GraphQL Query/Mutation with Higher-Order Components in React

Let's learn GraphQL mutations using Higher-Order Components with Apollo Client in React!

WE'LL COVER THE FOLLOWING

We've done `Query` and `Mutation` components from React Apollo to connect a data-layer (Apollo Client) with a view-layer (React).

The `Query` component executes the query when it is rendered, whereas the `Mutation` component gives access to a function that triggers the mutation. Both components use the *render props pattern* to make the results accessible in their child functions.

Enter the Higher-Order Components

Higher-Order Components (HOC) is a widely accepted alternative to React's render prop pattern. The **React Apollo** package implements a *Higher-Order Component* for queries and mutations as well, although the team behind Apollo does not advertise it, and even spoke in favor of render props as their first choice. Nonetheless, this lesson shows you the alternative, using a *Higher-Order Component* instead of a *Render Prop*, though our application will continue to use the render prop pattern afterward.

If you already have access to the query result in the `Profile` component's arguments, there is no `Query` component needed in the component itself:

Environment Variables

Key: Value:

REACT_APP_GITHUB... Not Specified...

GITHUB_PERSONAL... Not Specified...

```
const Profile = ({ data, loading, error }) => {
  if (error) {
    return <ErrorMessage error={error} />;
  }

  const { viewer } = data;

  if (loading || !viewer) {
    return <Loading />;
  }

  return <RepositoryList repositories={viewer.repositories} />;
};
```

src/Profile/index.js

There is no GraphQL involved here because all you see is the pure view-layer. Instead, the data-layer logic is extracted into a Higher-Order Component. We import the `graphql` HOC from the React Apollo package in order to apply it on the `Profile` component, which takes the query definition as an argument.

Environment Variables

Key: Value:

REACT_APP_GITHUB... Not Specified...

GITHUB_PERSONAL... Not Specified...

```
import React from 'react';
import gql from 'graphql-tag';
import { graphql } from 'react-apollo';

...

const GET_REPOSITORIES_OF_CURRENT_USER = gql`
  {
    viewer {
      ...
    }
  }
`;

const Profile = ({ data, loading, error }) => {
  ...
};

export default graphql(GET_REPOSITORIES_OF_CURRENT_USER)(Profile);
```

src/Profile/index.js

Conclusion

I find the ***HOC*** approach cleaner than the ***render props*** because it co-locates both the *data-layer* and the *view-layer* instead of inserting one into the other.

However, the team behind Apollo made the decision to favor render props instead. While I find the HOC approach more concise, the render prop pattern comes with its own advantages for mutating and querying data. For instance, imagine a query depends on a prop used as a variable. It would be cumbersome to access the incoming prop in a statically-defined Higher-Order Component, but it can be dynamically used in a render prop because it is used within the `Profile` component where the props are naturally accessible.

Another advantage is the power of composition for render props, which is useful when one query depends on the result of another. It can be achieved with HOCs as well, but again, it is more cumbersome. It boils down to seemingly never-ending “Higher-Order Components vs Render Props” discussions.


```
import React from 'react';

import Link from '../../Link';

import './style.css';

const Footer = () => (
  <div className="Footer">
    <div>
      <small>
        <span className="Footer-text">Built by</span>{' '}
        <Link
          className="Footer-link"
          href="https://www.robinwieruch.de"
        >
          Robin Wieruch
        </Link>{' '}
        <span className="Footer-text">with &hearts;</span>
      </small>
    </div>
```

```
    <div>
      <small>
        <span className="Footer-text">

          Interested in GraphQL, Apollo and React?
        </span>{' '}
        <Link
          className="Footer-link"
          href="https://www.getrevue.co/profile/rwieruch"
        >
          Get updates
        </Link>{' '}
        <span className="Footer-text">
          about upcoming articles, books &
        </span>{' '}
        <Link className="Footer-link" href="https://roadtoreact.com">
          courses
        </Link>
        <span className="Footer-text">.</span>
      </small>
    </div>
  </div>
);

export default Footer;
```

Exercises

1. Confirm your source code for the last section.
2. Come up with your own opinion about the advantages and disadvantages of using a Higher-Order Component or Render Prop.
3. Try to implement one of your mutations with a Higher-Order Component.