React GraphQL Query

Learn to implement GraphQL Query in React and fetch issues from an organization's repository.

WE'LL COVER THE FOLLOWING

- Fetching an ‘Organization’
- Response of the GraphQL Query
- `Organization` Component to display result
- Error Handling

In this lesson, we are going to implement our first GraphQL query in React and get issues as a result of the query from an organization’s repository for our **Issue Tracker** application.

Fetching an ‘Organization’

Let’s start by fetching only an organization. Here goes the query that is to be defined as a variable above the `App` component:

Environment Variables

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

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

We have used template literals in JavaScript to define the query as a string with multiple lines. It should be identical to the query used before in

GraphiQL or GitHub Explorer. Now, we will use `axios` to make a `POST` request to GitHub's GraphQL API.

The configuration for `axios` already points to the correct API endpoint and uses personal access token. The only thing left is passing the query to it as payload during a POST request. The argument for the endpoint can be an empty string because we defined the endpoint in the configuration. It will execute the request when the App component mounts in `componentDidMount()`. After the promise from `axios` has been resolved, only a console log of the result remains.

Environment Variables

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

```
...
const axiosGitHubGraphQL = axios.create({
  baseURL: 'https://api.github.com/graphql',
  headers: {
    Authorization: `bearer ${
      process.env.REACT_APP_GITHUB_PERSONAL_ACCESS_TOKEN
    }`,
  },
});

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

class App extends Component {
  ...
  componentDidMount() {
    this.onFetchFromGitHub();
  }

  onSubmit = event => {
    // fetch data

    event.preventDefault();
  };

  onFetchFromGitHub = () => {
    axiosGitHubGraphQL
      .post('', { query: GET_ORGANIZATION })
      .then(result => console.log(result));

  };
  ...
}
```

We use `axios` to perform a HTTP POST request with a GraphQL query as payload. Since `axios` uses promises, the promise resolves eventually and gives the result from the GraphQL API. We then display the result on the console. The implementation is in plain JavaScript using `axios` as HTTP client to perform the GraphQL request with plain HTTP.

Run the code below and you will get the result on the console.

> *If you get a 401 HTTP status code, you might not have set up your personal access token properly. Otherwise, if everything went fine, you should see the desired result on your console.*

Environment Variables

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

```
import React, { Component } from 'react';
import axios from 'axios';

const axiosGitHubGraphQL = axios.create({
  baseURL: 'https://api.github.com/graphql',
  headers: {
    Authorization: `bearer ${
      process.env.REACT_APP_GITHUB_PERSONAL_ACCESS_TOKEN
    }`,
  },
});

const TITLE = 'React GraphQL GitHub Client';
const GET_ORGANIZATION = `
  {
    organization(login: "the-road-to-learn-react") {
      name
      url
    }
  }
`;
class App extends Component {

  state = {
    path: 'the-road-to-learn-react/the-road-to-learn-react',
  };


   componentDidMount() {
    // fetch data
      this.onFetchFromGitHub();
  }

  onChange = event => {
    this.setState({ path: event.target.value });
  };

  onSubmit = event => {
    // fetch data

    event.preventDefault();
  };

        onFetchFromGitHub = () => {
    axiosGitHubGraphQL
      .post('', { query: GET_ORGANIZATION })
      .then(result => console.log(result));
  };
  render() {
    const { path } = this.state;

    return (
      <div>
        <h1>{TITLE}</h1>
        <form onSubmit={this.onSubmit}>
          <label htmlFor="url">
            Show open issues for https://github.com/
          </label>
          <input
            id="url"
            type="text"
            value={path}
            onChange={this.onChange}
            style={{ width: '300px' }}
          />
          <button type="submit">Search</button>
        </form>

        <hr />

        {/* Here comes the result! */}
      </div>
    );
  }
}

export default App;
```

If you open the output in a new tab or access the URL of your app, then you will see something similar to below in your browser console log:

Environment Variables

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

```
{
  "data":{
    "data":{
      "organization":{
        "name":"The Road to learn React",
        "url":"https://github.com/the-road-to-learn-react"
      }
    }
  },
  "headers": ...,
  "request": ...,
  "status": ...,
  "config": ...,
  "statusText": ...
}
```

Response of the GraphQL Query

The top-level information is everything which axios returns as meta information for the request. Till now, everything we have received is from axios and not GraphQL, which is why most of it is substituted with a placeholder. Axios has a data property that shows the result of your axios request.

On further examination, we can find that there are two `data` properties: one from axios, while the other comes from the GraphQL data structure.

The result of the GraphQL query is in the second `data` property. Here you will find the organization with its resolved name and URL fields as string properties.

In the next step, we will store the result holding the information about the organization in our React's local state. We will also store potential errors in the state as well.

