LEARNING GRAPHQL PART 1: DX

JOHN FAY

WANT TO FOLLOW ALONG?

keonik.github.io/learn-graphql

AGENDA

- What is graphql?
- (Pros) What does it solve?
- (Cons) What issues does it cause?
- How will it help Mile Two
- How it will help your professional development
- Demo

ABOUT ME

- 5 months of GraphQL exposure
- 2 month head down learning

LEARNING NEW TECH

1. Dismissal

> One more JavaScript library?! Just use jQuery already!

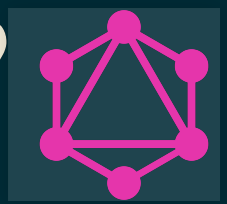
2. Interest

> Hmm, maybe I **should** check out this new library I keep hearing about…

3. Panic

> Help! I need to learn this new library **right now** or I'll be completely obsolete!

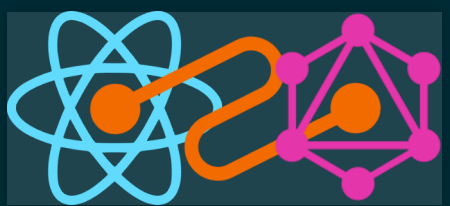
WHAT IS GRAPHQL?



Made by

React

Relay



A query language for your API

PROBLEM

GET /genres

```
[
    {
        name: 'Mystery'
    },
    {
        name: 'Romance'
    }
]
```

```
[
    {
        name: 'Mystery'
        books: [
            {
                title: 'The lost man',
                author: 'Jane Harper'
            },
            {
                title: 'And then there were none'
                author: 'Agatha Christie'
            }
        ]
    },
]
```

DESCRIBE YOUR DATA

```
type Genre {
    name: String
    books: [Book]
}
```

ASK FOR WHAT YOU WANT

```
genre(name: "Mystery"){
    name
    books{
        title
        author
    }
}
```

GET PREDICTABLE RESULTS

```
genre{
    name: 'Mystery'
    books: [
        {
            title: 'The lost man',
            author: 'Jane Harper'
        },
        {
            title: 'And then there were none'
            author: 'Agatha Christie'
        }
    ]
}
```

WHAT DOES IT SOLVE?

- Client driven instead of server driven
- Development speed improvements
- Self documenting
- Single trip data fetching

WHAT ISSUES DOES IT CAUSE?

- Superfluous Database Calls `N+1`
  - Solution: dataloader
- Overkill for small applications
  - Ensure you would benefit from it. More emphasis on frontend development.
- Time to relearn another thing even though REST works

WHAT ARE THE PEOPLE SAYING?

State of js 2019

WHO'S USING IT IN PRODUCTION?

AirBnB, GitHub, PayPal, Lyft, Starbucks, The New York Times, Twitter, Yelp

GRAPHQL FUNDAMENTALS

QUERIES

```
GET

useQuery, useLazyQuery
```

MUTATIONS

```
PUT, PATCH, POST, DELETE

useMutation
```

QUERIES

```
GET /books

    query {
        books{
            ...
        }
    }

GET /book/1

    query {
        book(id: 1) {
            ...
        }
    }
```

MUTATIONS

```
POST book?name=Storyteller

    mutation {
        createBook(
            {
                name: 'Storyteller'
            }
        ){
            ... what you want to return
        }
    }
```


NOW LETS SHOW THAT AWESOME FRONTEND
DX(DEVELOPER EXPERIENCE)

THINGS YOU'LL NEED

```
npm install apollo-boost graphql @apollo/react-hooks
```

APOLLO-BOOST - ZERO CONFIG APOLLO CLIENT

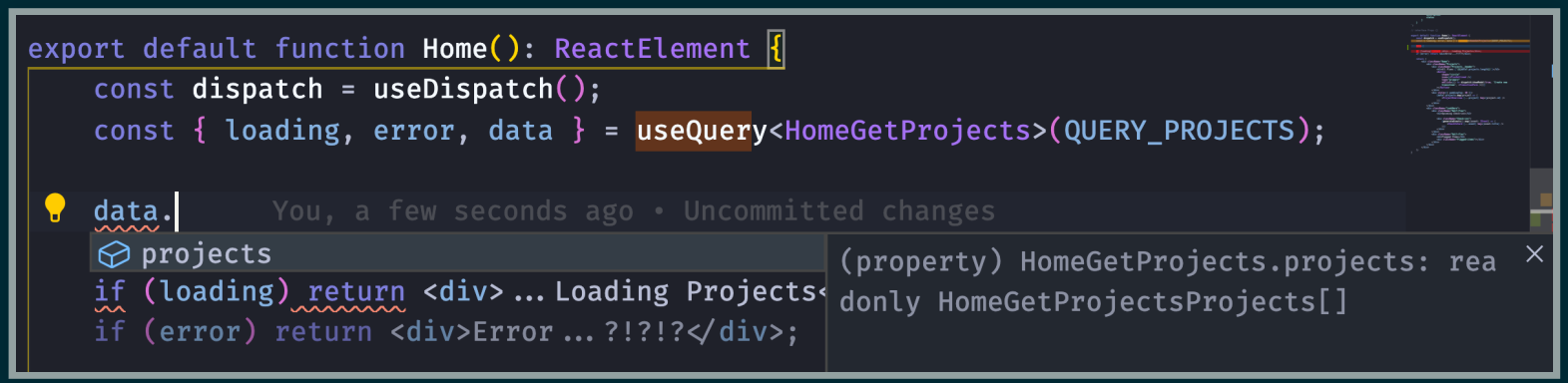
@APOLLO/REACT-HOOKS - HOOKS FOR YOUR QUERIES/MUTATIONS

GRAPHQL - WRITE QUERIES LIKE SO:

```
gql`
    query{
        users(id: 1){
            id
            name
        }
    }
`
```

SOME OTHER PERKS

AUTO GENERATED TYPE SAFE RESPONSES



THE END OF PART 1. STAY TUNED FOR API WORK!

LEARN GRAPHQL PART 2: API TIME!

THINGS YOU'LL NEED

```
npm install apollo-server pg reflect-metadata typeorm type-graphql
```

APOLLO-SERVER

-Server...

PG

Database - postgresql

REFLECT-METADATA

Makes the decorator @ recognized for classes

TYPEORM

Object relational mapping for NodeJS with latest javascript features

TYPE-GRAPHQL

Makes it so you can do graphql stuff but typed

CREATE A MODEL

```
export class User {
    id
    firstName
    lastName
}

export class UserTyped {
    id: number
    firstName: string
    lastName: string
}
```

You'll use these across projects so typing has it's benefits

CREATE ENTITY

```
import { Entity } from 'typeorm'

@Entity()
export class User {
    id: number
    firstName: string
    lastName: string
}
```

Tells server that this links to our orm

ADD TABLE COLUMNS

```
import { Entity, Column, PrimaryGeneratedColumn } from 'typeorm'

@Entity()
export class User {
    @PrimaryGeneratedColumn()
    id: number

    @Column()
    firstName: string

    @Column()
    lastName: string
}
```

Project is aware of database column

ADD TYPE-GRAPHQL DECORATORS

```
...
import {ObjectType, ID, Field} from 'type-graphql';

@ObjectType()
@Entity()
export class User {
    @Field(() => ID)
    @PrimaryGeneratedColumn()
    id: number

    @Field(() => String)
    @Column()
    firstName: string

    @Field(() => String)
    @Column()
```

Lets graphql query the User object and their fields