React Query

## All practice is done in project 18. Wild Oasis App

# What is React Query?

* A powerful **State Manager** for **REMOTE (server)** state
* Many features that allow us to write **less code**

# React Query Fundamentals

* All data is stored in cache
* Automatic loading and error States
* Automatic re-fetching to keep state synched
* Pre-fetching data
  + Pagination
* Easy remote state mutation (updating)
* Offline support
  + Since data is already cached
  + Components can still use this cached data

# Install and Setup

* Npm I @tanstack/react-query
* Npm I @tanstack/eslint-plugin-query
  + Eslint plugin for react query
* The idea of integrating with the app is similar to the **Context API** 
  + We will wrap the whole app at the top level in a **Provider**

### App integration:

1. New QueryClient()
2. We Wrap the application
   1. <QueryProvider client={queryClient}>
      1. APP
   2. </QueryProvider>

### QueryClient() Options

* We can provide **Options** to our queryClient
* New QueryClient({defaultOptions: {
  + staleTime
    - how much the data will stay stale until is **re-fetched**
    - staleTime: 60 \* 1000 ( 1 minute )

# React Query DevTools

* Just a simple package , not in Browser like Redux Dev Tools
  + npm i @tanstack/react-query-devtools
* There are multiple ways to use the DevTools
  + Floating Mode
    - This will place a floating element in our app so we can access the DevTools
    - Place it as close to the Top of the application
    - <ReactQueryDevtools initialIsOpen={false}/>
  + Embedded mode
    - This will show the dev tools as a fixed element in the app
    - <button
    - onClick={() => setIsOpen(!isOpen)}
    - >{`${isOpen ? 'Close' : 'Open'} the devtools panel`}</button>
    - {isOpen && <ReactQueryDevtoolsPanel onClose={() => setIsOpen(false)} />}

# Fetching the data from supabase

* **useQuery**  - custom hook
  + queryKey : [‘cabin’]
    - unique key to identify data
  + queryFn: getCabins
    - the function to return the data
    - we already have the function that **RETURNs A PROMISE**

A screen shot of a computer program

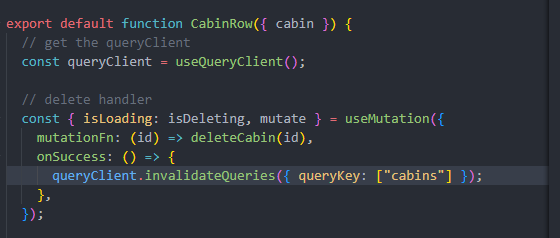
Description automatically generated

# Mutate the Remote Server State

* We will use React Query to delete an entry from the Supabase DB
* Mutations are used to perform **create/update/delete** data
* We will use **useMutation** hook
* const { isLoading, mutate } = useMutation({
* mutationFn: (id) => deleteCabin(id),
* });

WE will delete based on the **id** of the cabin

* To update the UI, we will **invalidate the cache** as soon as the **MUTATION is done**
  + We use the **onSuccess** handler of the **useMutation hook**
  + We get the **queryClient** using the **useQueryClient hook**



* We also have the **onError** handler, that receives the Error