React query

React Query (now known as TanStack Query) is a powerful data-fetching and state management library for React applications. It simplifies the process of fetching, caching, synchronizing, and updating server state in your UI.

React Query is primarily used to manage server state—data fetched from APIs or databases. Unlike local state (managed with useState or useReducer), server state is asynchronous, shared across components, and needs to be kept in sync with the backend.

Automatic Caching, Background Data Synchronization, Simplified Code, Mutation Handling, DevTools, Improved Performance

https://github.com/vinaycoder/react-docs/edit/master/react-query/reactQuery.js

```
useQueries({ queries: queryArray, ...options });

useQuery({ ...queryObj, ...options });

useMutation({
   mutationFn: (payLoad) => fetchApi(url, { ...config, body: payLoad }),
   ...options,
 });


 const { data, isLoading, error } = useQuery({
   queryKey: ['todos'],
   queryFn: () => fetch('/api/todos').then(res => res.json())
 });
```

1. **how to create a structure of react query**

in App.js

```
rootElement.render(
<AppContextProvider {...props}>
<ReactQueryProvider>
<YourContainerComponent />
</ReactQueryProvider>
</AppContextProvider>,
);
```

create a ReactQueryProvider

```
import React from 'react';
import PropTypes from 'prop-types';
import { QueryClientProvider, QueryClient } from '@tanstack/react-query';

const queryClient = new QueryClient({
defaultOptions: {
queries: {
retry: false,
},
},
});
const ReactQueryProvider = ({ children }) => (
<QueryClientProvider client={queryClient}>{children}</QueryClientProvider>
);

ReactQueryProvider.propTypes = {
children: PropTypes.node,
};

export default ReactQueryProvider;
```

create a function that use payload get api call in service

```
import { groupOptions, useReactQueries, useReactQueryMutation } from './reactQuery';
```

```
export const pageLoadApi = (options) => useReactQueries({
queryArray: [
groupOptions(apiURl),
groupOptions(anotherApiUrl),
groupOptions(apiURl, { staleTime: 0, cacheTime: 0, config }),
],
options: { ...options },
});
```

Now this function we can use in component or create a hook so this function will give the result

```
import { pageLoadApi } from '../services';
const combine = (res) => { // combine function give the api reponse
}
const results = pageLoadApi({ combine }) || {};
```

In post call you can use below functions

```
const useReactQueryMutation = ({ url, config = {}, options = {} }) => { // own function name
if (!url) return {};

return useMutation({   // react query retrun this
mutationFn: (payLoad) => fetchApi(url, { ...config, body: payLoad }),
...options,
});
};
```