When Front-End Met Back-End

A GraphQL Love Story

I’m Jake 👋

@jakedawkins

Meet Client

Meet Client

- Display
- Loading assets
- Orchestrating interactions
- First line of defense
- Not good at serious data storage or processing

Meet Server

@jakedawkins

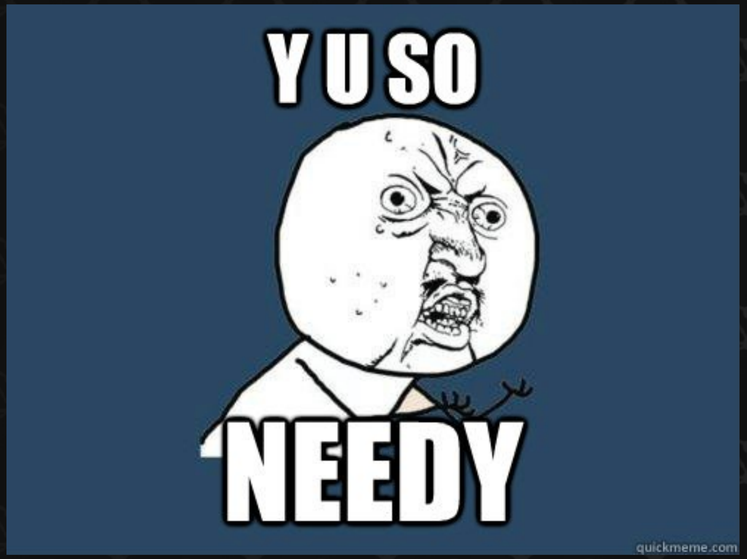
Meet Server

- Workhorse - great at processing data
- Methodical
- Useless without someone to display the data

So… They're Perfect for One Another

Client is Needy

- Multiple Requests
- Needs complicated data
- Needs data from other sources

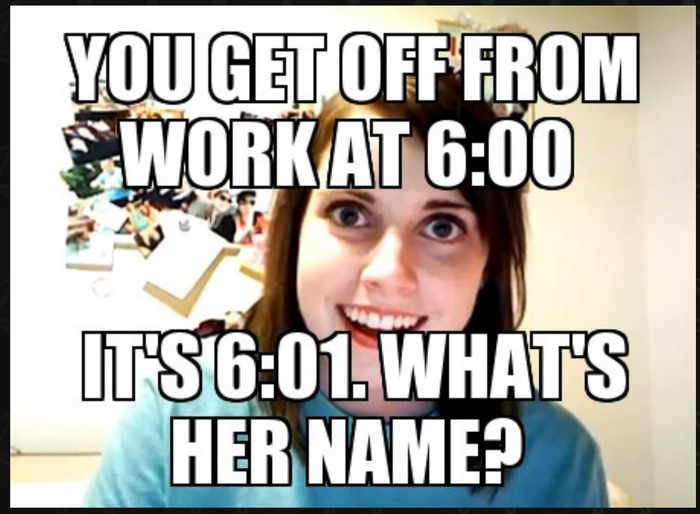


Client is Tired

- Rendering work
- Deals with people all day
- Doesn’t like data processing

Server is Controlling

- Front-end needs information from others
- Wants to reduce conflicting data



Server is Confusing

- Documentation
  - either out of date or missing
- Changes its mind

Server is Fragile

- Many reasons a server can fail
- Hard to pinpoint errors
- Many points of possible failure

