



DevFest  
Siberia 2020

DEVFEST  
SIBERIA

5-6th  
DECEMBER

# GraphQL Introspection

Also just GraphQL, why not

Igor Dultsev <[igor.dultsev@global-fashion-group.com](mailto:igor.dultsev@global-fashion-group.com)>





A little bit about me

My name is Igor

I do this development thingy

I have twitter ([@yhaskell](#))



# A little bit of history

I love history slides in my presentations

## First came HTTP\*

Nice protocol for sending text through Inter-webs

You can GET and you can POST!

Later, more verbs were added

HTML is sent through; CGI for interactivity and processing

\* This is a lie



# A little bit of history

I love history slides in my presentations

## SOAP\*

Formerly XML-RPC

Sending messages, commands etc through  
network

XML: Validateable, strictly typed

Goes good with HTTP

\* If this was a jump-scare, would it work?



# A little bit of history

I love history slides in my presentations

## REST\*

Links formatted in a specific way

HTTP verbs determine actions

Accepts JSON, responds with JSON

\* Praise our lord and saviour



REST is the way...

But there is some caveats



**WHAT DO WE WANT?**



**DOCUMENTED API  
THAT ALWAYS WORKS**



**WHEN DO WE WANT IT**



**ALWAYS!**



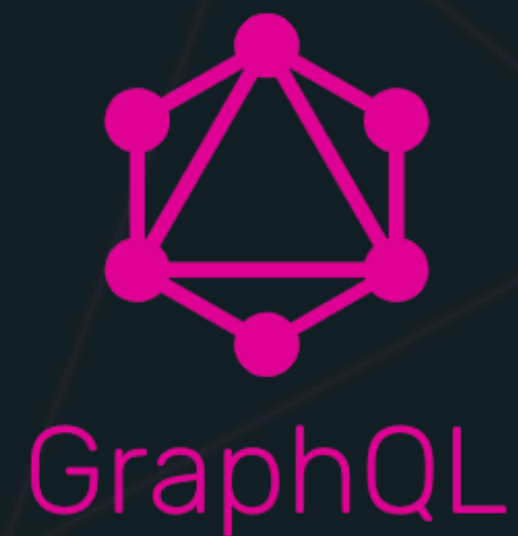


# What do we miss in REST

Personal opinions presented as the only and final truth

- Automatic validation of incoming data
- Therefore, strict typing for endpoints
- A nice way to get data for multiple objects in one go
- Getting only the data we need





Describe your data

```
type Project {  
  name: String  
  tagline: String  
  contributors: [User]  
}
```

Ask for what you want

```
{  
  project(name: "GraphQL") {  
    tagline  
  }  
}
```

Get predictable results

```
{  
  "project": {  
    "tagline": "A query language for APIs"  
  }  
}
```

\* Stolen from official website, which is, by the way, <https://graphql.org/>



# So, What is GraphQL?

A query language for your API

Also, a runtime to fulfil the queries

Self-description built into spec

Strongly typed

Validation is in the spec too

Protocol-independent



# GraphQL Query

- 3 types: query, mutation, subscription
  - queries provides idempotent data access
  - mutations, well, mutate data
  - subscriptions encapsulate data streaming



# GraphQL Query

```
query {  
  distilleries(country: SCOTLAND) {  
    id  
    name  
    address { address zip city }  
    phone  
  }  
}
```



# GraphQL Query

```
{  
  "data": {  
    "distilleries": [  
      {  
        "id": "s7-cXydY2js",  
        "name": "Lagavulin",  
        "address": {  
          "street": null,  
          "city": "Lagavulin, Isle of Islay",  
          "zip": "PA42 7DZ"  
        },  
        "phone": "+44 149 6302 749"  
      },  
      ...  
    ]  
  }  
}
```



# GraphQL Mutation

```
mutation MakeOrder {  
  orderItems(items: [  
    { productId: "laajKUXbM0s", count: 2 },  
    { productId: "i7m-2dUWSn4", count: 7 }  
  ]) {  
    id  
    price  
  }  
}
```



# GraphQL Mutation

```
{  
  "orderItems": {  
    "id": "cJ9-dc7Snekc",  
    "price": 576.24  
  }  
}
```



# GraphQL Schema

Multiple type kinds:

- Scalar
- Union
- Object
- Enum
- Input
- Interface
- NonNull
- List



# GraphQL Scalar Type

`scalar Date`

(Processing code must be defined in the server code)



# GraphQL Object Type

```
type User {  
  id: ID!  
  name: String!  
}
```



# GraphQL Enum Type

```
enum State {  
  INITIALIZED  
  PROCESSING  
  ERROR  
  SUCCESSFUL  
}
```



# GraphQL Interface Type

```
interface Response {  
  success: Boolean!  
}
```



# GraphQL Interface Type

Now we can extend the interface to some object types

```
type ErrorResponse extends Response {  
  success: Boolean!  
  errors: [Error!]  
}
```

```
type SuccessUserCreatedResponse extends Response {  
  success: Boolean!  
  user: User!  
}
```



# GraphQL Union Type

```
type Leaf {  
  data: Int!  
}
```

```
type Node {  
  left: Tree  
  right: Tree  
}
```

```
union Tree = Leaf | Node
```




# GraphQL Additional Types

List: `[T]`

Non null: `T!`





Let's play with some GraphQL



# GraphQL Introspection

Reflection for your API

Schema data accessible via GraphQL

Required to be available by spec





Let's play with some more GraphQL





DevFest  
Siberia 2020

DEVFEST  
SIBERIA

5-6th  
DECEMBER



# Thank you!

Igor Dultsev

[igor.dultsev@global-fashion-group.com](mailto:igor.dultsev@global-fashion-group.com)

Twitter: @yhaskell

<https://github.com/yhaskell/gdg-siberia-2020>