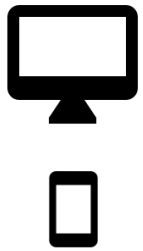
GraphQL

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.

1



HTTP GET

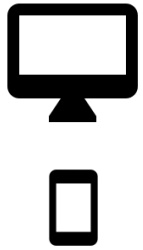
```
{
  “user”: {
    “id”: “er3tg439frjw”
    “name”: “Mary”,
    “address”: { … },
    “birthday”: “July 26, 1982”
  }
}
```

/users/<id>

/users/<id>/posts

/users/<id>/followers

2



HTTP GET

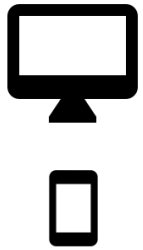
```
{
  “posts”: [{
    “id”: “ncwon3ce89hs”
    “title”: “Learn GraphQL today”,
    “content”: “Lorem ipsum … ”,
    “comments”: [ … ],
  }]
}
```

/users/<id>

/users/<id>/posts

/users/<id>/followers

3



```
{
  “followers”: [{
    “id”: “leo83h2dojsu”
    “name”: “John”,
    “address”: { … },
    “birthday”: “July 26, 1982”
  },
  …]
}
```

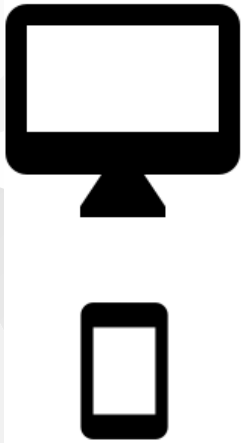
HTTP GET

/users/<id>

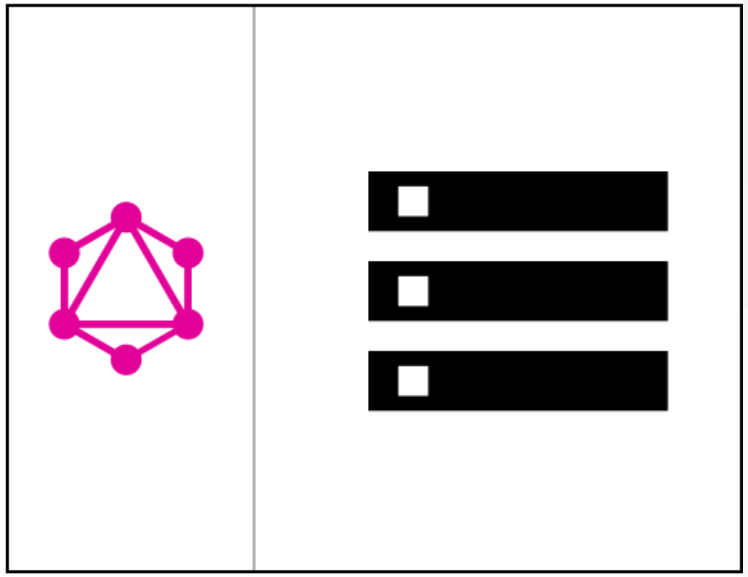
/users/<id>/posts

/users/<id>/followers

```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```



HTTP POST



```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

GraphQL History

- GraphQL was developed internally by Facebook in 2012.
- Facebook publicly released GraphQL as an open specification in 2015.
- GraphQL was moved to the newly-established GraphQL Foundation, hosted by the non-profit Linux Foundation, in 2018.

GraphQL vs REST

- REST is an architectural style for building APIs.
- GraphQL is often confused with being a database technology. This is a misconception, GraphQL is a query language for APIs - not databases.

Why GraphQL?

- **Ask for what you need, get exactly that**
- Get many resources in a single request
  - Underfetching
  - Overfetching
- Describe what's possible with a type system
- Move faster with powerful developer tools
- Evolve your API without versions

Core Concepts

- Query
  - Read operation

```
query {
  users {
    id
    username
    posts {
      id
      text
    }
  }
}
```

```
query {
  user(id: 1) {
    id
    username
    posts {
      id
      text
    }
  }
}
```

Core Concepts (cont'd)

- Mutation
  - Create, update, delete operation

```
mutation {
  createUser(username: "user1", password: "123456") {
    id
    username
  }
}
```

Core Concepts (cont'd)

- Subscription
  - Real-time operation

```
subscription {
  newUser {
    id
    username
  }
}
```

Type System

- GraphQL has its own type system that's used to define the schema of an API.

```
type User {
  id: ID!
  username: String!
  password: String!
  posts: [Post]
}

type Post {
  id: ID!
  text: String!
}
```

Object Types and Scalar Types

- Object types represent a kind of object you can fetch from your service, and what fields it has.
- Scalar types represent primitive leaf values, including `String`, `Int`, `Float`, `Boolean`, and `ID`. User can define custom scalar types.

Defining a Schema

- GraphQL schema is the core of any GraphQL server implementation.
- It describes the functionality available to the client applications that connect to it.

```
type Query {
  users: [User]!
  user(id: ID!): User
}
```

```
type Mutation {
  createUser(username: String!, password: String!): User!
}

type Subscription {
  newUser: User!
}
```

Resolvers

- Resolvers are functions that are responsible for returning values for fields that exist on the GraphQL types defined in the schema.

```
query {
  user {
    username
    email
  }
}
```

```
const resolvers = {
  Query: {
    user: (parent, { id }, context, info) => {
      const user = users.find(user => user.id === id)
      return {
        username: user.username
        email: user.email
      }
    }
  }
};
```

GraphQL Server

- GraphQL server can be implemented in any programming language.
- GraphQL.js
- Apollo Server

GraphQL Client

- Apollo Client
- Relay
- graphql-hooks

Sample Code