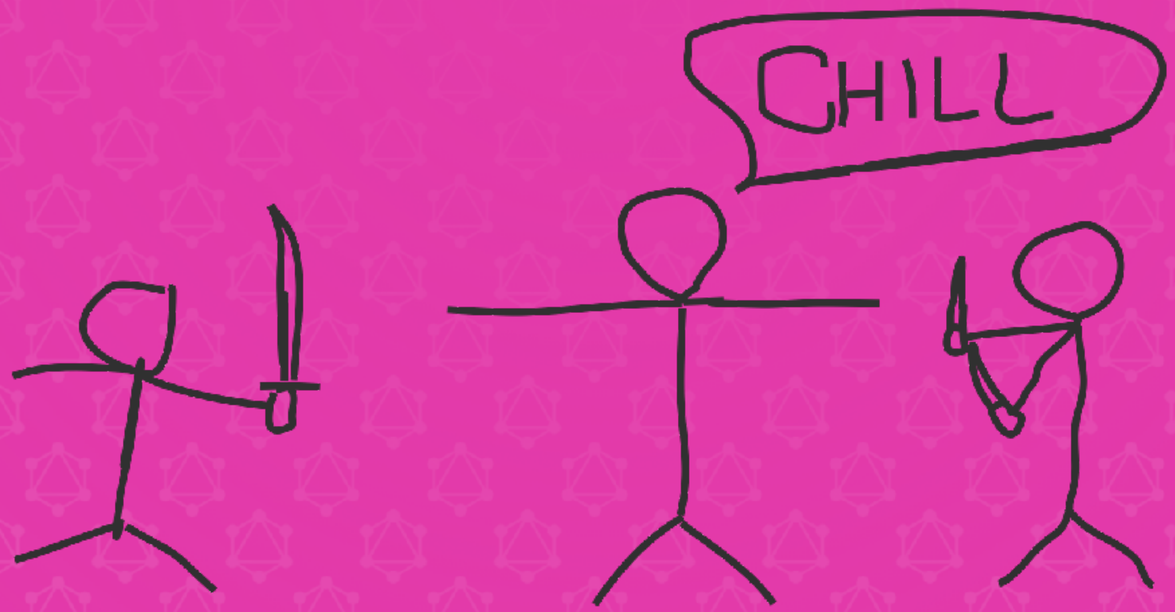
GraphQL

A means of resolving the communications differences between the client and server

@jakedawkins

GraphQL

A means of resolving the communications differences between the client and server



@jakedawkins

Client is Needy

GraphQL is Understanding

- Client can ask for exactly what it wants
- Multiple entities in a single request
- Nested and recursive data shapes

Aliased Field

```
# don't ask for what
# you don't need!
{
  player {
    first_name
    last_name
  }
  club {
    name {
      abbreviation
    }
  }
  player2: player {
    known_name
    birth_date
  }
}
```

