# TANSCTAK/React query

Installation

### 2. TanStack React Query installations

TanStack React Query is used for data fetching, caching, and synchronization in React applications.

bash

Copy code

**npm install @tanstack/react-query**

# basic setup after tanstack/react query

Here are the installation commands for the commonly used React packages you mentioned, along with their brief descriptions:

**1. React Router DOM**

React Router DOM is used for client-side navigation in React applications.

bash

Copy code

npm install react-router-dom

**2. TanStack React Query**

TanStack React Query is used for data fetching, caching, and synchronization in React applications.

bash

Copy code

**npm install @tanstack/react-query**

**3. React Hook Form**

React Hook Form is a library for building forms in React with easy validation and improved performance.

bash

Copy code

npm install react-hook-form

**4. Axios (Optional)**

For making HTTP requests, you can use Axios.

bash

Copy code

npm install axios

**5. Zustand (Optional)**

For state management in React applications.

bash

Copy code

npm install zustand

**Dev Server**

If you are using Vite (which you are), after installing these packages, you can run your application using:

bash

Copy code

npm run dev

**Additional Commands**

* **Installing all at once**:

bash

Copy code

npm install react-router-dom @tanstack/react-query react-hook-form axios zustand

Let me know if you need setup instructions or example usage for any of these libraries!

# Steps-to work with tansatck query

Step -1 **npm install @tanstack/react-query**

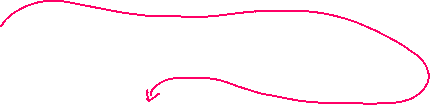
Step-2 goto app.js or app.jsx or main.jsx and wrap with queryclientprocider referenced with queryclient

First import both

For vite users its main.jsx and plain react its index.jsx

Im gonna use main.jsx

import { StrictMode } from 'react'  
import { createRoot } from 'react-dom/client'  
import {QueryClientProvider,QueryClient,} from "@tanstack/react-query";  
import './index.css'  
import App from './App.jsx'  
  
  
const queryclient=new QueryClient();  
  
createRoot(*document*.getElementById('root')).render(  
  
 <StrictMode>  
<QueryClientProvider client={queryclient}>  
 <App />  
</QueryClientProvider>  
 </StrictMode>  
)







Now to start using it on any component and start working with apis

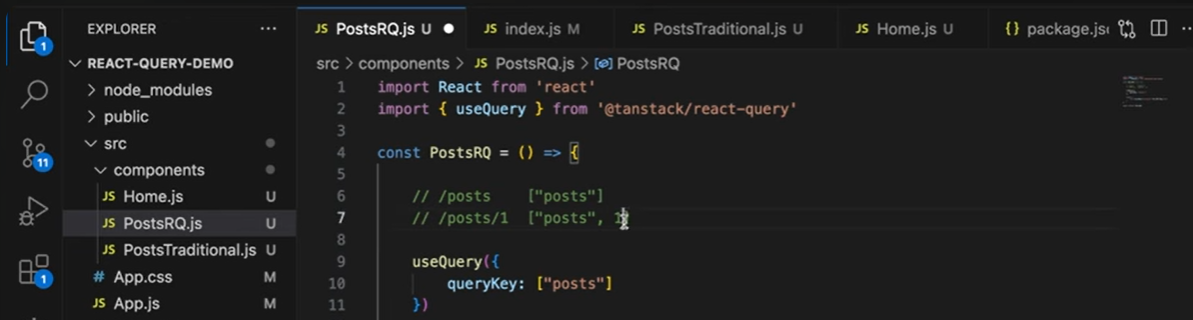




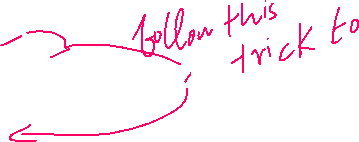
# Using React query to fetch data from apis

We need query key and queryfn

Query key i.e unique key







Now example code:

Steps :fetch the data using axios

Step 2:thn destructure usequery and use querykey and queryfn(function ref)

