GraphQL: A query language for your API

Nathan Smith
Lead Platform Engineer @ Aunalytics
2/16/17 for Local Variables

What is GraphQL?

"GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools."

It is also a standardized language for describing entity-relationship models and the basis for a rich ecosystem of tools, libraries and frameworks.

What isn't GraphQL?

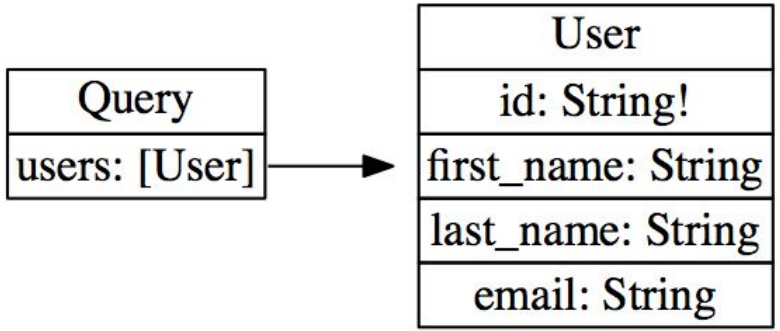
- A database abstraction (although you could make one)
- It is not a DB query language (unless the DB or a layer on top of it supports it)
- It is not a complete server framework (although there are many plugins for existing frameworks)
- It does not solve all your problems (although it does solve many).

Concepts

Schema

The schema is the API contract that describes what is available for interaction. It supports the following types:

- Scalars
  - Int
  - Float
  - String
  - Boolean
- Custom Objects
- Lists
- Non-Null
- Enum
- Plus a few other more exotic ones

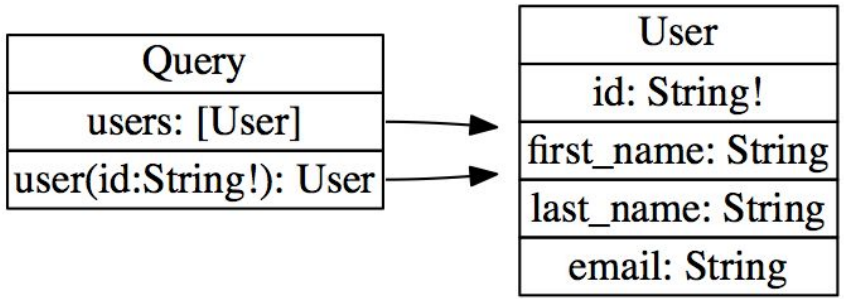


Operations

- Query: read-only fetch
- Mutation: a write followed by a fetch

Query

The query entity is a root entity in the ERM and the place where any data fetching begins. Relationships must exist to other entities to make them query-able. There can be multiple query entities.



Query: Selection Set

A set of selections on an entity which can be Fields, FragmentSpreads or InlineFragments.

```
query {
  users {
    id
    first_name
    last_name
    email
  }
}
```

Query: Field

"A selection set is primarily composed of fields. A field describes one discrete piece of information available to request within a selection set."

"Some fields describe complex data or relationships to other data. In order to further explore this data, a field may itself contain a selection set, allowing for deeply nested requests. All GraphQL operations must specify their selections down to fields which return scalar values to ensure an unambiguously shaped response."

Query: Argument

Some fields require arguments to be fetched, which may be required or optional. Arguments are unordered and have types.

```
query {
  user(id: "1") {
    id
    first_name
    last_name
    email
  }
}
```

Query: Field Alias

It is possible to rename a field in the result via a field alias. This allows the fetching of multiple result of the same field while varying the field arguments.

```
query {
  user1: user(id: "1") {
    email
  }
  user2: user(id: "2") {
    email
  }
}
```

Query: Fragment

There are several kinds of fragments, but essentially fragments allow the user to declare a reusable selection set and use it on multiple fields of similar types.

```
{
  users {
    ...userFields
  }
  user(id: "1") {
    ...userFields
  }
}

fragment userFields on User {
  id
  first_name
  last_name
  email
}
```

Query: Variable

“A GraphQL query can be parameterized with variables, maximizing query reuse, and avoiding costly string building in clients at runtime.”

```
query userWithVariable($uid: String!) {
  user(id: $uid) {
    id
    first_name
    last_name
    email
  }
}
```

```
{
  "uid": "1"
}
```

Introspection

In addition to being able to query data, the schema is also query-able. There is a standard query that is commonly used to get information about all entities and relationships in an ERM which many tools use to assist the user.

Mutation

In addition to querying, GraphQL also supports mutations by specifying a mutation root entity.

Tools

- http://nathanrandal.com/graphql-visualizer
- https://www.npmjs.com/package/graphqlviz
- https://github.com/skevy/graphiql-app
- https://github.com/chentsulin/awesome-graphql
- http://graphql.org/code/
- Spec: http://facebook.github.io/graphql/

Tools: GraphQLViz

```
$ graphqlviz --help

  GraphQL Server CLI visualizer

  Options:
    -a, --noargs     render without field arguments
    -v, --verbose    print introspection result
    -s, --sort       sort fields

  Usage
    $ graphqlviz [url]
        Renders dot schema from [url] endpoint

  Examples
    $ graphqlviz https://localhost:3000 | dot -Tpng -o graph.png
    $ graphqlviz http://graphql-swapi.parseapp.com | dot -Tpng | open -f -a Preview
    $ graphqlviz path/to/schema.json | dot -Tpng | open -f -a Preview
    $ cat result.json | graphqlviz | dot -Tpng | open -f -a Preview
```

Demo!!!

https://github.com/NathanRSmith/graphql-demo

Questions?