React Query is a popular library for managing and caching remote data in React applications. React Query's cache is an important feature that enables the library to optimize data fetching and reduce unnecessary network requests.

The cache in React Query is an in-memory cache that stores data fetched from remote sources, such as REST APIs or GraphQL endpoints. When a query is executed for the first time, React Query checks if the data for that query is already in the cache. If the data is present in the cache and is not stale, React Query returns the cached data instead of making a new network request.

The cache in React Query is customizable and allows you to configure settings such as cache time-to-live (TTL), cache key serialization, and cache eviction policies. You can also use the cache to store intermediate query results or to store user-generated data that is not fetched from a remote source.

Overall, React Query's cache provides a powerful mechanism for optimizing data fetching and improving the performance of your React applications.

Default time is 5 min.

Syntax :

 const{isLoading, data,isError, error}=useQuery('rqsuper-heroes',fetchData,

  {

    cacheTime:5000,

  })

Code :

import React from 'react'

import { useQuery } from 'react-query'

import axios from 'axios'

const fetchData=()=>{

  return axios.get('http://localhost:4000/superheroes')

}

export const RQSuperHeroesPage = () => {

  //Step 1 : Increase parameter to hold other valyer

  const{isLoading, data,isError, error}=useQuery('rqsuper-heroes',fetchData,

  {

    cacheTime:5000,

  })

    if(isLoading){

      return <h2>Loading......</h2>

    }

// step 2 : Add condition to check error or not

    if(isError){

      return <h2>{error.message}</h2>

    }

      return (

        <div>

          RQSuperHero Page

          {data?.data.map((hero)=>{

            return <div key={hero.name}>{hero.name}</div>

          })}

        </div>

      )

    }

