

# GQL ALL THE THINGS

# MANAGING LOCAL STATE WITH APOLLO







# I HAVE CSS-IN-GQL STICKERS

<https://github.com/braposo/graphql-css>







**MY NAME IS SARA**



Front End developer at YLD

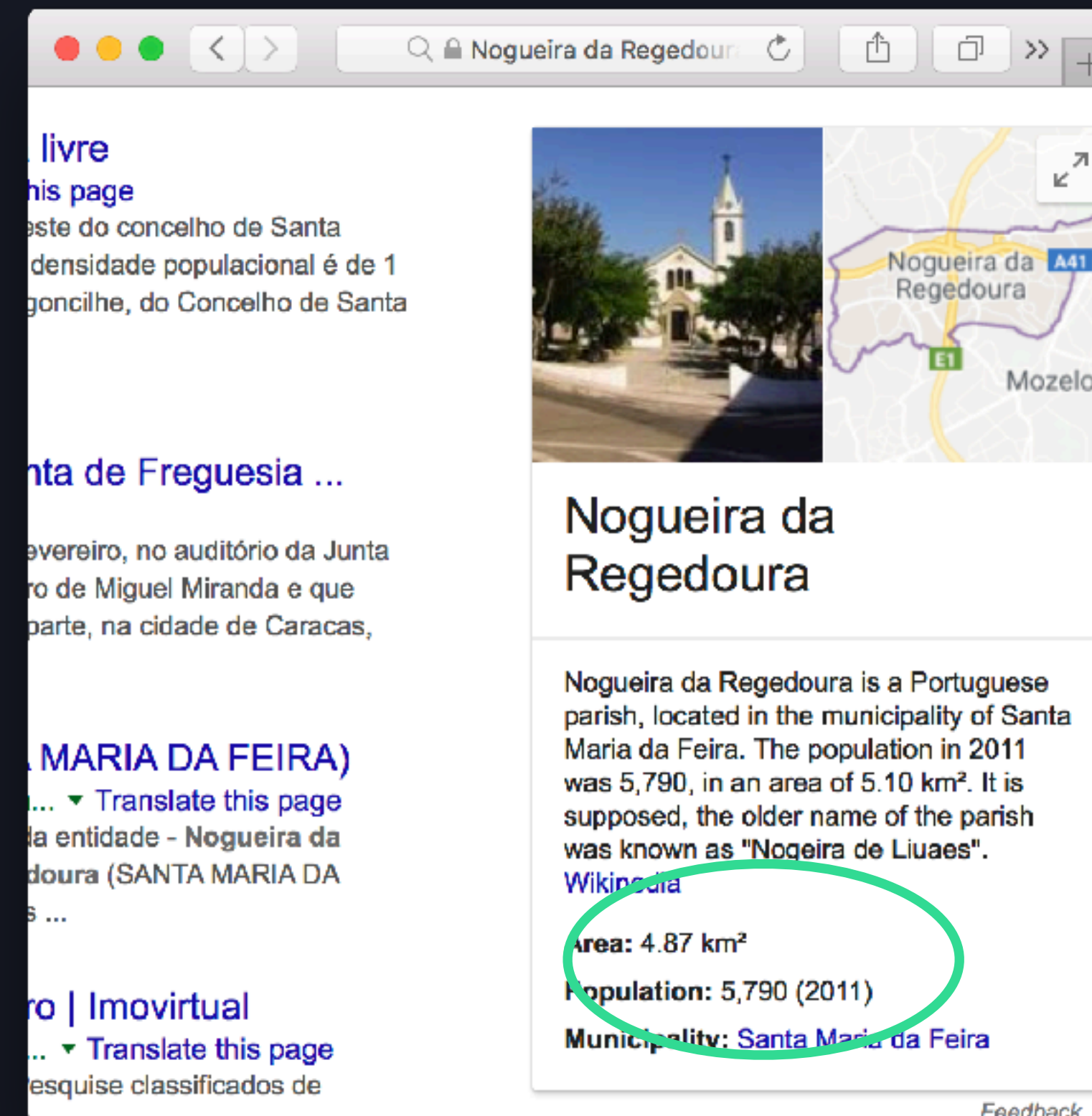
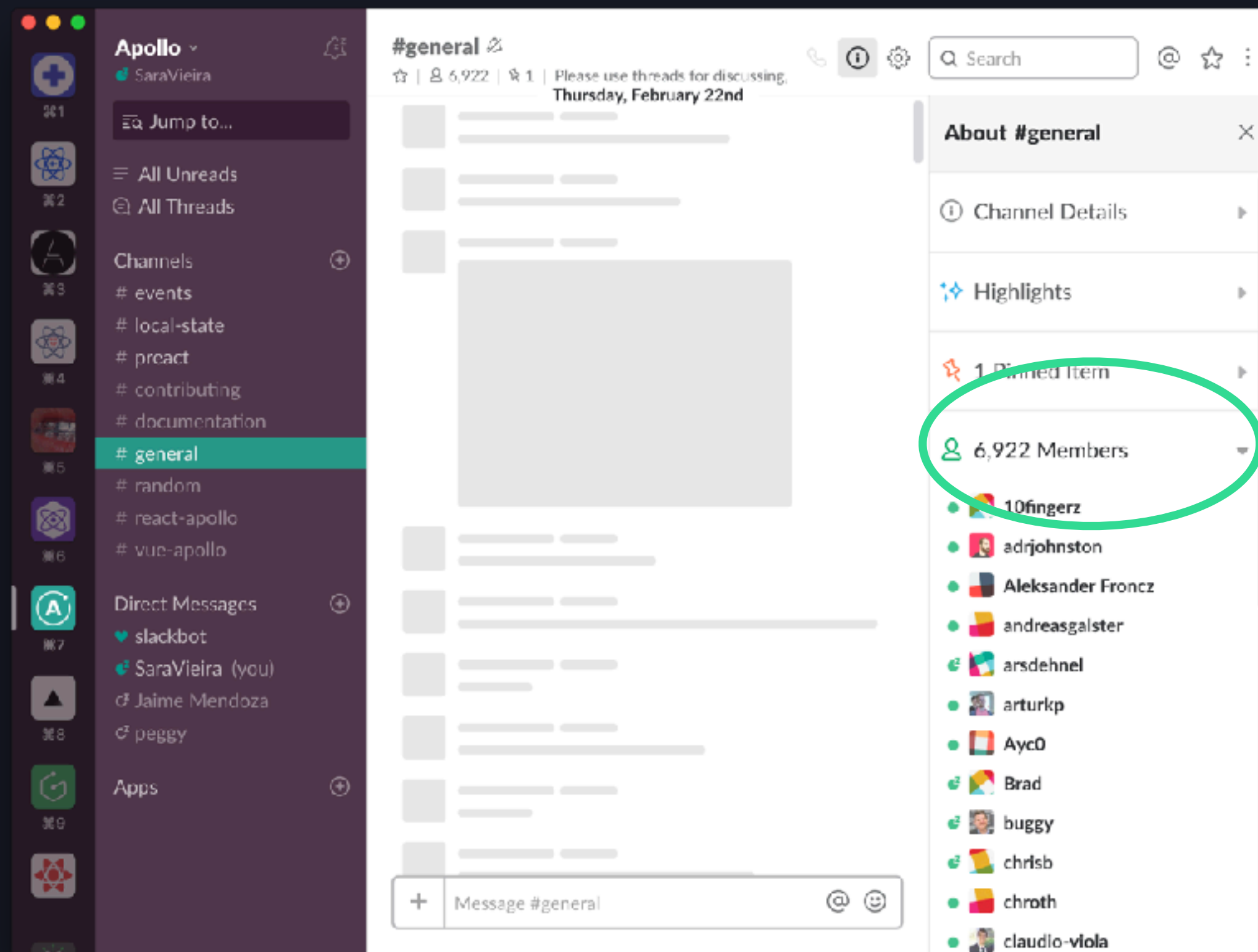
Really into shitty movies

