GraphQL

The introduction of a new way of dealing with data

Muhammad Sholahuddin

@saladinid

Front End Engineer at Pinjam.co.id

Muhammad
Sholahuddin

Front End Engineer at Pinjam.co.id

@saladinid

Agenda

> Opening
> What’s GraphQL?
> How does it work?
> Demo
> Q & A

REST

A.P.I

HTTP

Request

Web Service

Response

Facebook

JSON

let me tell

you **a story**

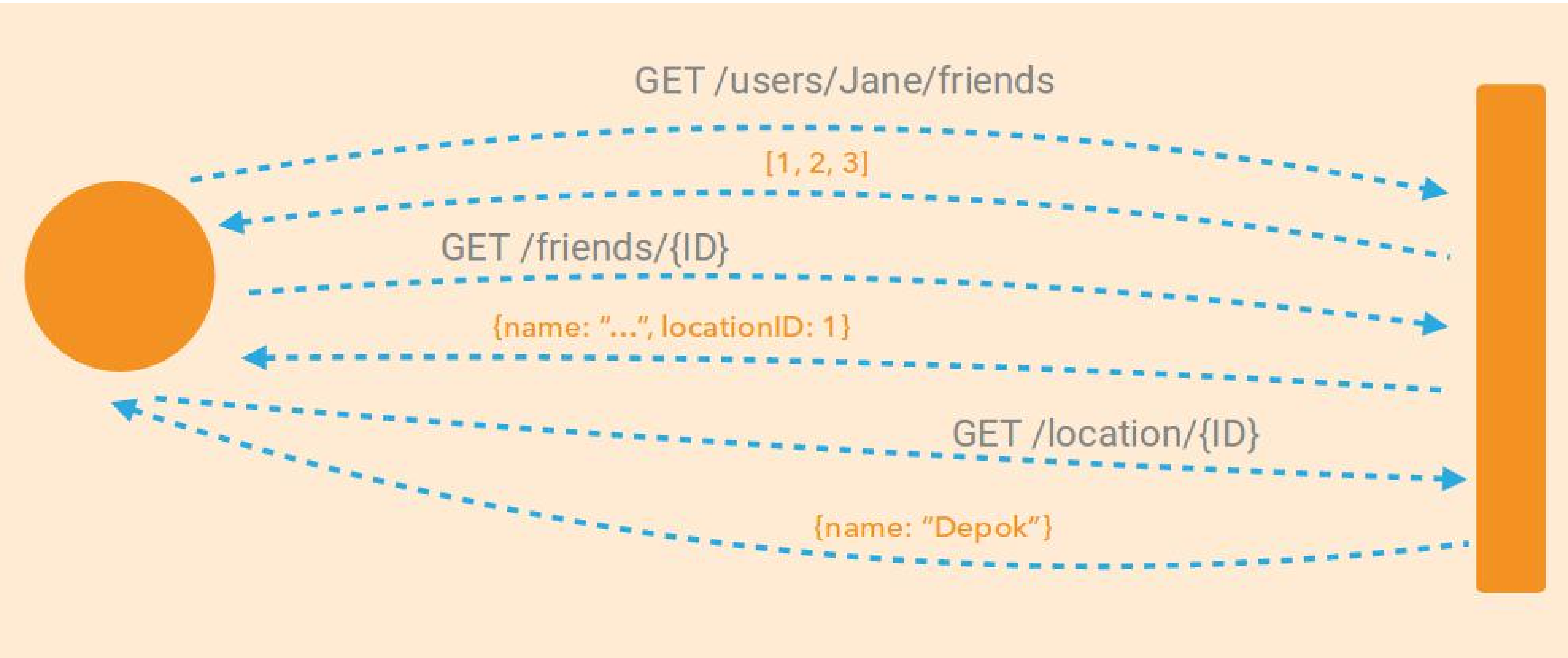
API

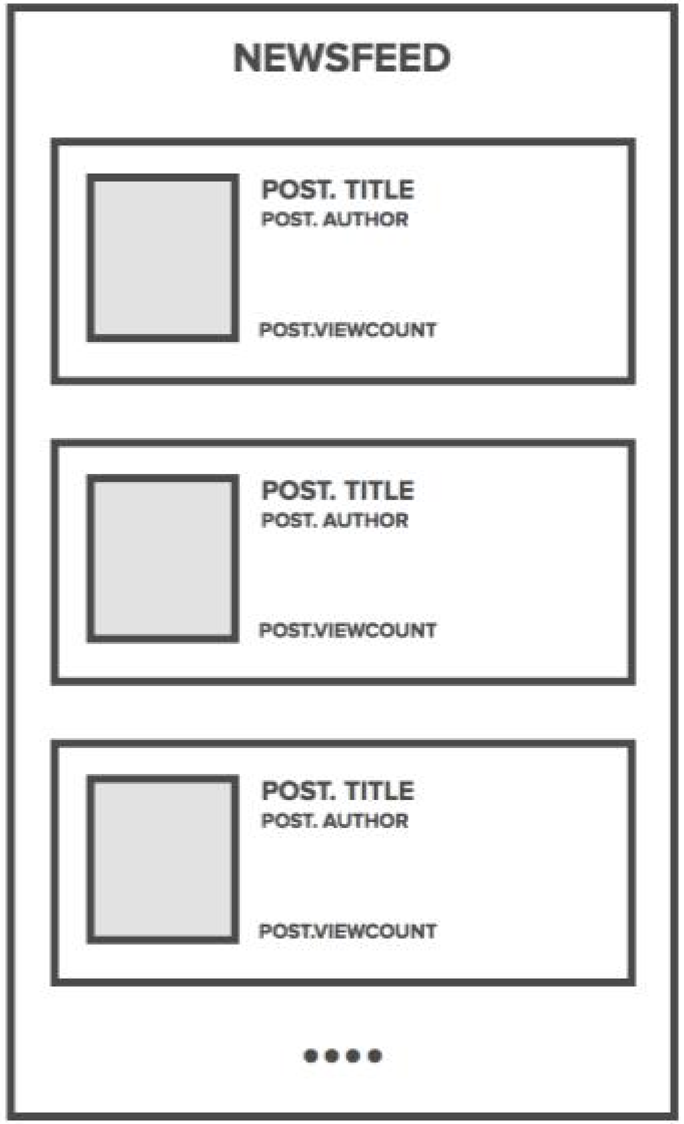
NEED SOME REST

List of Jane's friends
+ Their names
+ Their locations

```
GET /users/Jane/friends                 [1, 2, 3]
GET /friends/{ID}        {name: "...", locationID: 1}
GET /location/{ID}                  {name: "Depok"}
```

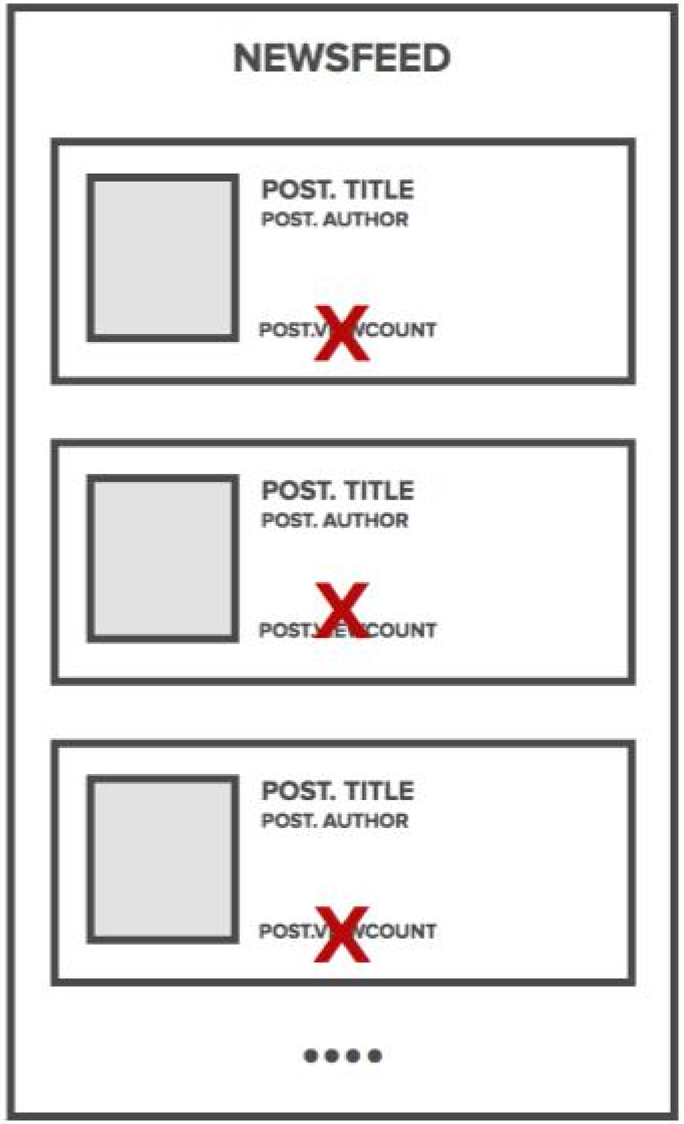
Multiple Round Trip (Multiple Calls)