@jakedawkins

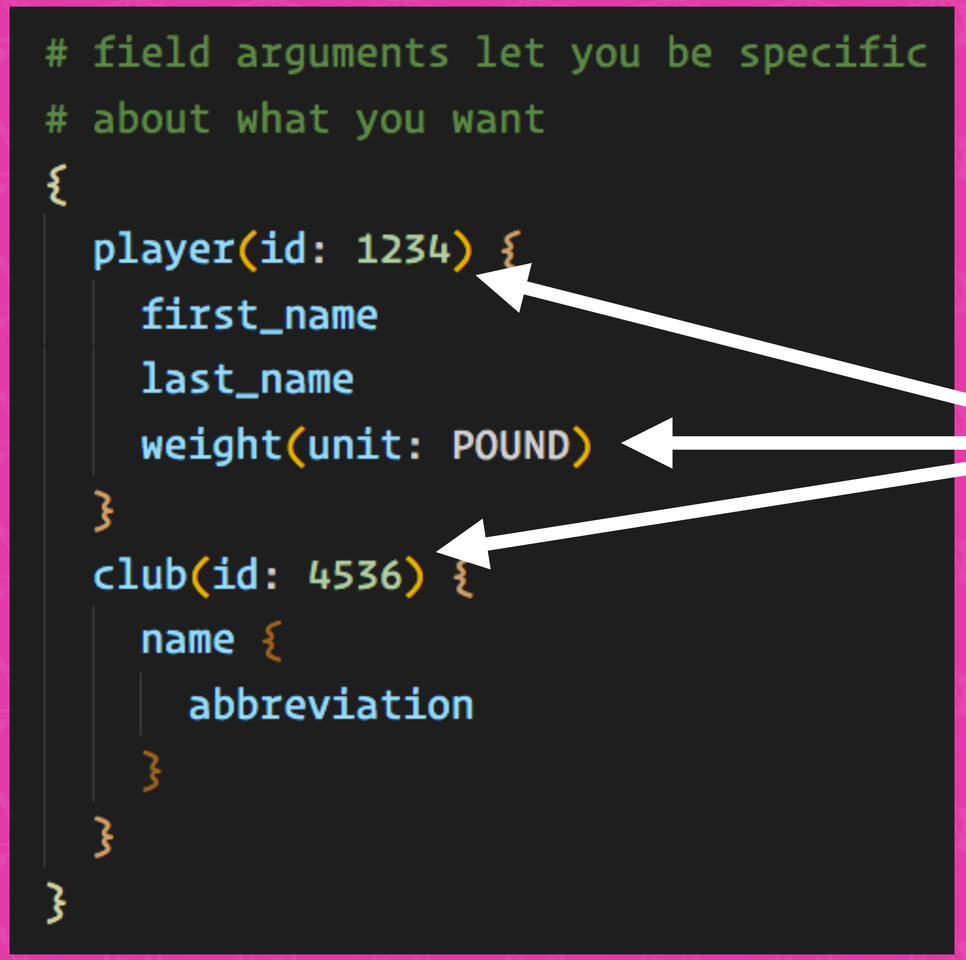
```
# recursive query
# see a person's groups and
# everyone they're in a group with
{
  person {
    first_name
    groups {
      members {
        first_name
      }
    }
  }
}
```

@jakedawkins

Client is Tired

GraphQL is Hardworking

- Client only gets what it needs
- Field arguments allow the front-end to be very specific about what it wants
- Bonus: Client knows the response data shape




```
# field arguments let you be specific
# about what you want
{
  player(id: 1234) {
    first_name
    last_name
    weight(unit: POUND)
  }
  club(id: 4536) {
    name {
      abbreviation
    }
  }
}
```

@jakedawkins

Server is Controlling

GraphQL is Connectable

- Aggregating data from multiple sources is simple and done on a per-field basis
- Allows for a single server to be our source of truth
- Bonus: 3rd party API changes don’t require app updates.

Request

```
{
  likes(handle: "jakedawkins") {
    twitter
    facebook
    instagram
  }
}
```

Response

```
"data": {
  "likes": {
    "twitter": 1000,
    "facebook": 0,
    "instagram": 999
  }
}
```

@jakedawkins

Whatever the parent returns

Field arguments

Context (business logic/auth/etc)

```
async likes(parent, { handle }, { models }){
  const [twitter, facebook, instagram] = await Promise.all([
    models.twitter.getLikes(handle),
    models.facebook.getLikes(handle),
    models.instagram.getLikes(handle),
  ]);
  return models.reducers.likes({
    twitter,
    facebook,
    instagram
  })
}
```

@jakedawkins

Server is Confusing

GraphQL is Self-Documenting

- Fields are typed
- No need for multiple endpoints
- Versioning is rarely needed because API changes don’t have to be breaking changes
- Everything is available to introspection