\*please y'all need to watch troll 2\*

Really into football

Lives out of airbnb's

# LIVES IN THE MIDDLE OF NOWHERE





# ALERT

# I DON'T LIKE REDUX

*Dan is great tho ...*



- We got this awesome React project that is all Apollo and GraphQL, want to help?
- That sounds awesome. What do you use for state management?
- Redux.



# WHY?



# GQL ALL THE THINGS

GitHub, Inc.

Sign in or Sign up

apollographql / apollo-link-state

Watch46

Star776

Fork53

Code

Issues16

Pull requests2

Projects0

Insights

Manage your application's state with Apollo!

133 commits

4 branches

2 releases

16 contributors

MIT

Branch: master

New pull request

Find file

Clone or download

chantlong and peggyrayzis apollo link state with apollo boost

Latest commit 404b6a1 16 days ago

.github	[apollo-bot] Update the Templates with docs label	a month ago
examples	Remove visibility filter mutation from schema	22 days ago
packages/apollo-link-state	Update CHANGELOG.md	14 days ago
.gitignore	initial commit	6 months ago
.npmignore	add npmignore	5 months ago
.travis.yml	Temporarily disabling danger until it works again	4 months ago
CONTRIBUTING.md	cleaned up repo	6 months ago
LICENSE	cleaned up repo	6 months ago
README.md	apollo link state with apollo boost	22 hours ago
appveyor.yml	cleaned up repo	6 months ago
codecov.yml	cleaned up repo	6 months ago
dangerfile.ts	cleaned up repo	6 months ago







# HOW?



FIRST... APOLLO BOOST







**REMEMBER THIS ?**





~/Projects

```
> yarn add yarn add apollo-client apollo-cache-  
inmemory apollo-link-http react-apollo graphql-  
tag graphql
```

I CAN'T REMEMBER THIS

NOW:





~/Projects

> yarn add apollo-boost graphql react-apollo|

```
import React from 'react';
import ApolloClient from 'apollo-boost';
import { ApolloProvider } from 'react-apollo';
import Main from './Main'
```

```
const client = new ApolloClient({
  // YOUR STUFF
});
```

```
const App = () => (
  <ApolloProvider client={client}>
    <Main />
  </ApolloProvider>
)
```

# DONE!







# NOW STATE



```
export const client = new ApolloClient({  
  uri: 'https://figma-graphql.now.sh/graphql',  
  clientState: {  
    defaults, ← What ?  
    resolvers ← What ?  
  }  
})
```

```
const App = () => (  
  <ApolloProvider client={client}>  
    <Main />  
  </ApolloProvider>  
)
```

# DEFAULTS

*Your base state. What you start with*

```
export const defaults = {  
  modal: {  
    modalIsOpen: false,  
    __typename: 'modal'  
  }  
}
```



# RESOLVERS

*This is here all the magic happens to retrieve and update your local data in the Apollo cache*

```
export const resolvers = {
  Mutation: {
    openModal: (_, params, { cache }) => {
      const data = {
        modal: {
          modalIsOpen: true,
          __typename: "modal"
        }
      };

      cache.writeData({ data });
      return null;
    }
  }
};
```

# THE MUTATION

*You know ... The GraphQL part of the equation*



```
import { gql } from 'apollo-boost'
```

```
export default gql`  
  mutation openModal {  
    openModal @client  
  }  
`
```

# THE QUERY

*You know ... The GraphQL part of the equation*

```
import { gql } from "apollo-boost";

export default gql`
  {
    modal @client {
      modalIsOpen
    }
  }
`;
```



# WE MADE THE BEHIND THE SCENES

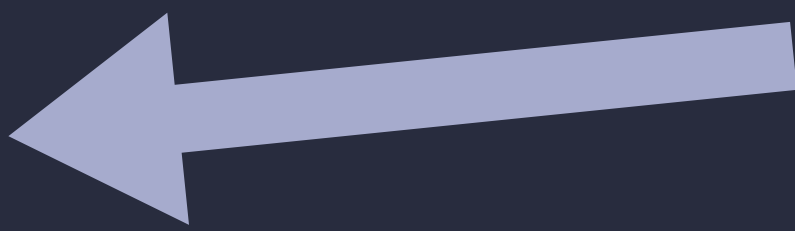
*How do you actually call these things ?*

# THE QUERY COMPONENT

```
import React from "react";
import Modal from "react-modal";
import { Query } from "react-apollo";
```

```
import { GET_MODAL } from "../queries/";
```

```
export default () => (
  <div>
    <Query query={GET_MODAL}>
      ({ { loading, error, modal: { modalIsOpen } }) => {
        if (loading) {
          return "loading man ... Chill ... ";
        }
        if (error) return `Error!: ${error}`;
        return (
          <Modal isOpen={modalIsOpen}>
            <h1>Of course they do ... </h1>
          </Modal>
        );
      }
    </Query>
  </div>
);
```



