

# Apollo Client Error Handling in React

In this lesson, you will learn how to handle GraphQL errors and network errors when using Apollo Client in React.

## WE'LL COVER THE FOLLOWING ^

- `apollo-link-error`
- `apollo-link`
- Exercises
- Reading Tasks

Before diving into GraphQL mutations in React with Apollo Client, this lesson should clarify error handling with Apollo in React.

The error handling happens on two levels:

- the application level
- query/mutation level

Both can be implemented with the two cases that follow. On a query level, in your `Profile` component, you have access to the query `data` and `loading` properties. Apart from these, you can also access the `error` object, which can be used to show a conditional error message.

## Environment Variables ^

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

...

```
import RepositoryList from '../Repository';
import Loading from '../Loading';
import ErrorMessage from '../Error';
```



```

...
const Profile = () => (
  <Query query={GET_REPOSITORIES_OF_CURRENT_USER}>
    {{ { data, loading, error } }} => {
      if (error) {
        return <ErrorMessage error={error} />;
      }

      const { viewer } = data;

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

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

export default Profile;

```

src/Profile/index.js

Whereas the **ErrorMessage** component from the *src/Error/index.js* could look like the following:

#### Environment Variables



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

```

import React from 'react';

import './style.css';

const ErrorMessage = ({ error }) => (
  <div className="ErrorMessage">
    <small>{error.toString()}</small>
  </div>
);

export default ErrorMessage;

```



src/Error/index.js

If we now try to change the name of a field in our query to something not offered by GitHub's GraphQL API, and observe what's rendered in the browser. We should see something like this:

**Error: GraphQL error: Field 'viewers' doesn't exist on type 'Query'.**


Or, if you simulate offline functionality, you'll see: `Error: Network error: Failed to fetch`.

That's how errors can be separated into GraphQL errors and network errors. You can handle errors on a component or query level, but it will also help with mutations later.

## apollo-link-error #

To implement error handling on an application level, we will add another Apollo package: `apollo-link-error`.

You can import it in your `src/index.js` file and create such an error link:

Environment Variables 


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

```
import React from 'react';
import ReactDOM from 'react-dom';
import { ApolloProvider } from 'react-apollo';
import { ApolloClient } from 'apollo-client';
import { HttpLink } from 'apollo-link-http';
import { onError } from 'apollo-link-error';
import { InMemoryCache } from 'apollo-cache-inmemory';

...

const errorLink = onError(({ graphQLErrors, networkError }) => {
  if (graphQLErrors) {
    // do something with graphql error
  }

  if (networkError) {
    // do something with network error
  }
});
```



src/index.js

You could differentiate the error handling at the application level into *development* and *production* mode. During development, it might be sufficient to console log the errors to a developer console in your browser while in production mode, you can set up an error tracking service like [Sentry](#). It will

production mode, you can set up an error tracking service like [Sentry](#). It will teach you to identify bugs in a web dashboard more efficiently.

## apollo-link #

Now you have two links in your application: `httpLink` and `errorLink`. To combine them for use with the Apollo Client instance, we'll add yet another useful package in the Apollo ecosystem that makes link compositions possible in the command line: `apollo-link`.

Secondly, we use it to combine our two links in the `src/index.js` file:

Environment Variables 

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

```
...
import { ApolloClient } from 'apollo-client';
import { ApolloLink } from 'apollo-link';
import { HttpLink } from 'apollo-link-http';
import { onError } from 'apollo-link-error';
import { InMemoryCache } from 'apollo-cache-inmemory';

...

const httpLink = ...

const errorLink = ...

const link = ApolloLink.from([errorLink, httpLink]);

const cache = new InMemoryCache();

const client = new ApolloClient({
  link,
  cache,
});
```



src/index.js

That's how two or multiple links can be composed for creating an Apollo Client instance. There are several links developed by the community and Apollo maintainers who extend the Apollo Client with advanced functionality. Remember, it's important to understand that links can be used to access and modify the GraphQL control flow. When doing so, be careful to chain the control flow in the correct order.

The `apollo-link-http` is called a **terminating link** because it turns an operation into a result that usually occurs from a network request. On the other side, the `apollo-link-error` is a **non-terminating link**. It only enhances your terminating link with features, since a terminating link has to be the last entity in the control flow chain.

You can mess up with your query and see whether the error handling works or not:

For example, change the argument `direction: DESC` to `direction: DC` in

```
GET_REPOSITORIES_OF_CURRENT_USER = gql{
  ...
  orderBy: { direction: DESC, field: STARGAZERS }
  ...
};
```

in `src/Profile/index.js`.

Environment Variables



Key:

Value:

REACT\_APP\_GITHUB... Not Specified...

GITHUB\_PERSONAL... Not Specified...

```
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. Implement the [apollo-link-retry](#) in case a network request fails

## Reading Tasks #

1. Read more about [different Apollo Error types and error policies](#)
2. Read more about [Apollo Links](#)
3. Read more about [composable Apollo Links](#)