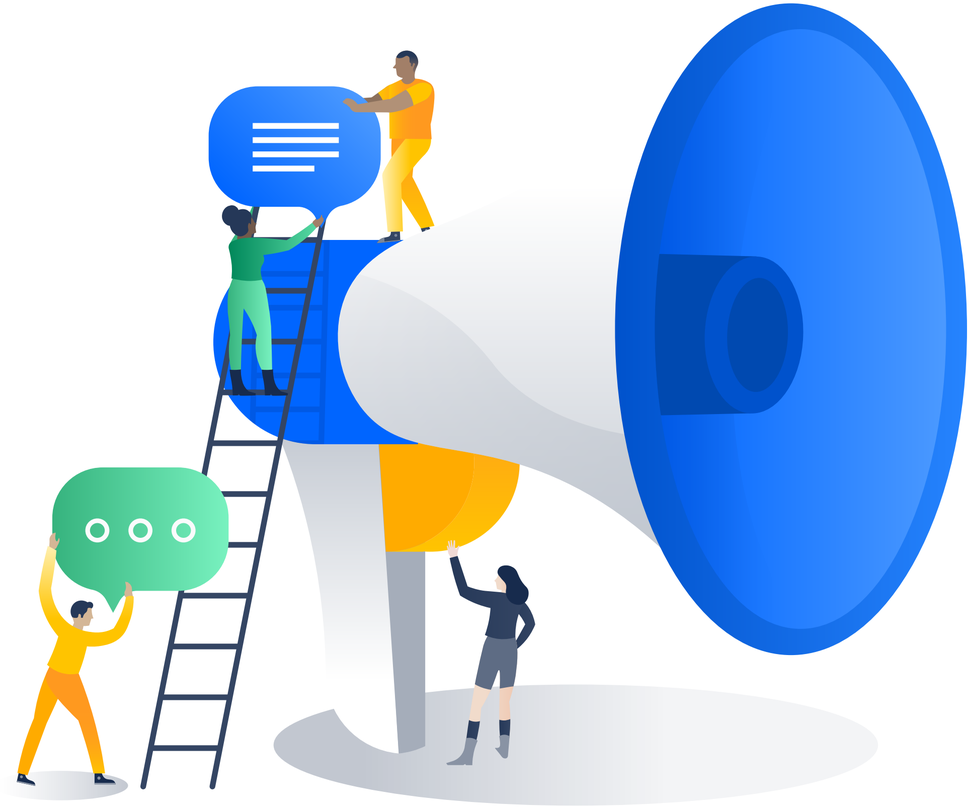
GraphQL Queries in Apollo

Fetch data with Query components and build a Subscriptions server

SAIDUR RAHMAN | SOFTWARE ENGINEER | @SAIDUR2



Hi, I’m Saidur. 👋

Agenda

Queries in React Apollo

Subscriptions in React Apollo

Building Subscriptions Server

Demo

```
import gql from "graphql-tag";
import { Query } from "react-apollo";

const GET_DOGS = gql`
  {
    dogs {
      id
      breed
    }
  }
`;

const Dogs = ({ onDogSelected }) => (
  <Query query={GET_DOGS}>
    {({ loading, error, data }) => {
      if (loading) return "Loading...";
      if (error) return `Error! ${error.message}`;

      return (
        <select name="dog" onChange={onDogSelected}>
          {data.dogs.map(dog => (
            <option key={dog.id} value={dog.breed}>
              {dog.breed}
            </option>
          ))}
        </select>
      );
    }}
  </Query>
);
```

The Query component

One of the most important building blocks of your Apollo application.

Creating a GraphQL query

Pass a GraphQL query string and provide a function that tells React what to render

Receiving Data

Apollo Client returns object containing loading, error, and data properties that you can use to render your UI.

```
import gql from "graphql-tag";
import { Query } from "react-apollo";

const GET_DOGS = gql`
  {
    dogs {
      id
      breed
    }
  }
`;

const Dogs = ({ onDogSelected }) => (
  <Query query={GET_DOGS}>
    {({ loading, error, data }) => {
      if (loading) return "Loading...";
      if (error) return `Error! ${error.message}`;

      return (
        <select name="dog" onChange={onDogSelected}>
          {data.dogs.map(dog => (
            <option key={dog.id} value={dog.breed}>
              {dog.breed}
            </option>
          ))}
        </select>
      );
    }}
  </Query>
);
```

The Query component

One of the most important building blocks of your Apollo application.