All Fields

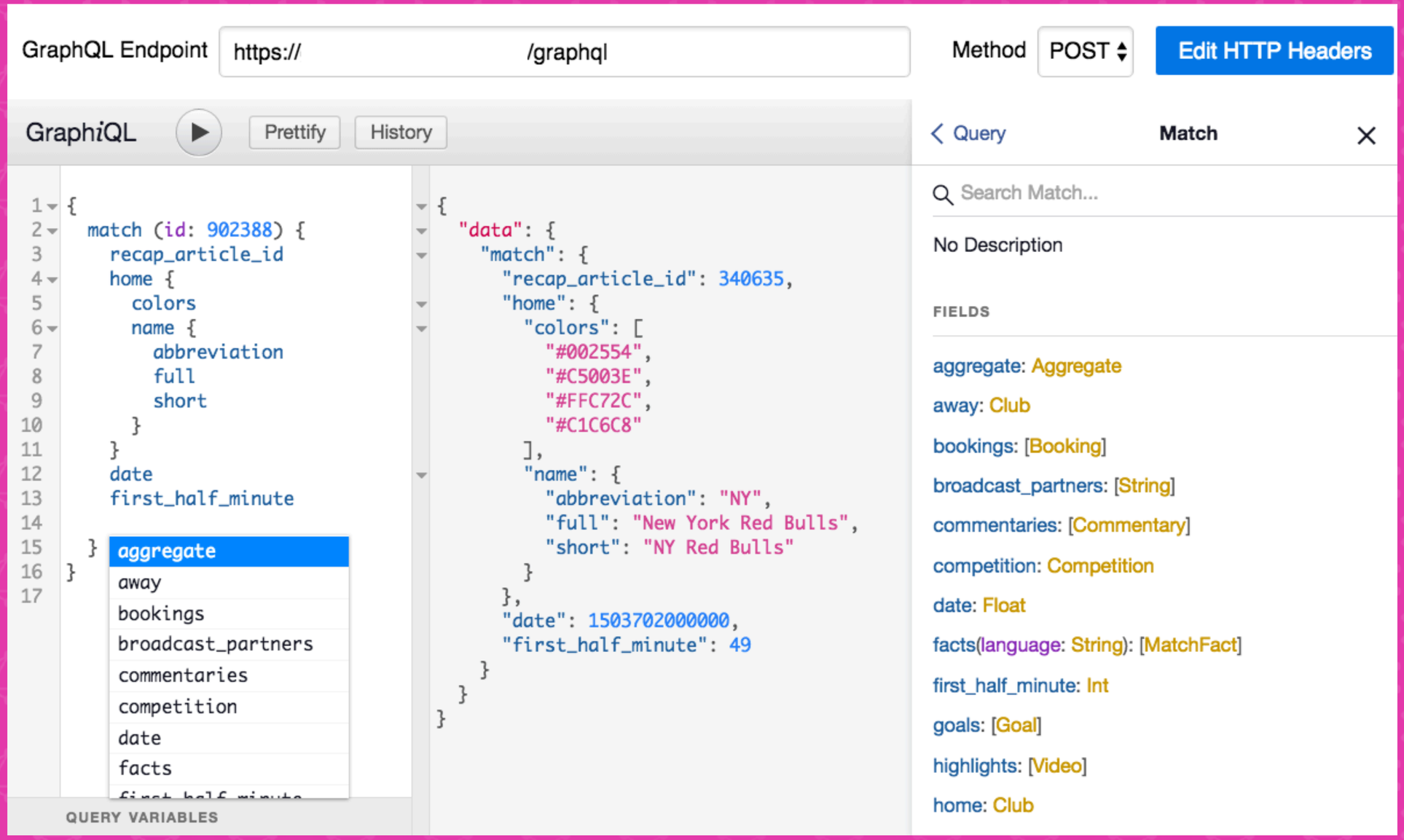
Field Arguments

Argument Types

```
# introspection of a server's schema
{
  __schema {
    queryType {
      name
      fields {
        name
        description
        deprecationReason
        isDeprecated
        args {
          name
          defaultValue
          type {
            kind
            ofType {
              name
            }
          }
        }
      }
    }
  }
}
```

```
{
  "data": {
    "__schema": {
      "queryType": {
        "name": "Query",
        "fields": [
          {
            "name": "match",
            "description": "",
            "deprecationReason": null,
            "isDeprecated": false,
            "args": [
              {
                "name": "id",
                "defaultValue": null,
                "type": {
                  "kind": "NON_NULL",
                  "ofType": {
                    "name": "Float"
                  }
                }
              }
            ]
          },
```

