Starting with Apollo Boost

This lesson introduces to Apollo Client with Apollo Boost which we will use to execute GraphQL queries.

WE'LL COVER THE FOLLOWING

The application in this chapter will start from introducing **Apollo Client** with **Apollo Boost**. The latter allows you to create a zero-configuration Apollo Client to get you started with the fastest and most convenient way.

This chapter focuses on the Apollo Client instead of React for the sake of learning.

In the upcoming lessons, you will consume GitHub's GraphQL API, and then output the queries and mutation results in the command line. To do this, you will need a personal access token on GitHub's website, which we covered in one of the previous chapters.

Apollo-Boost Package

The apollo-boost package gives access to a zero-configuration Apollo Client, and the graphql package allows GraphQL queries, mutations, and subscriptions on both the client and server. It is JavaScript's reference implementation of Facebook's GraphQL specification.

In the next steps, you will configure and use the Apollo Client that comes with Apollo Boost in the `index.js` file of the project. The project stays small, and you will only implement it in this section, so, for now, we can have everything in one file for the sake of learning.

In your `index.js` file, we import the Apollo Client from Apollo Boost. After that, we create a client instance by calling its constructor with a URI. The client needs to know where the data comes from, and where it should be written, so you can pass GitHub's API endpoint to it.

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

```
import ApolloClient from 'apollo-boost';
const client = new ApolloClient({
  uri: 'https://api.github.com/graphql',
});
```

index.js

Configuring your Personal Access Token

The Apollo Client already works this way. Remember, however, that GitHub's GraphQL API requires a personal access token. That's why we have to define it once when creating the Apollo Client instance. Therefore, you can use the `request` property to define a function which has access to the context of each request made through the Apollo Client. There, we pass the authorization header using Apollo Boost as one of its default headers.

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

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

index.js

We did the same for the previous application in the previous chapter, using only axios for plain HTTP requests. We configured axios once with the GraphQL API endpoint to default all requests to this URI, and to set up the authorization header. The same happened here because it's enough to configure your client once for all the following GraphQL requests.

Remember, we have to replace the `YOUR_GITHUB_PERSONAL_ACCESS_TOKEN` string with our personal access token that you created on GitHub's website before. However, when you are locally executing this code, you may not want to put your access token directly into the source code, so you can create a `.env` file which holds all of your environment variables in the project folder.

In any Node.js application, we use the key as an environment variable in our source code with the following package: dotenv. Afterward, you can use the personal access token from the .env file in your `index.js` file. For now, you have already stored your personal access token as environment variables on our website.

Environment Variables

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

```
import ApolloClient from 'apollo-boost';
import 'dotenv/config';
const client = new ApolloClient({
  uri: 'https://api.github.com/graphql',
  request: operation => {
    operation.setContext({
      headers: {
        authorization: `Bearer ${GITHUB_PERSONAL_ACCESS_TOKEN}`,
      },
    });
  },
});
```

index.js

Cross-Fetch Package

If you start your application locally with `npm start` without query or mutation and just Apollo Client, you might see the following error:

`Error: fetch is not found globally and no fetcher passed, to fix pass a fetch for your environment …`.

The error occurs because the native fetch API, which is used to make requests to remote APIs on a promise basis, is only available in the browser. You can't access it in a Node.js application that runs only in the command line.

Environment Variables

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

index.js

.ed_required_keys

```
import 'dotenv/config';
import ApolloClient from 'apollo-boost';

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

However, the Apollo Client uses the fetch API to perform queries and mutations, usually from a browser environment and not Node.js environment. As you may remember, a query or mutation can be performed with a simple HTTP request, so the Apollo Client uses the native fetch API from a browser to perform these requests. The solution is to use a node package which makes fetch available in a Node.js environment.

Fortunately, there are packages to address this issue: `cross-fetch`. We will import it anonymously in our code:

Environment Variables

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

```
import 'cross-fetch/polyfill';
import ApolloClient from 'apollo-boost';
```

index.js

The error should disappear when you run the application below, but nothing happens just yet. An instance of the Apollo Client is created with a configuration. In the following lesson, we will perform our first query with Apollo Client.

Environment Variables

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

```
import 'dotenv/config';
import 'cross-fetch/polyfill';
import ApolloClient from 'apollo-boost';

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

index.js

Exercise

1. Confirm your source code for the last section

Reading Task

1. Read more about other view integrations such as Angular and Vue.