Introducción a Apollo

ABRIL 2019

AGENDA

- GraphQL basics
- HoC vs Function as child
- Query components pattern
- Apollo cache (single source of truth)
- Optimistic updates, refetch and pagination
- Advanced features (batchhttp, @client, subscriptions)
- Hooks

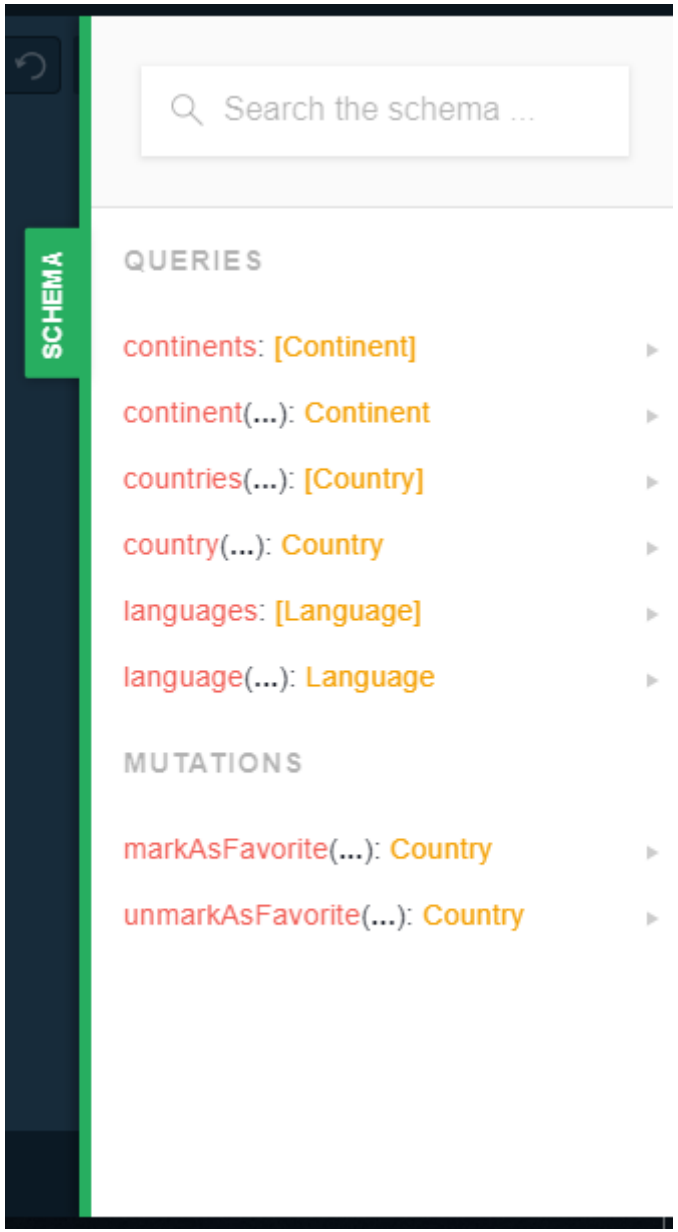
GRAPHQL BASICS

- Query language
- Ability to select return fields
- Single endpoint over HTTP
- Queries and mutations

https://github.com/APIs-guru/graphql-apis

https://github.com/ChristianRuiz/countries

https://github.com/ChristianRuiz/ApolloIntro



BACK TO 2017: HOC

- Added props
  - data
    - loading
    - error

Pros:

