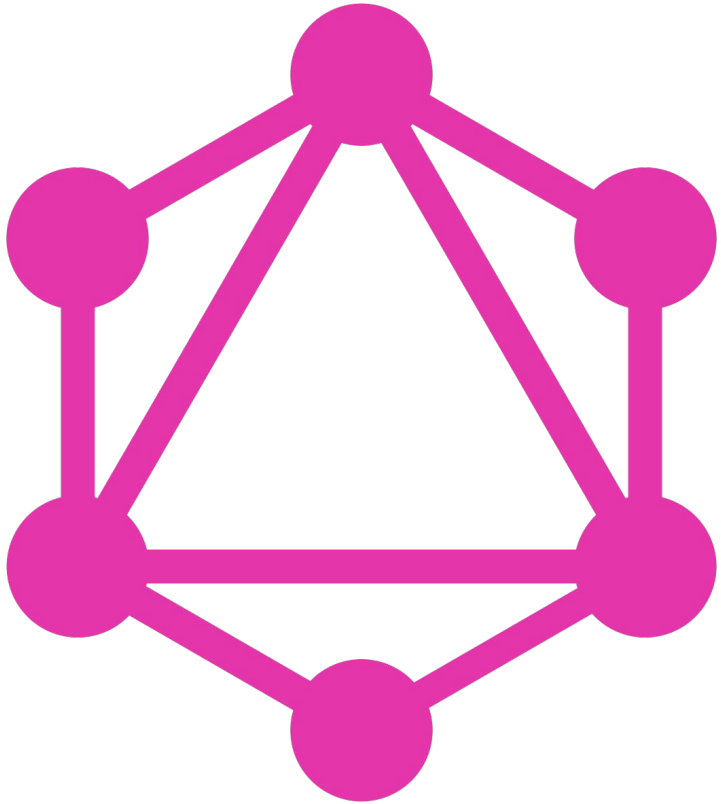
GraphQL

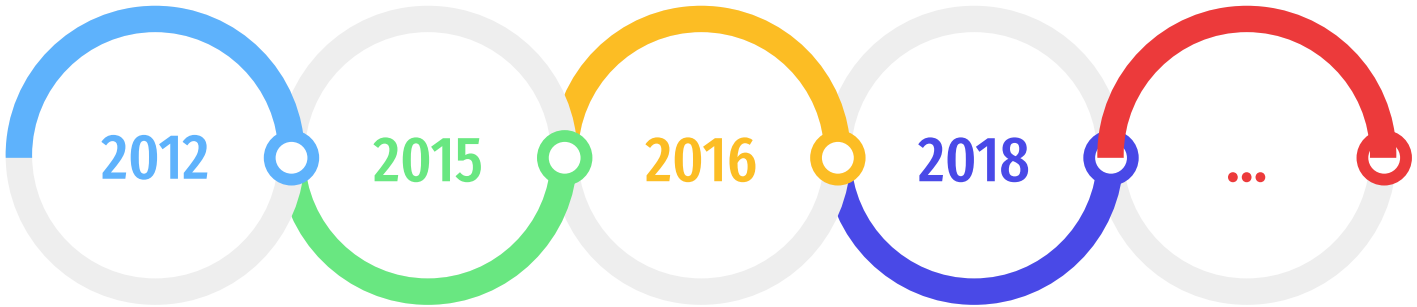
Breno Salles (up202103389)
Diogo Costa (up202100686)

What is?

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data.



History



Created

Was created by two employees of the Facebook platform team

Open Source

Facebook made it public

Production Ready

After it was made open source it evolved with the feedback and help of the community

Moved

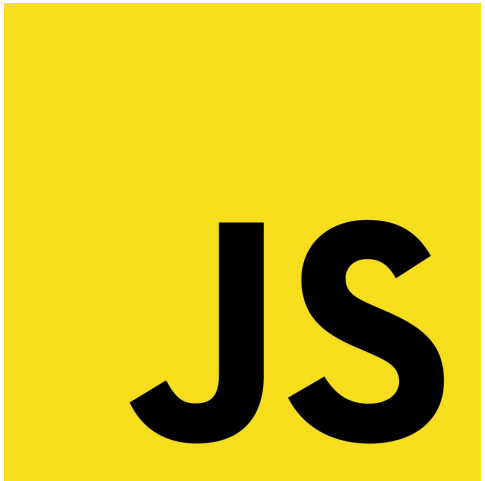
GraphQL Foundation now oversees the project

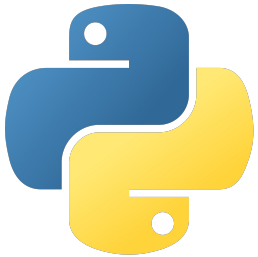
Today's adoption (adopters)

Widely used in production by more than 90 companies.