Querykey if no params or stats are added just the routename,if getby id or similar means use two key,one is routename and id

This indicates thn whenever id is added the refetching happens

The use of key is to keep usequery updated on any change

import React, { useState } from 'react';  
import { useQuery } from "@tanstack/react-query";  
import *axios* from "axios";  
import Card from "./Card.jsx";  
import Button from "./Button.jsx";  
import {NavLink} from "react-router-dom";  
  
const Posts = () => {  
 const [singleProduct, setSingleProduct] = useState(0);  
*console*.log(singleProduct);  
 // Fetch posts based on the selected product  
 const fetchPosts = async () => {  
 const url = singleProduct !== 0  
 ? `http://localhost:4000/posts/${singleProduct}` // Single product URL  
 : 'http://localhost:4000/posts'; // All products URL  
 const response = await axios.get(url);  
 return response.data;  
 };  
  
 // Fetch data using React Query  
 **const { data, isLoading, isError, error } = useQuery({  
 queryKey: ["posts", singleProduct],//if you are trying to fetch single product then also metion that state  
 queryFn: fetchPosts  
 });**  
  
 // Loading and Error handling  
 if (isLoading) {  
 return <h1>Loading...</h1>;  
 }  
  
 if (isError) {  
 return <h1>{error.message}</h1>;  
 }  
  
  
  
 return (  
 <div className="flex flex-wrap w-screen h-screen justify-center relative">  
 <div className="flex flex-col left-0 top-8 gap-2 fixed">  
 {[0,1, 2, 3, 4, 5].map((categ, index) => (  
 <Button key={index} cate={categ} onClick={() => {  
 *console*.log("Button clicked:", categ); // Check if button click is working  
 setSingleProduct(categ); // Set selected product ID  
 }} />  
 ))}  
 </div>  
  
 {/\* Render data based on whether it is an array or a single object \*/}  
 {*Array*.isArray(data) ? (  
 data.length > 0 ? (  
 data.map((post) => <NavLink key={post.id} to={`/productDetails/${post.id}`} ><Card key={post.id} {...post} /></NavLink>)  
 ) : (  
 <h1>No products found</h1>  
 )  
 ) : (  
 data ? <Card key={data.id} {...data} /> : <h1>No product found</h1>  
 )}  
 </div>  
 );  
};  
  
export default Posts;

2nd example

import React from 'react'  
import {useParams} from "react-router-dom";  
import {useQuery} from "@tanstack/react-query";  
import *axios* from "axios";  
  
const ProductDetailPage = () => {  
 const {id}=useParams() //extract id from url  
  
 const fetchproductdata=async()=>{  
 const response=await axios.get(`http://localhost:4000/posts/${id}`);  
 return response.data;  
 }  
  
 const {data,isError,isLoading,error}=useQuery({  
 queryKey:["productId",id],  
 queryFn: fetchproductdata  
 });  
  
 if(isLoading){  
 return <h1>Loading...</h1>  
 }else if(isError){  
 return <h1>{error.message}</h1>  
 }  
  
  
 return (  
 <div className="flex p-4 w-full h-screen items-center justify-center">  
  
 <div className="flex w-full h-[18rem] items-center justify-center box-border m-0 p-0">  
 <div  
 className="text-sm w-[14rem] text-center flex flex-col h-full justify-center items-center border-2 p-2">  
 <h1>{data.title}</h1>  
 <img className="h-[14rem] w-full" src={data.image} alt={data.title}/>  
 </div>  
 <div className="flex flex-col bg-red-100 gap-3 h-fit w-[20rem] items-center border-4 rounded-md p-2">  
 <h2 className="font-poppins first-letter:uppercase font-semibold">{data.*category*}</h2>  
 <p className="font-poppins font-light text-sm">{data.description}</p>  
 </div>  
 </div>  
  