@jakedawkins



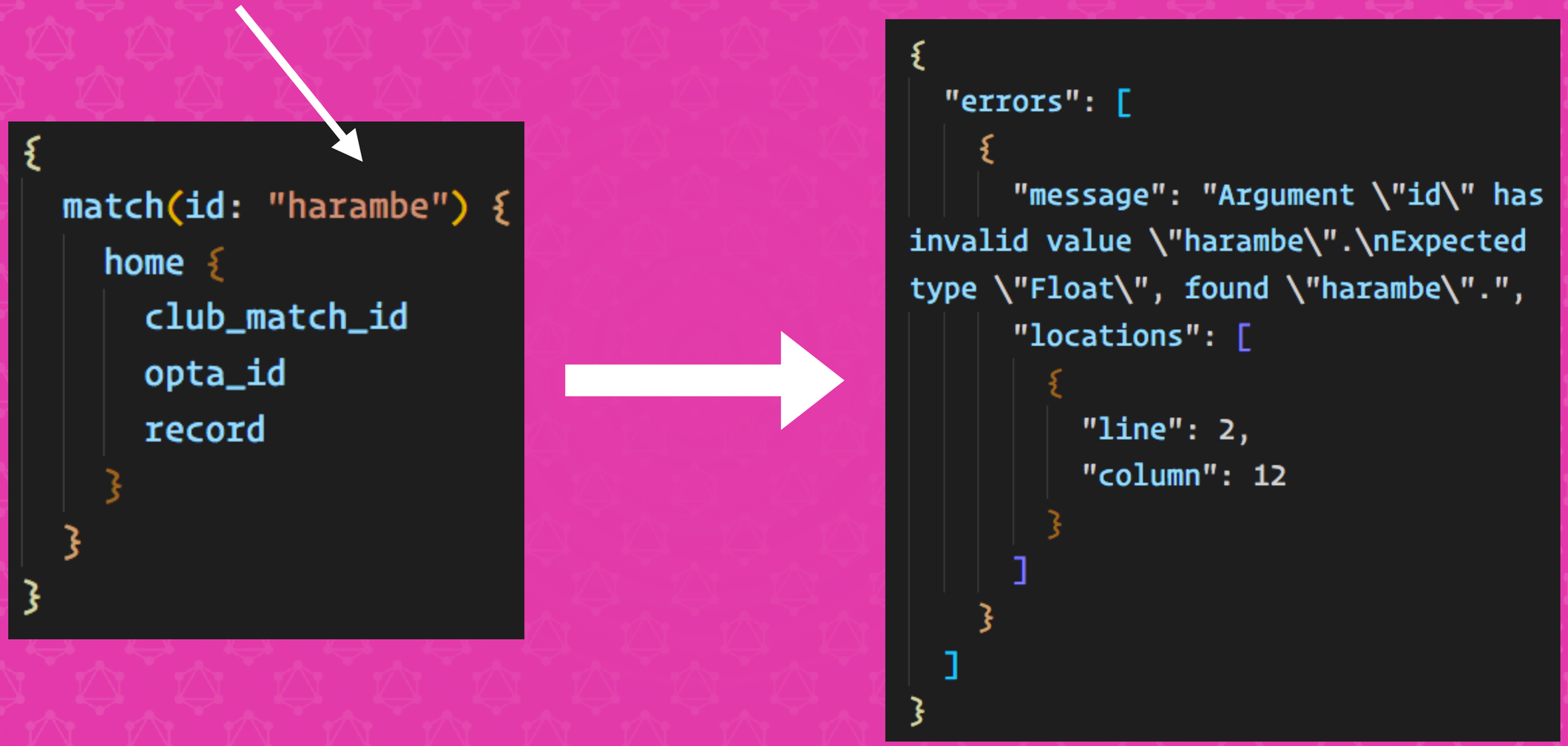
Request **Response** **Documentation** 🎉