Today's adoption (languages)

23 languages supported





Today's adoption (tools)

Server

- GraphQL.js (19.2k stars)
- Apollo Server (13k stars)
- GraphQL Yoga (7.3k stars)
- GraphQL Helix (1k stars)

Client

- Apollo Client (18.3k stars)
- Relay (17.4k stars)
- AWS Amplify (9k stars)
- GraphQL request (5k stars)

Tools

- GraphiQL (14k stars)
- GraphQL Code generator (9k stars)
- Postgraphile (12k stars)
- GraphQLShield (3k stars)

Example

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"
  }
}
```

GraphQL Operations

Query

Read only fetch

Subscription

Long-lived request that fetches data in response to source events

Mutation

A write followed by a fetch

Fields

- A field describes one piece of information available to request within a selection set.
- Some fields can describe complex data or relationships to other data.

```
{
  hero {
    name
    # Queries can have comments!
    friends {
      name
    }
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        }
      ]
    }
  }
}
```

Arguments

- Arguments often map directly to function arguments within a GraphQL service's implementation.
- Every field and nested object can get its own set of arguments, making GraphQL a complete replacement for making multiple API fetches.
- Arguments can be of many different types (e.g. enumeration).

```
{
  human(id: "1000") {
    name
    height(unit: FOOT)
  }
}
```

```
{
  "data": {
    "human": {
      "name": "Luke Skywalker",
      "height": 5.6430448
    }
  }
}
```

Aliases

- By default a field's response key in the response object will use that field's name.
- Using alias you can define a different response key by specifying an alias.

```
{
  empireHero: hero(episode: EMPIRE) {
    name
  }
  jediHero: hero(episode: JEDI) {
    name
  }
}
```

```
{
  "data": {
    "empireHero": {
      "name": "Luke Skywalker"
    },
    "jediHero": {
      "name": "R2-D2"
    }
  }
}
```

Fragments

- Fragments allow for the reuse of common repeated selections of fields, reducing duplicate text in the document.
- Fragments are consumed by using the spread operator (...)

```
{
  leftComparison: hero(episode: EMPIRE) {
    ...comparisonFields
  }
  rightComparison: hero(episode: JEDI) {
    ...comparisonFields
  }
}

fragment comparisonFields on Character {
  name
  appearsIn
  friends {
    name
  }
}
```


```
{
  "data": {
    "leftComparison": {
      "name": "Luke Skywalker",
      "appearsIn": [
        "EMPIRE",
        "JEDI"
      ],
      "friends": [
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        }
      ]
    },
    "rightComparison": {
      "name": "R2-D2",
      "appearsIn": [
        "NEWHOPE",
      ],
      "friends": [
        {
          "name": "Luke Skywalker"
        }
      ]
    }
  }
```

Mutations

- Mutations provides a way to modify server-side data.
- A mutation can contain multiple fields, just like a query.

```
# Variables
{
  "ep": "JEDI",
  "review": {
    "stars": 5,
    "commentary": "This is a great movie!"
  }
}

mutation CreateReviewForEpisode($ep: Episode!, $review: ReviewInput!) {
  createReview(episode: $ep, review: $review) {
    stars
    commentary
  }
}
```