 </div>  
 )  
}  
export default ProductDetailPage

# DEVTOOLS

# Query on click

We need to destructure refetch,and add enabled:false

Now just call thet refetch fuction in onclick event

Fetch the data as usual

Now in useQuery ,destructure the **refetch** function

And set enables:false

Now in the button onclick event call the refetch function or its reference

import *axios* from "axios";  
import {useQuery} from "@tanstack/react-query";  
import {NavLink} from "react-router-dom";  
import Card from "./Card.jsx";  
import {useState} from "react";  
  
const FetchOnClick = () => {  
  
  
 // Fetch posts based on the selected product  
 const fetchPosts = async () => {  
 const url = 'http://localhost:4000/posts'; // All products URL  
 const response = await axios.get(url);  
 return response.data;  
 };  
  
 // Fetch data using React Query  
 const { data, isLoading, isError, error,**refetch** } = useQuery({  
 queryKey: ["posts"],//if you are trying to fetch single product then also metion that state  
 queryFn: fetchPosts,  
 **enabled:false  
 });**  
  
 // Loading and Error handling  
 if (isLoading) {  
 return <h1>Loading...</h1>;  
 }  
  
 if (isError) {  
 return <h1>{error.message}</h1>;  
 }  
  
//what if i wanna toggle display on click and not display on click  
 const [display, setDisplay] = useState(true);  
  
 return (  
 <div className="flex flex-wrap w-screen h-screen justify-center">  
 <button className="bg-slate-800 p-3 w-fit h-fit font-poppins font-[900] text-slate-300" **onClick={refetch**}>fetch post</button> {/\*//directly call the reftech reference\*/}  
 { data?.map((post) => <NavLink key={post.id} to={`/productDetails/${post.id}`} ><Card key={post.id} {...post} /></NavLink>)  
 }  
 </div>  
 );  
}  
export default FetchOnClick

toggling using function to toggle and using refetch function,inside of handlediplay

import *axios* from "axios";  
import {useQuery} from "@tanstack/react-query";  
import {NavLink} from "react-router-dom";  
import Card from "./Card.jsx";  
import {useState} from "react";  
  
const FetchOnClick = () => {  
  
  
 // Fetch posts based on the selected product  
 const fetchPosts = async () => {  
 const url = 'http://localhost:4000/posts'; // All products URL  
 const response = await axios.get(url);  
 return response.data;  
 };  
  
 // Fetch data using React Query  
 const { data, isLoading, isError, error,refetch ,isFetching} = useQuery({  
 queryKey: ["posts"],//if you are trying to fetch single product then also metion that state  
 queryFn: fetchPosts,  
 enabled:false  
 });  
  
 // Loading and Error handling  
 if (isLoading) {  
 return <h1>Loading...</h1>;  
 }  
  
 if (isError) {  
 return <h1>{error.message}</h1>;  
 }  
  
//what if i wanna toggle display on click and not display on click  
 const [display, setDisplay] = useState(false);  
 const handleToggle=(e)=>{  
 setDisplay(!display);  
 refetch();  
 }  
  
 //when you use state here outside of return statement it wont work  
 //use the below logic state for set button in return statement..  
 //use state get updated in return statement  
  
  
  
 return (  
  
 <div className="flex flex-wrap w-screen h-screen justify-center">  
 <button className="bg-slate-800 p-3 w-fit h-fit font-poppins font-[900] text-slate-300" onClick={handleToggle}>{display?"hide data":"fetch data"}</button> {/\*//directly call the reftech reference\*/}  
 { display && data?.map((post) => <NavLink key={post.id} to={`/productDetails/${post.id}`} ><Card key={post.id} {...post} /></NavLink>)  
 }  
 </div>  
 );  
}  
export default FetchOnClick

# Polling or refetching

Use this only for fetching data

The properties refetchInterval and refetchIntervalInBackground in React Query are used to periodically refetch data in the background. Here’s a breakdown of each:

1. **refetchInterval**

* **Purpose**: Sets an interval for how often a query should be refetched automatically. In this case, refetchInterval: 1000 means the query will be refetched every 1 second (1000 ms).
* **Usage**: Useful for real-time data updates, like live stock prices, dashboards, or notifications.
* **Syntax**:

javascript

Copy code

refetchInterval: 1000 // Refetches data every second

1. **refetchIntervalInBackground**

* **Purpose**: Determines whether the query should continue to refetch even when the app is in the background (e.g., when the browser tab is not active).
* **Usage**: Set to true if you want the app to keep fetching in the background, which is helpful if the data needs to stay up-to-date continuously. However, it can consume more bandwidth and battery, so it’s typically used selectively.
* **Syntax**:

javascript

Copy code

refetchIntervalInBackground: true // Keeps refetching even in the background

1. **Example Usage in a Query**

javascript

Copy code

const { data, error, isLoading } = useQuery({

queryKey: ['posts'],

queryFn: fetchPosts,

refetchInterval: 1000, // Fetches data every second

refetchIntervalInBackground: true, // Fetches even in the background

});

This configuration ensures that the data is refreshed every second, regardless of whether the user actively views the tab or app.

# USEMUTATION hooks for post,put,del

# Post Mapping or adding the data api

Donot use use state and thn use it to mutate the posting data,

Since usestate works asynchronously it doesn’t pic objects immediately..as it takes time to get updated

So directly insert form Data to mutate(formdata) to trigger the post mapping

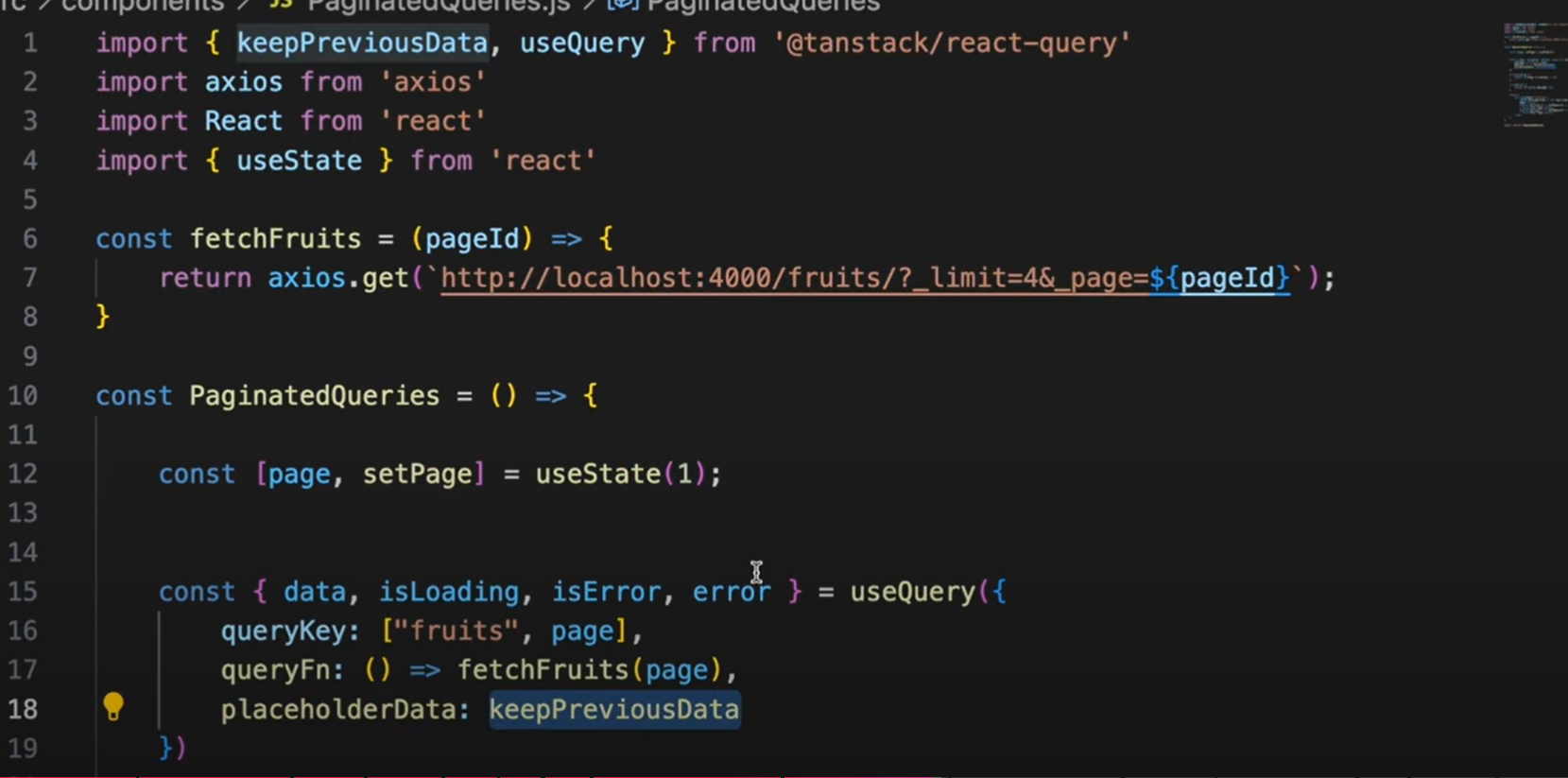
Step1:import refrence of queryClient use its hook useQueryClient