Server is Fragile

GraphQL is Flexible

- Errors are returned like regular data with clear descriptions
- Everything is nullable by default
- Incoming queries are evaluated against syntax and schema rules

Incorrect Argument Type



@jakedawkins

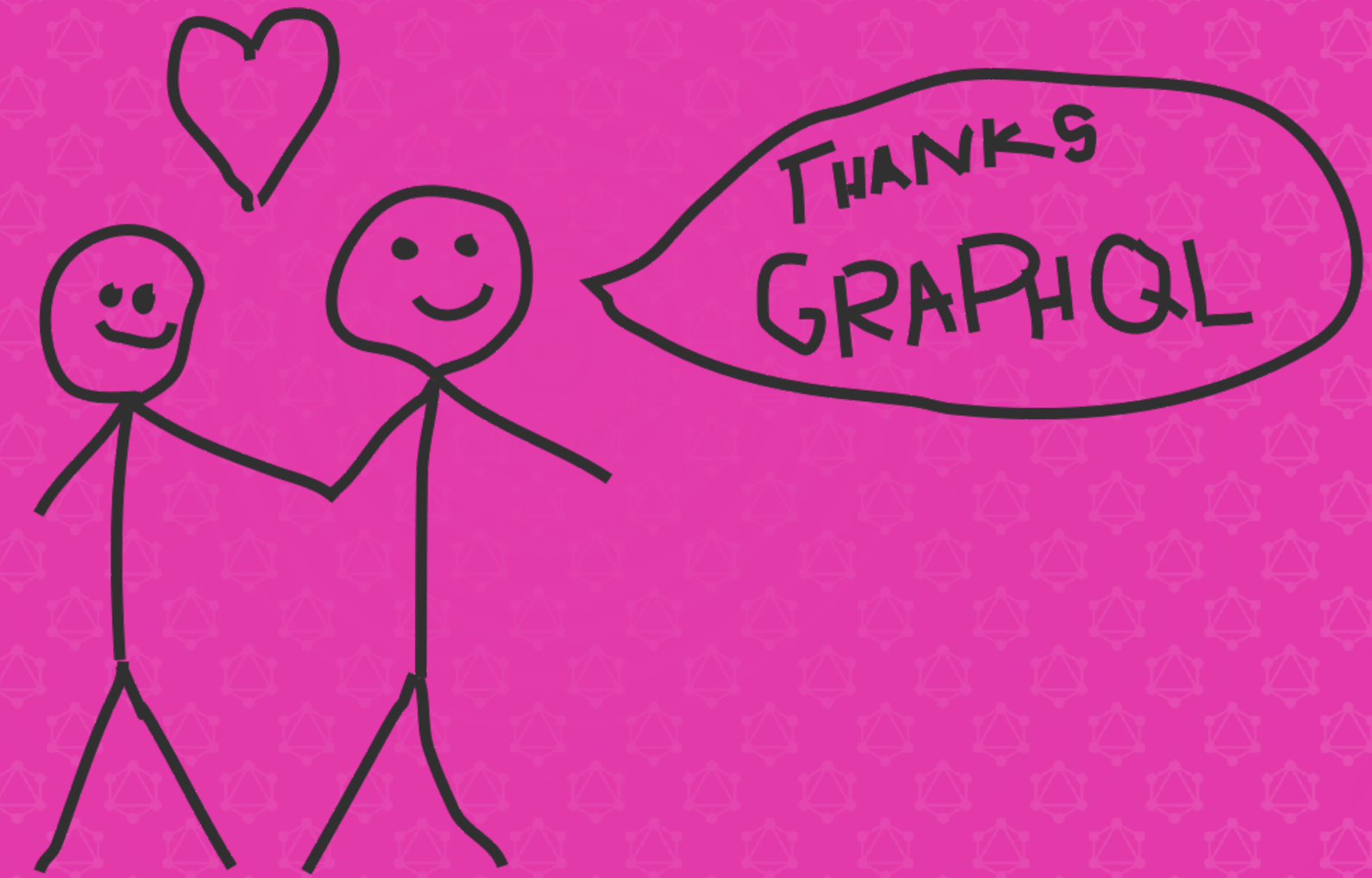
Client is needy. GraphQL is understanding.

