**useFetch()**

* useFetch() is a custom hook in react which allow us to fetch data from server.
* Usefetch() is inside react-fecth-hook library, which we have to download it first.
* Instal = **npm i react-fetch-hook**

**Example:**

import React from 'react'

import useFetch from 'react-fetch-hook'

export default function UsefetchHook() {

  const {data, isLoading, error}

=  useFetch('https://jsonplaceholder.typicode.com/posts')

  if(error){

      return <div> Error : {error.message}</div>

  }else if (isLoading){

      return <div>Loading.....</div>

  }else{

      return(

          <ul>

              {data.map((item) =>(

                  <li key={item.id}>

                      {item.title}

                  </li>

              ))}

          </ul>

      )

  }

}

* Here, from above example we have data, error & isLoading properties from usefetch()
* By default isloading value is true. Data is for response data which we will get the response from the particular server.
* And error is for any kind of interruption while fetching is data.
* useFetch() is pretty simple to use.

**useQuery()**

* useQuery() is another react library or we can say custom hook which helps us to fetch the data from the server.
* Big difference between usefetch() & useQuery() is, useQuery() stores the fetch data in cache and not hit the server for unnecessary changes for which application performance will increase.
* useQuery(‘key’, callback function which include fetch or axios method).
* Install = npm install react-query-hook
* To use useQuery() we need QueryClient() & QueryClientProvider.
* QueryClientProvider is wrap the main component and pass the client attribute as QueryClient() instance.

Example:

import React from 'react'

import { QueryClient, QueryClientProvider, useQuery } from 'react-query'

const queryClient = new QueryClient()

const Example = () =>{

   const {data,isLoading,error} =   useQuery('repoData', () => fetch('https://jsonplaceholder.typicode.com/posts').then(res => res.json()))

   if (error) {

       return <div>An error has occured {error.message}</div>

   }

   else if (isLoading){

       return <div>Loading.....</div>

   }

   else{

       return (

           <div>

               <ul>

                   {data.map((item) =>(

                       <li key={item.id}>

                           {item.title}

                       </li>

                   ))}

               </ul>

           </div>

       )

   }

}

function UseQueryHook() {

  return (

    <div>

          <QueryClientProvider client={queryClient}>

              <Example/>

          </QueryClientProvider>

    </div>

  )

}

export default UseQueryHook