```
{
  "data": {
    "createReview": {
      "stars": 5,
      "commentary": "This is a great movie!"
    }
  }
}
```

How it works behind the scenes?

Queries

Runs in **parallel**.

Mutations

Run in **series**, which prevents race conditions.

https://www.graphql.com/articles/4-years-of-graphql-lee-byron

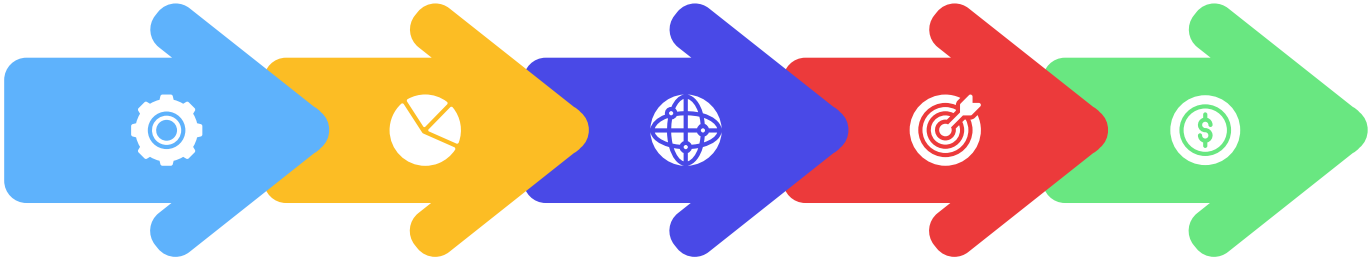
Best practices

Think in graphs, not endpoints

Unlike traditional APIs, GraphQL exposes all of the data from a single endpoint.

GraphQL is a thin interface

It's not intended to do things like authentication, authorization, caching, database query and optimization.



Naming Matters

As soon as a client start using a field, it becomes much harder to change the name.

Describe the data, not the view

Make sure that your API isn't linked too closely with the demands of your initial interface.

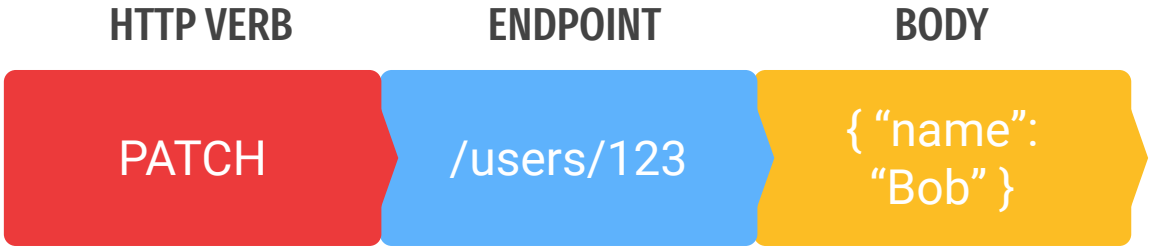
Hide implementation details

If a client has access to fields he should not, implementation changes could cause their code to break.

What is REST?

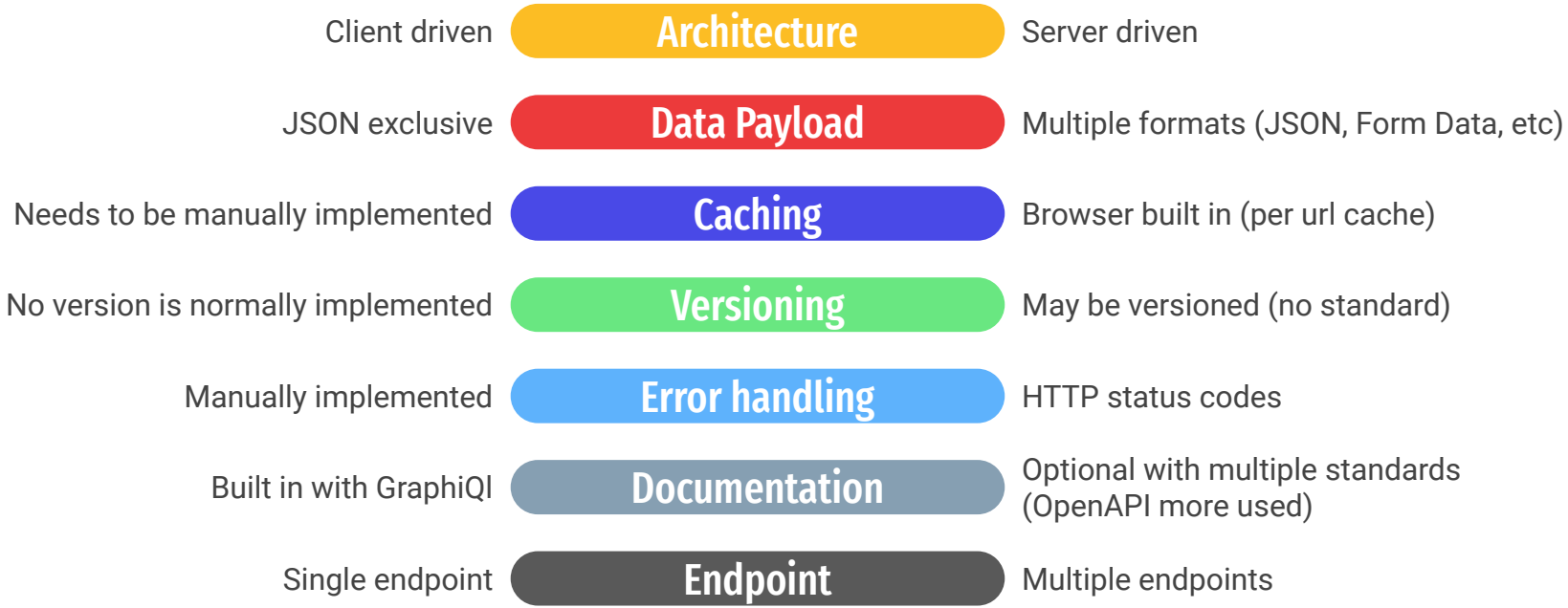
- Coined by Roy Fielding in 2000 with his PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures".
- REST is an architectural style.