Step2:u need this to invlaidateQuery(querykey) which take query key to trigger the get mapping onve data is posted use correct key

import React, {useState} from 'react'  
import {useForm} from "react-hook-form";  
import {useMutation,useQueryClient} from "@tanstack/react-query";  
import *axios* from "axios";  
  
const AddDetails = () => {  
 //no need if on change event if you are using react hoook form  
 //register is also and object which keep appending details once automatically,so use {...register()} with inputs spread operator  
 const {handleSubmit,register,reset}=useForm()  
 const [posts, setPosts] = useState({});  
 const handleSubmitData=(productdata)=>{  
 *console*.log(productdata)  
 setPosts(prev=>prev=productdata);//this usstate works Asynchronously so might be empty  
 *console*.log(posts) //donot use state in such scinarios  
 //here call the mutate funtion mutate() or its alias name if you ahev given any ,i have given  
 postdata(productdata);//this will pic parameters,pass form data directly not with state  
 reset();  
 }  
 //for post update del you need query client obj  
 const queryClient = useQueryClient();  
 //now call the api with seperate fuction to make it modular  
 const addPosts=async(productdata)=>{  
 const add=await axios.post("http://localhost:4000/posts",productdata)  
 }  
 //use mutations  
 //give alias name for mutate if you want  
 //next call the mutate function  
 const {mutate:postdata,isSuccess,isError,error}=useMutation({  
 //remeber to call mutate function in hadnle submit to trigger post api  
 mutationFn:addPosts,  
 onSuccess: () => {  
 queryClient.invalidateQueries(["posts"]); // Matches default state in `Posts`  
 },  
 });  
  
 return (  
 <div className="bg-slate-800">  
 <form className="flex flex-col"  
 onSubmit={handleSubmit(handleSubmitData)}> {/\*//the data will be automatically passed by react hook for the fuction defined in handle submit\*/}  
 <input placeholder="Enter title" {...register('title')} type="text"/>  
 <input placeholder="Enter price" {...register('price')} type="text"/>  
 <input placeholder="Enter decription" {...register('description')} type="text"/>  
 <input placeholder="Enter category" {...register('category')} type="text"/>  
 <input placeholder="Enter imageUrl" {...register('image')} type="text"/>  
 <button type="submit" className="bg-blue-500 text-white py-2 px-4 rounded">  
 Submit  
 </button>  
 </form>  
 </div>  
  
 )  
}  
export default AddDetails

# PAGINATION

Keep previous data will keep the previous data and prevent reload when we go back



Just add buttons and change the page usestate value increment or decrement

