1. Install **@apollo/client** using the following command:

**npm install @apollo/client**

It is used to connect to the graphql server, cache data in case of re-rendering and finally to write queries to get data.

1. Typically following classes are imported from @apollo/client package:

* **ApolloClient:** Used for connecting client with graphql server.
* **InMemoryCache:** Used to cache data in case when the component re-renders.
* **ApolloProvider:** Is a state management which wraps the highest-level component so that it could be accessed by any component.
* **useQuery:** To pass queries so that an API call to the graphql server is made.
* **useLazyQuery:** Used to make API call on demand.
* **gql:** Converts the contained string into graphql equivalent query.

1. We need to instantiate an object of ApolloClient class.

const client = new ApolloClient({

cache: new InMemoryCache(),

uri: "http://localhost:4000/graphql"

})

The constructor takes an object with properties:

* **cache:** how to cache data in case of re-rendering
* **uri:** path of graphql server

**Note:** uri will end with /graphql since the graphql server always serves the data on /graphql.

1. Wrap the highest-level component with ApolloClient and pass client as a prop to ApolloClient.

<ApolloProvider client={client}>

…….

</ApolloProvider>

1. We can now make api calls using our graphql client. For that we should import **useQuery** and **gql**.
2. We can write the query as:

const QUERY\_LIST\_OF\_USERS = gql`

query GetAllUsers {

users {

id

name

username

age

}

}

`;

**Note:** According to graphql standard, variable name of the graphql query should be in caps and space should be replaced with underscore.

1. When we pass the query to useQuery hook, it returns the result with few other values, like loading (Boolean flag), error etc.

const { data, loading, error } = useQuery(QUERY\_LIST\_OF\_USERS);

1. We can make use of these values for conditional rendering.
2. useLazyQuery hook can be used to fetch data on demand along with few other fields like error, loading, data etc.

const GET\_MOVIE\_DATA = gql`

query GetMovieByName($name: String!){

movie(name: $name) {

name

yearOfRelease

}

}

`;

Above query would be passed as a graphql query to useLazyQuery hook.

const [fetchMovie, {

data: movieSearchData,

}] = useLazyQuery(GET\_MOVIE\_DATA);

1. useLazyQuery returns following values:

* function to be called
* data, loading, error after the function is called.

Here, ***fetchMovie*** function would be used to make the graphql call. The second value returned by useLazyQuery would be data, loading, error, etc.

To pass the value to ***fetchMovie*** function, we need to pass a object with property as ***variables***.

<button onClick={() => fetchMovie({

variables: {

name: movieSearched

}

})}>

Submit

</button>