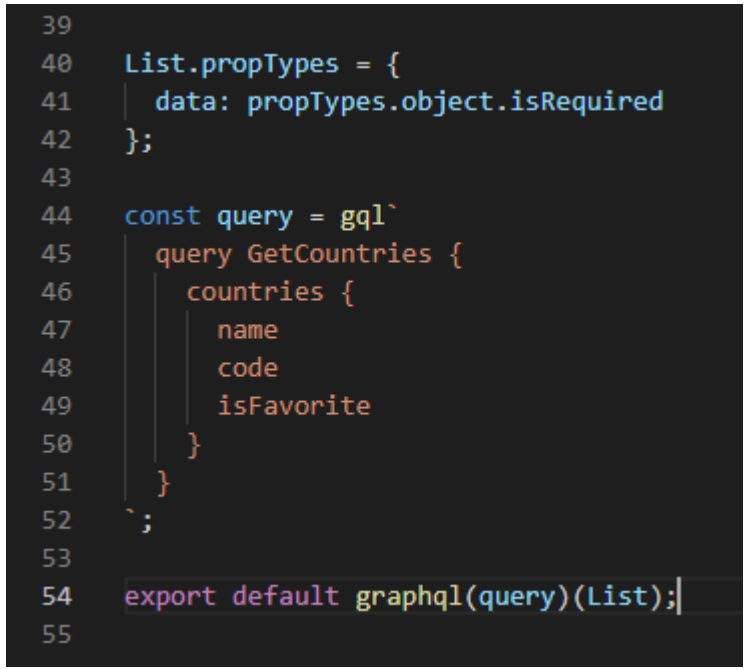
Isolated components

Conts:

Overriding props

One to one design

Tightly-coupled



```
39
40  List.propTypes = {
41    data: propTypes.object.isRequired
42  };
43
44  const query = gql`
45    query GetCountries {
46      countries {
47        name
48        code
49        isFavorite
50      }
51    }
52  `;
53
54  export default graphql(query)(List);
55
```

INTO 2018: FUNCTION AS A CHILD

- Render prop pattern
- Used in React Context API
- Declarative components and variables

Pros:

No extra props

More visual in a render

Conts:

Adds noise to the React tree

Tightly-coupled

```
10  export const QUERY = gql`
11    query GetCountries {
12      countries {
13        name
14        code
15        isFavorite
16      }
17    }
18  `;
19
20  const List = () => (
21    <div>
22      <Query query={QUERY}>
23        {
24          ({ loading, error, data }) => {
25            // TODO: handle loading and error
26            const { countries } = data;
27
28            return (<ul>
29              {countries.map(country => (<li>{country.name}</li>))}
30            </ul>);
31          }
32        }
33      </Query>
34    </div>
35  );
36
37  export default List;
38
```

