Getting Started with GitHub's GraphQL API

This lesson will introduce you to GitHub's GraphQL API and how will we go about utilizing it to learn GraphQL!

WE'LL COVER THE FOLLOWING

In this lesson, we'll cover everything you need to get started with **GitHub's GraphQL API**, and to learn to use GraphQL in JavaScript from a client-side perspective by consuming their API.

GitHub is one of the first major tech brands to adopt GraphQL. The released public GraphQL API (official documentation), is quite popular among developers who are already familiar with GitHub.

You must be familiar with the common GitHub terminologies and how to consume GitHub account data using its GraphQL API. We will build multiple applications throughout this course using this API so make sure to give this lesson a thorough read to avoid any confusions.

Feeding the API with Data on GitHub

If you don't have an account on GitHub yet or aren't familiar with it, follow this official GitHub Learning Lab.

For our interactions with GitHub's GraphQL API, we will use our own account with information to read/write from/to data. Before that, you need to complete your GitHub profile by providing additional information so you can recognize it later when it is read by the API.

Paginated Data

GitHub's GraphQL API allows you to request multiple repositories at once, which is useful for pagination.

> *Pagination* is a programming mechanic invented to work with large lists of items. For example, imagine you have more than a hundred repositories in your GitHub account, but your UI only shows ten of them. Transferring the whole list across the wire for each request is impractical and inefficient because only a subset is needed at a time, which pagination allows.

Using pagination with GitHub's GraphQL API lets you adjust the numbers to your own needs, so make sure to adjust the numbers (e.g. limit, offset) to your personal requirements (e.g. available repositories of your GitHub account or available repositories of a GitHub organization). You want to have at least enough repositories in your collection to see the pagination feature in action, so I recommend more than **twenty (20)**, assuming each page will display **ten (10)**, or use **five (5)** repositories when displaying **two (2)**.

Issues and Pull Requests

Once you dive deeper into GitHub's GraphQL API and start requesting nested relationships, for example, issues of repositories, or pull requests of repositories, make sure that the repositories have a few issues or pull requests. So, you'll see something when we implement the feature to show all the issues in a repository. It might be better to request repositories from a GitHub organization where there will be plenty of issues and pull requests.

Until now you have been given a brief overview of the GitHub's GraphQL API, next we'll further explore and learn to set up for it in the next lesson.