**Wat?**



**CAN WE MAKE THIS  
SMALLER?**

HELL YEAH  
LET'S CREATE OUR VERY OWN  
QUERY COMPONENT

```
import React from "react";
import { Query } from "react-apollo";

export default ({ children, ...props }) => (
  <Query { ...props}>
    {({ loading, error, data }) => {
      if (loading) { return 'loading' };
      if (error) return `Error!: ${error}`;
      return children(data);
    }}
  </Query>
);
```



OUR CODE NOW:

```
import React from "react";
import Modal from "react-modal";

import { Query } from "../components/";
import { GET_MODAL } from "../queries/";

export default () => (
  <div>
    <Query query={GET_MODAL}>
      ({ { modal: { modalIsOpen } }) => (
        <Modal isOpen={modalIsOpen}>
          <h1>Of course they do ... </h1>
        </Modal>
      )}
    </Query>
  </div>
);
```

# FITS IN ONE SLIDE





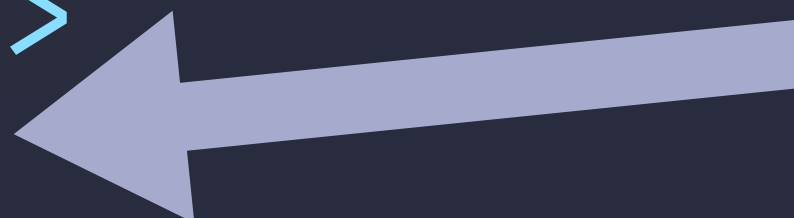


**BUT NO MODAL :(**

# THE MUTATION COMPONENT

```
import React from "react";
import { Mutation } from "react-apollo";
import { OPEN_MODAL } from "../queries/";
```

```
export default () => (
  <Mutation mutation={OPEN_MODAL}>
    {openModal => (
      <button onClick={openModal}>
        Wait... Everyone loves modals right ?
      </button>
    )}
  </Mutation>
);
```



**Wat?**



STILL THINK IT'S TOO  
MUCH CODE ?

# APOLLO CONSUMER

*Used for very simple mutations to the cache*

```
import React from "react";
import { ApolloConsumer } from "react-apollo";

export default () => (
  <ApolloConsumer>
    {cache => (
      <button onClick={() => cache.writeData({
        data: { modal: { isOpen: true } }
      })}
      >
        Wait... Everyone loves modals right ?
      </button>
    )}
  </ApolloConsumer>
);
```

# WHERE IS THE MUTATION?

*It's done on the fly ! No need to define it before 🎉*



CAN YOU MAKE THE QUERY AND  
MUTATION IN ONE SLIDE ME

**KEN... HOLD MY BEER**

```

const MODAL = gql`
  {
    modalOpen @client
  }
`;

export default () => (
  <Query query={MODAL}>
    ({ { data, client } }) => (
      <Fragment>
        <button
          onClick={() =>
            client.writeData({
              data: { modalOpen: true }
            });
          }
        >
          A MODAL
        </button>
        <Modal isOpen={data.modalOpen}>YEAH SON</Modal>
      </Fragment>
    )}
  </Query>
);

```

THE MODAL DOES NOT  
CLOSE



FINE...

```
const changeState = (client, value) =>
  client.writeData({
    data: { modalOpen: value }
  });

export default () => (
  <Query query={MODAL}>
    ({ data, client }) => (
      <Fragment>
        <button onClick={() => changeState(client, true)}>A MODAL</button>
        <Modal
          isOpen={data.modalOpen}
          onRequestClose={() => changeState(client, false)}>
          YEAH SON
        </Modal>
      </Fragment>
    )}
  </Query>
);
```

DEMO:

[HTTPS://CODESANDBOX.IO/S/  
4X2PLK2V3W](https://codesandbox.io/s/4x2plk2v3w)

DEMO WITH VARIABLES AND REMOTE DATA:

[HTTPS://CODESANDBOX.IO/S/  
05Z509VRP9](https://codesandbox.io/s/05z509vrp9)







A black cat is sitting on the left side of the frame, looking towards the right. Next to it is a white unicorn plush toy with a gold horn and a rainbow-colored mane. The background is a dark, textured surface with a repeating pattern of small, raised circles.

# THANK YOU 🇳🇱

<https://link-state-is-dope.now.sh/>