

\* TonHACK query  
 JS Library for managing Server-Side State  
 in react

- Axios:

It is a promise-based HTTP client for JS which is used to make HTTP ~~request~~ request to fetch / send data to a server.

- QueryClientProvider:

It is a crucial component that provides a QueryClient instance to your react application.

- QueryClient:

It is responsible for managing all the data fetching, caching & state management related to our query. Track which query are active.

```
const queryClient = new QueryClient();
```

```
<QueryClientProvider client={queryClient}>
```

```
  :
```

```
<QueryClientProvider>
```

- useQuery:

It fetches data from an API / Async func. It automatically caches the result. It handles loading / error states for you. It can automatically re-fetch data on window focus or on interval. It keeps your UI in Sync with your server.



```
const {data, isLoading, isError, error} =
  useQuery(['key'], fetcher);
```

### • queryKey:

It is typically an array or string that uniquely identifies a query. It allows ReactQuery to determine if the data in the cache is associated with a particular req.

It is used to cache data with a specific key & refetch / update data when certain dependencies change.

### • queryFn:

It is the fun. that actually fetches your data.

```
const {data} = useQuery({
  queryKey: ["posts"], // it stores data in a
  queryFn: getpostsData, // fun that defines fetch
  // method
});
```

### • useMutation:

Used for CRUD operation & allows triggering manual side effects.

### • gcTime (garbage collection Time):

CacheTime option in ReactQuery has been renamed to gcTime.

By default, inactive queries are garbage collected after 5 min.



- StaleTime:

It is a Configuration Option that determines how long fetched data is considered fresh before it needs to be refreshed.

The default StaleTime is 0, meaning data becomes stale immediately after being fetched.

- Polling:

It refers to the technique of fetching data from an API at regular intervals to keep the UI up-to-date with latest info.

'refetchInterval' option

'refetchIntervalBackground' Option

- useMutation:

```
const mutation = useMutation( mutationFn,  
  // optional configuration  
  { } );
```

Options : onSuccess

onSettled

onError

mutationKey

- mutate():

It is used to execute the mutation in React Query.

When you call `mutate()`, it tells React Query to run the mutation fun. defined inside the `useMutation` hook.



- queryClient.setQueryData:

It is used to update the cached data for a specific query.

// demo testing fun.  
In this case, it's the query with the key ["post", pageNumber], which likely represents the list of posts on the current page.

- useInfiniteQuery:

