Executing a GraphQL Query Using Apollo Client

In this lesson, you will learn how to send a query to GitHub's GraphQL API using Apollo Client.

WE'LL COVER THE FOLLOWING

- Exercises
- Reading Tasks

Now we are going to send our first query to GitHub's GraphQL API using Apollo Client. First of all, we will import the following utility from Apollo Boost to define the query:

Environment Variables

Key:	Value:
REACT_APP_GITHUB...	Not Specified...
GITHUB_PERSONAL...	Not Specified...

```
import 'cross-fetch/polyfill';
import ApolloClient, { gql } from 'apollo-boost';
```

index.js

Next, we will define our query with JavaScript template literals:

Environment Variables

Key:	Value:
REACT_APP_GITHUB...	Not Specified...
GITHUB_PERSONAL...	Not Specified...

```
...
const GET_ORGANIZATION = gql`
  {
    organization(login: "the-road-to-learn-react") {
      name
```

```
        url
      }


    }
`;
```

index.js

Now, we use the Apollo Client imperatively to send the query to GitHub's GraphQL API. Since the Apollo Client is promise-based, the `query()` method returns a promise that you can eventually resolve. It's sufficient to console log the result because we are running the application in the command line for now.

Environment Variables	
Key:	Value:
REACT_APP_GITHUB...	Not Specified...
GITHUB_PERSONAL...	Not Specified...

```
...
client
  .query({
    query: GET_ORGANIZATION,
  })
  .then(console.log);
```

index.js

That's all there is on sending a query with the Apollo Client. As noted, Apollo Client uses HTTP under the hood to send the defined query as payload in the POST method. The output after running the application should look similar to the following:

Environment Variables	
Key:	Value:
REACT_APP_GITHUB...	Not Specified...
GITHUB_PERSONAL...	Not Specified...

```
{
  data: {
    organization: {
      name: 'The Road to learn React',
      url: 'https://github.com/the-road-to-learn-react',
      __typename: 'Organization'
    }
  },
  loading: false,

  networkStatus: 7,
  stale: false
}
```

The requested information from the GraphQL query can be found in the `data` object. There, we will find the `organization` object with its `name` and `url` fields. The Apollo Client automatically requests the GraphQL meta field `__typename`. The meta field can be used by the Apollo Client as an identifier, to allow caching and optimistic UI updates which we'll cover in the last chapter.

More meta-information about the request can be found next to the `data` object. It shows whether the data is still loading or whether the requested data is stale on the server-side and also specific details about the network status.

Let's run the code to see the result of our query:

Environment Variables

Key:	Value:
REACT_APP_GITHUB...	Not Specified...
GITHUB_PERSONAL...	Not Specified...

index.js

.ed_required_keys

.ed_keys

```
import 'dotenv/config';
import 'cross-fetch/polyfill';
import ApolloClient, { gql } from 'apollo-boost';

const client = new ApolloClient({
  uri: 'https://api.github.com/graphql',
  request: operation => {
    operation.setContext({
      headers: {
        authorization: `Bearer ==GITHUB_PERSONAL_ACCESS_TOKEN==`,
      },
    });
  },
});

const GET_ORGANIZATION = gql`
  {
    organization(login: "the-road-to-learn-react") {
      name
      url
    }

  }
`;

client
  .query({
    query: GET_ORGANIZATION,
  })
  .then(console.log);
```

Exercises

1. Confirm your source code for the last section
2. Explore GitHub's GraphQL API
   - Get comfortable navigating through their documentation
   - Add other fields for the `organization` field

Reading Tasks

1. Read more about why you should use Apollo Client
2. Read more about the network status property and its possible values