# React query

## 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,
});
};
```