Client is tired. GraphQL is hardworking.

Server is controlling. GraphQL is connectable.

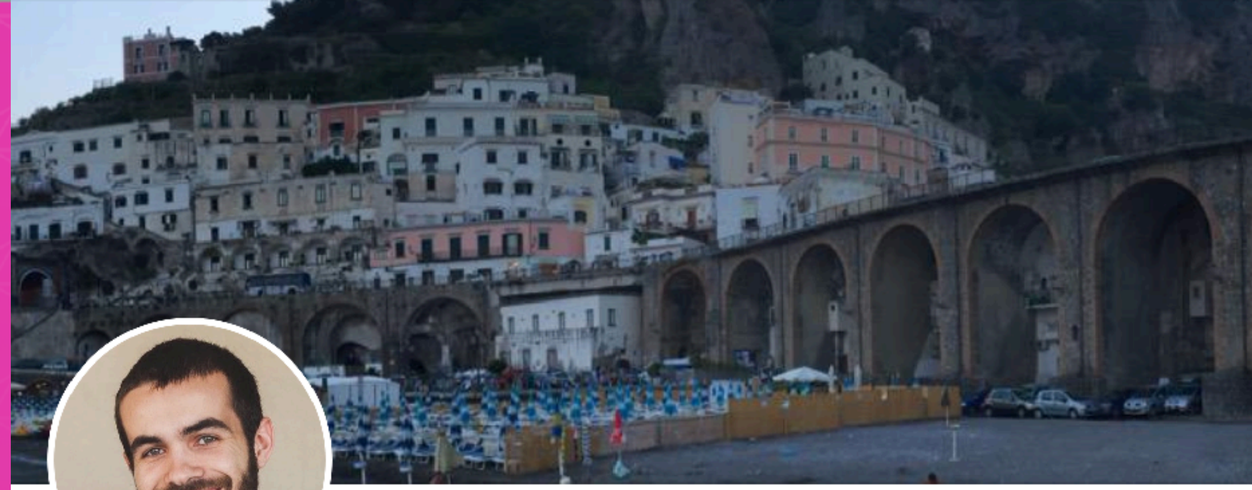
Server is confusing. GraphQL is self-documenting.

Server is fragile. GraphQL is flexible.



@jakedawkins

@jakedawkins

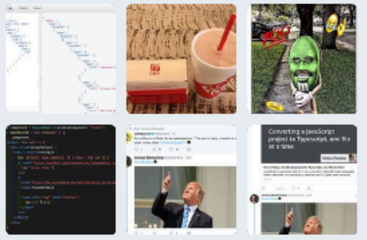


Tweets 2,644 | Following 419 | Followers 451 | Likes 2,956 | Lists 0 | Moments 0

Jake Dawkins
@JakeDawkins

Lead Software Engineer @MLS. React/Node/GraphQL. Purveyor of fine memes.

New York, NY
jakedawkins.com
Joined August 2008
Born on August 11, 1994
314 Photos and videos



Tweets | Tweets & replies | Media

Jake Dawkins @JakeDawkins · 9h
Ever wonder why half the code/screenshots I post have Harambe references in them? Well wonder no more. I wrote about it (and more) 🔥

> **Jake Dawkins** @JakeDawkins
> I just published “Make Coding Fun” medium.com/p/make-coding-…

1 3

Jake Dawkins @JakeDawkins · 9h
I just published “Make Coding Fun”

Make Coding Fun – Jake Dawkins – Medium
I love puzzles. I love memes. Stick figures are the peak of my artistic abilities. These are the reasons I love coding.
medium.com


Business Cat
Thanks you
quickmeme.com