- Mainly server oriented.
- Serves as an Application Interface between two separate separate components (server and client).
- Constraints, that when fully implemented, make an HTTP API RESTful.

How it works?



Endpoint/Method	GET	POST	PUT	PATCH	DELETE
/users	Fetches all users available	Creates a single user	(Normally irrelevant)	(Normally irrelevant)	(Normally irrelevant)
/users/[:id]	Fetches complete information of a single user	(Normally irrelevant)	Changes an ENTIRE user	Changes certain fields of an user	Deletes an user

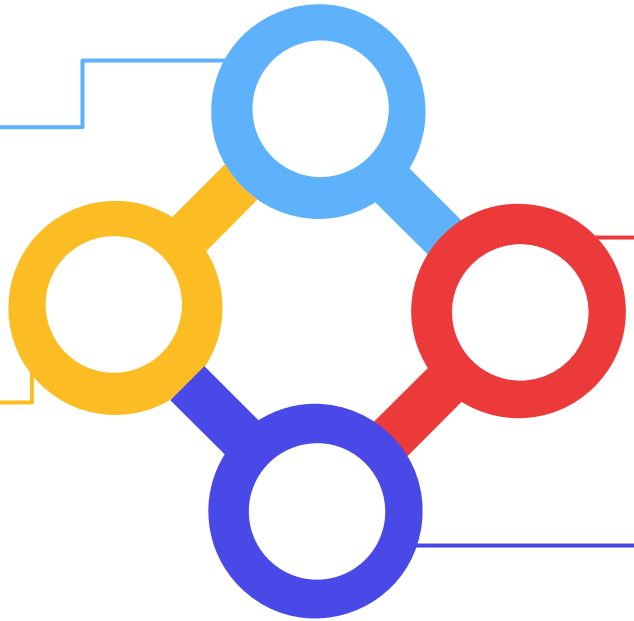
GraphQL vs REST



What is REST good for?

Error Reporting and Monitoring
Follows status code conventions

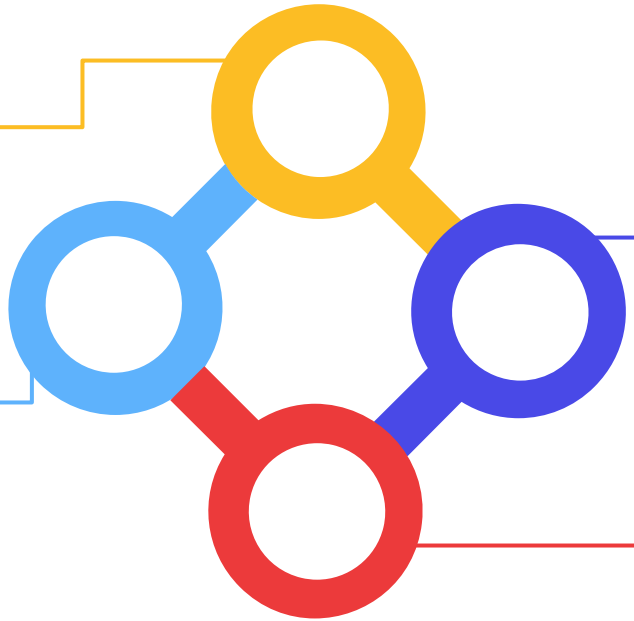
Security
Harder to overflow the system with complex queries



Caching
Browsers implement caching mechanism natively

Scaling
Easier horizontal scaling

What is GraphQL good for?



Documentation

Your code is your documentation

Simplicity

One api endpoint for all your queries

Client Oriented

Heavily directed to solve client issues

Atomic

Just get what you want, no more, no less

THANKS

Exercise available at https://github.com/diogofalken/cosn-graphql-demo