClientProvider to set one
▶ 3 stack frames were collapsed.

App
src/App.js:12
   9 | const queryClient = new QueryClient()
  10 |
  11 | const App = () => {
> 12 |   const { data, status } = useQuery("posts", fetchPosts);
  13 |
  14 |   return (
  15 |     <div>
View compiled

▶ 17 stack frames were collapsed.

Module.<anonymous>
src/index.js:7
   4 | import App from './App';
   5 | import reportWebVitals from './reportWebVitals';
   6 |
>  7 | ReactDOM.render(
   8 |   <React.StrictMode>
   9 |     <App />
  10 |   </React.StrictMode>,
View compiled
```

After, reading the official docs for React Query I realized that I need to wrap the app component with a Query client. Approaching the Higher-Order Component approach this was quickly fixed like this:

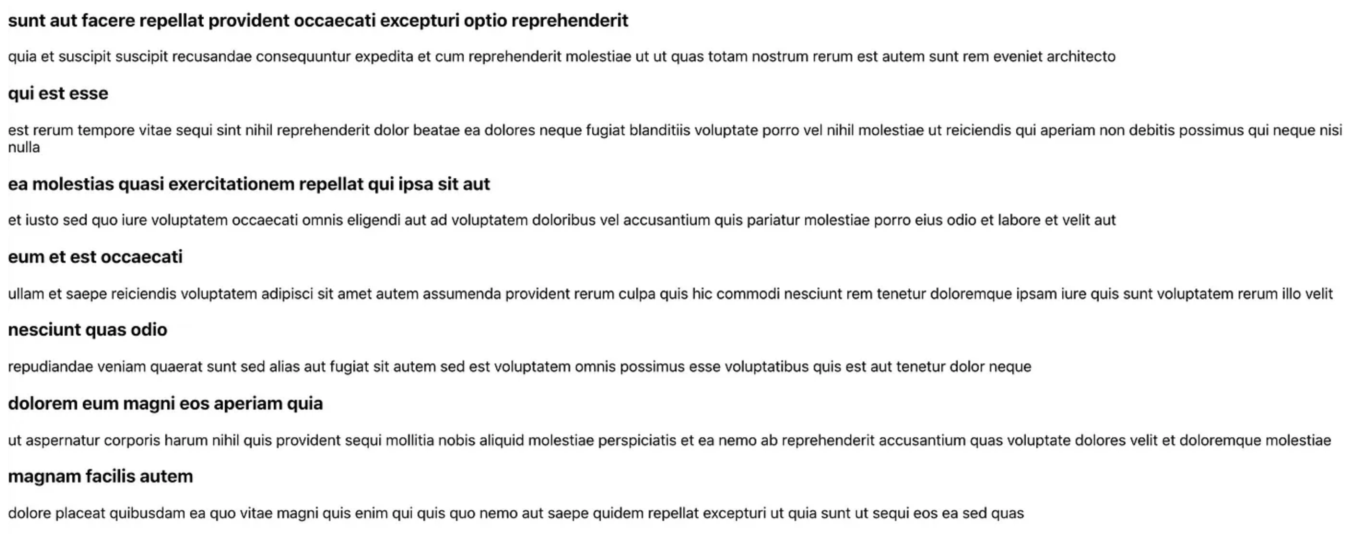


```
import React from "react";
import { useQuery, QueryClient, QueryClientProvider } from "react-query";
```



```
  return (
    <div>
      {status === "error" && <p>Error fetching data</p>}
      {status === "loading" && <p>Fetching data...</p>}
      {status === "success" && (
        <div>
          {data.map((post) => (
            <div>
              <h3 key={post.id}>{post.title}</h3>
              <p>{post.body}</p>
            </div>
          ))}
        </div>
      )}
    </div>
  );
};

export default function Wraped() {
  return (
    <QueryClientProvider client={queryClient}>
      <App />
    </QueryClientProvider>
  );
}
```

Essentially what I did is instead of exporting App regularly I wrapped it with QueryClient that was provided by React Query. Now the program was working fine with browser:


sunt aut facere repellat provident occaecati excepturi optio reprehenderit

quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto

qui est esse

est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla

ea molestias quasi exercitationem repellat qui ipsa sit aut

et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut

eum et est occaecati

ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit

nesciunt quas odio

repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque

dolorem eum magni eos aperiam quia

ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae

magnam facilis autem

dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas

This is how I got started using the useQuery and React Query. I would appreciate any comments or suggestions. Thank you for reading.

Follow

Written by Dhrubo Dh

8 Followers

More from Dhrubo Dh