Environment Variables

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

```
class App extends Component {
  state = {
    path: 'the-road-to-learn-react/the-road-to-learn-react',
    organization: null,

    errors: null,
  };
  ...
  onFetchFromGitHub = () => {
    axiosGitHubGraphQL
      .post('', { query: GET_ORGANIZATION })
      .then(result =>
        this.setState(() => ({
          organization: result.data.data.organization,
          errors: result.data.errors,
        })),
      );
  }
  ...
}
```

`Organization` Component to display result

In the second step, you can display the information about the organization in your `App` component's `render()` method:

Environment Variables

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

```
class App extends Component {
  ...
  render() {
    const { path, organization } = this.state;
    return (
      <div>
        <h1>{TITLE}</h1>
        <form onSubmit={this.onSubmit}>
          ...
        </form>
        <hr />
        <Organization organization={organization} />
      </div>
    );
  }
}
```

We will introduce the Organization component as a new **functional stateless component** to keep the render method of the `App` component concise.

Environment Variables

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

```
class App extends Component {
  ...
}

const Organization = ({ organization }) => (
  <div>
    <p>
      <strong>Issues from Organization:</strong>
      <a href={organization.url}>{organization.name}</a>
    </p>
  </div>
);
```

Error Handling

In the final step, you have to decide what should be rendered when nothing is fetched yet, and what should be rendered when errors occur.

To solve these edge cases, you can use conditional rendering in React. For the first edge case, simply check whether an `organization` is present or not.

Environment Variables

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

```
class App extends Component {
  ...
  render() {
    const { path, organization, errors } = this.state;

    return (
      <div>
        ...
        <hr />
        {organization ? (
          <Organization organization={organization} errors={errors} />
        ) : (
          <p>No information yet ...</p>
        )}
      </div>
    );
  }
}
```

For the second edge case, you have passed the errors to the Organization component. In case there are errors, it should simply render the error message of each error. Otherwise, it should render the organization. There can be multiple errors regarding different fields and circumstances in GraphQL.

Environment Variables

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

```
const Organization = ({ organization, errors }) => {
  if (errors) {
    return (
      <p>
        <strong>Something went wrong:</strong>
        {errors.map(error => error.message).join(' ')}
      </p>
    );
  }

  return (
    <div>
      <p>
        <strong>Issues from Organization:</strong>
        <a href={organization.url}>{organization.name}</a>
      </p>
    </div>
  );
};
```

Congratulations! You performed your first GraphQL query in a React application, a plain HTTP POST request with a query as payload. You used a configured axios client instance for it. Afterward, you were able to store the result in React's local state to display it later.

Environment Variables

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

```
import React, { Component } from 'react';
import axios from 'axios';

const axiosGitHubGraphQL = axios.create({
  baseURL: 'https://api.github.com/graphql',
  headers: {
    Authorization: `bearer ${
      process.env.REACT_APP_GITHUB_PERSONAL_ACCESS_TOKEN

    }`,
  },
});

const TITLE = 'React GraphQL GitHub Client';
const GET_ORGANIZATION = `
  {
    organization(login: "the-road-to-learn-react") {
      name
      url
    }
  }
`;
class App extends Component {

   state = {
    path: 'the-road-to-learn-react/the-road-to-learn-react',
  };

   componentDidMount() {
    // fetch data
      this.onFetchFromGitHub();
  }

  onChange = event => {
    this.setState({ path: event.target.value });
  };

  onSubmit = event => {
    // fetch data

    event.preventDefault();
  };

        onFetchFromGitHub = () => {
    axiosGitHubGraphQL
      .post('', { query: GET_ORGANIZATION })
      .then(result =>
            this.setState(() => ({
          organization: result.data.data.organization,
          errors: result.data.errors,
        })),
        );
  };
  render() {
      const { path, organization, errors } = this.state;

    return (
      <div>
        <h1>{TITLE}</h1>
        <form onSubmit={this.onSubmit}>
          <label htmlFor="url">
            Show open issues for https://github.com/
          </label>
          <input
            id="url"
            type="text"
            value={path}
            onChange={this.onChange}
            style={{ width: '300px' }}
          />
          <button type="submit">Search</button>
        </form>

        <hr />

       {organization ? (
          <Organization organization={organization} errors={errors} />
        ) : (
          <p>No information yet ...</p>
        )}
      </div>
    );
  }
}

const Organization = ({ organization, errors }) => {
  if (errors) {
    return (
      <p>
        <strong>Something went wrong:</strong>
        {errors.map(error => error.message).join(' ')}
      </p>
    );
  }

  return (
  <div>
    <p>
      <strong>Issues from Organization:</strong>
      <a href={organization.url}>{organization.name}</a>
    </p>
  </div>
);
};

export default App;
```