Creating a GraphQL query

Pass a GraphQL query string and provide a function that tells React what to render

Receiving Data

Apollo Client returns object containing loading, error, and data properties that you can use to render your UI.

```
import gql from "graphql-tag";
import { Query } from "react-apollo";

const GET_DOGS = gql`
  {
    dogs {
      id
      breed
    }
  }
`;

const Dogs = ({ onDogSelected }) => (
  <Query query={GET_DOGS}>
    {({ loading, error, data }) => {
      if (loading) return "Loading...";
      if (error) return `Error! ${error.message}`;

      return (
        <select name="dog" onChange={onDogSelected}>
          {data.dogs.map(dog => (
            <option key={dog.id} value={dog.breed}>
              {dog.breed}
            </option>
          ))}
        </select>
      );
    }}
  </Query>
);
```

The Query component

One of the most important building blocks of your Apollo application.

Creating a GraphQL query

Pass a GraphQL query string and provide a function that tells React what to render

Receiving Data

Apollo Client returns object containing loading, error, and data properties that you can use to render your UI.

Receiving Data

Polling
Achieve near real-time data by causing the query to refetch on a specified interval or dynamic polling

Refetching
If you want to reload the query in response to a user action instead of an interval

Manually firing a query
Delay firing your query until the user performs an action? Directly call client.query() instead

Subscriptions
Push data from the server to the clients that choose to listen to real time messages from the server

Agenda

Queries in React Apollo

Subscriptions in React Apollo

Building Subscriptions Server

Demo

Apollo Client Subscriptions

Usage

Specify a set of fields to be delivered to the client and result is sent every time a particular event happens on the server (similar to query)

Implementation

The server is written to send a new result to the client every time an event occurs

When to use

The initial state is large or you care about low-latency updates in the case of specific events. Most cases polling and fetch will do the job.

CLIENT TRANSPORT

```
import { split } from 'apollo-link';
import { HttpLink } from 'apollo-link-http';
import { WebSocketLink } from 'apollo-link-ws';
import { getMainDefinition } from 'apollo-utilities';

// Create an http link:
const httpLink = new HttpLink({
  uri: 'http://localhost:3000/graphql'
});

// Create a WebSocket link:
const wsLink = new WebSocketLink({
  uri: `ws://localhost:5000/`,
  options: {
    reconnect: true
  }
});

// using the ability to split links, you can send data to each link
// depending on what kind of operation is being sent
const link = split(
  // split based on operation type
  ({ query }) => {
    const { kind, operation } = getMainDefinition(query);
    return kind === 'OperationDefinition' && operation === 'subscription';
  },
  wsLink,
  httpLink,
);
```

Configure Apollo to send queries and mutations over HTTP, but subscriptions over websocket

SUBSCRIPTION COMPONENT

```
const COMMENTS_SUBSCRIPTION = gql`
  subscription onCommentAdded($repoFullName: String!) {
    commentAdded(repoFullName: $repoFullName) {
      id
      content
    }
  }
`;

const DontReadTheComments = ({ repoFullName }) => (
  <Subscription
    subscription={COMMENTS_SUBSCRIPTION}
    variables={{ repoFullName }}
  >
    {({ data: { commentAdded }, loading }) => (
      <h4>New comment: {!loading && commentAdded.content}</h4>
    )}
  </Subscription>
);
```

Agenda

Queries in React Apollo

Subscriptions in React Apollo

Building Subscriptions Server

Demo

Setting up a Subscriptions Server

Step 1

Declare subscriptions in the GraphQL schema

Step 2

Setup a PubSub instance that our server will publish new events to

Step 3

Hook together PubSub event and GraphQL subscription.

Step 4

Setting up Subscriptions Server, a transport between the server and the clients

Agenda

Queries in React Apollo

Subscriptions in React Apollo

Building Subscriptions Server

Demo

Q&A


ATLASSIAN