QUERY COMPONENTS PATTERN

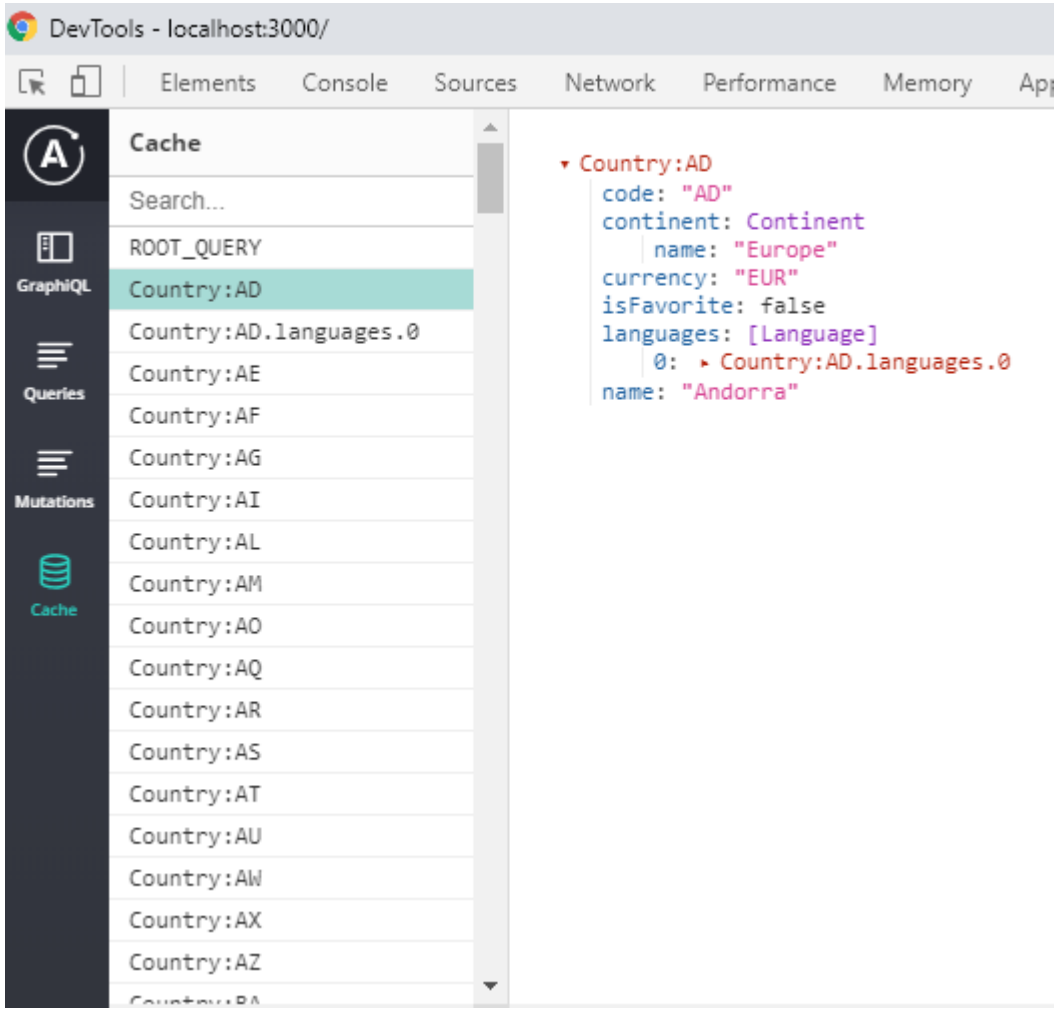
- Function as a child
- Decouples Apollo
- Easy to track source of data

```
8   const MapPicker = () => (
9     // ...
10    <CitiesQuery country="Netherlands">
11      {({ allCities }) =>
12        <CitiesList cities={allCities} />
13      }
14    </CitiesQuery>
15    // ...
16  );
17
```

https://blog.apollographql.com/query-components-with-apollo-ec603188c157

APOLLO ~~STORE~~ CACHE

- Single source of truth
- Don’t stop the data flow
- Uses [Type]:[id] as key
- [Apollo dev tools]



APOLLO MAGIC

- Refetch

https://www.apollographql.com/docs/react/essentials/queries.html#refetching

- Optimistic UI

https://www.apollographql.com/docs/react/features/optimistic-ui.html

- Pagination

https://www.apollographql.com/docs/react/features/pagination.html

ADVANCED FEATURES

- Batching (needs ApolloBoost migration)

https://www.apollographql.com/docs/link/links/batch-http

- Local state queries and mutations (@client)

https://www.apollographql.com/docs/react/essentials/local-state#local-resolvers

- Testing (MockedProvider)

https://www.apollographql.com/docs/react/recipes/testing#mockedprovider

- Subscriptions

https://www.apollographql.com/docs/react/advanced/subscriptions

INTO THE FUTURE: HOOKS

- Waiting for suspense

  Github issue
- 3rd party package

  react-apollo-hooks

```
14  const Detail = ({ code }) => {
15    const { loading, data } = useGetCountryByCodeQuery(code);
16    const { country } = data;
17
18    return (
19      <div className="country-detail">
20        <Paper elevation={1} key={code} className="country-detail-card">
21          {loading ? (
22            <Loading />
23          ) : (
24            <Fragment>
25              <div className="country-detail-card-content">
26                <img src={`https://www.countryflags.io/${code}/flat/64.png`} alt={code} />
27                <Typography variant="h5" component="h3">
28                  {country.name}
29                </Typography>
30                <Typography component="p">
31                  <b>Continent: </b>
32                  {country.continent.name}
33                </Typography>
```

CHRISTIAN RUIZ
christian.ruiz@beezy.net