v1

Data Version



v2

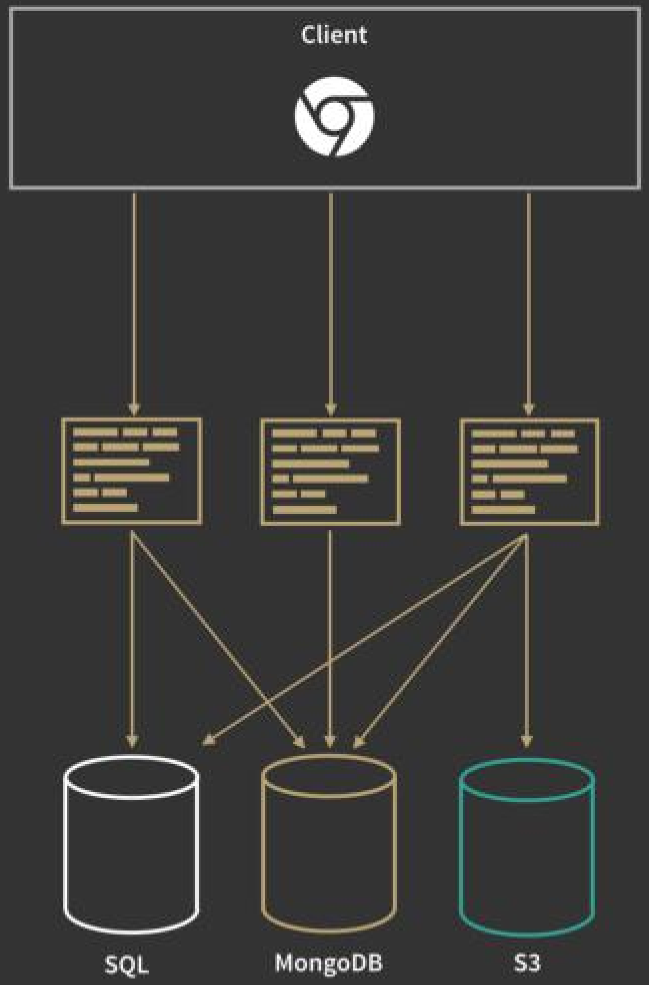
GraphQL

What is GraphQL?

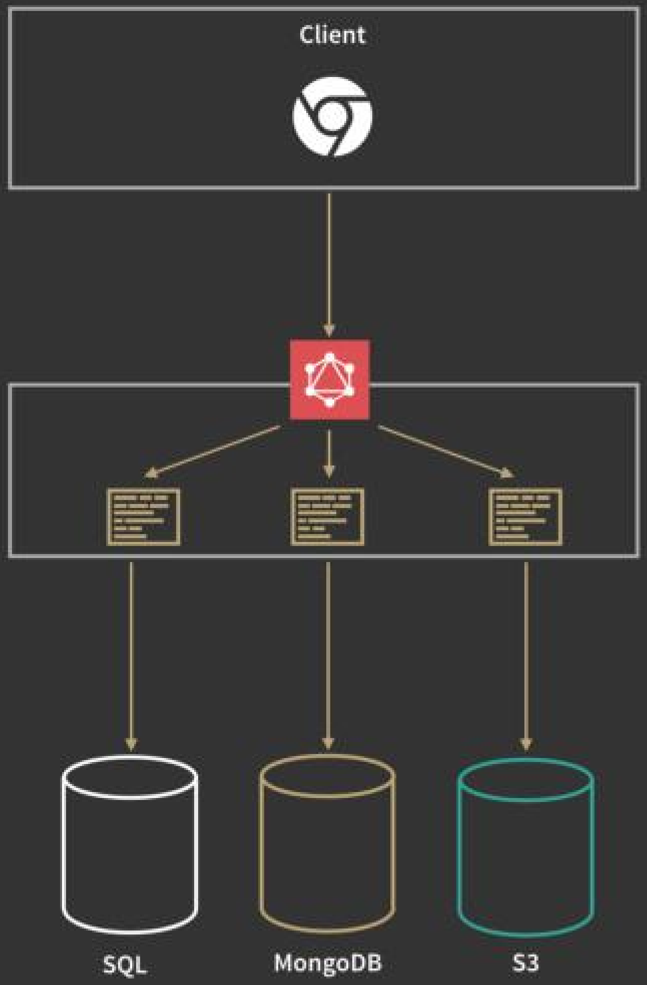
GraphQL is a data query language and runtime designed and used at Facebook to request and deliver data to mobile and web apps since 2012 (Open Source on 2015)

Source: http://graphql.org

REST



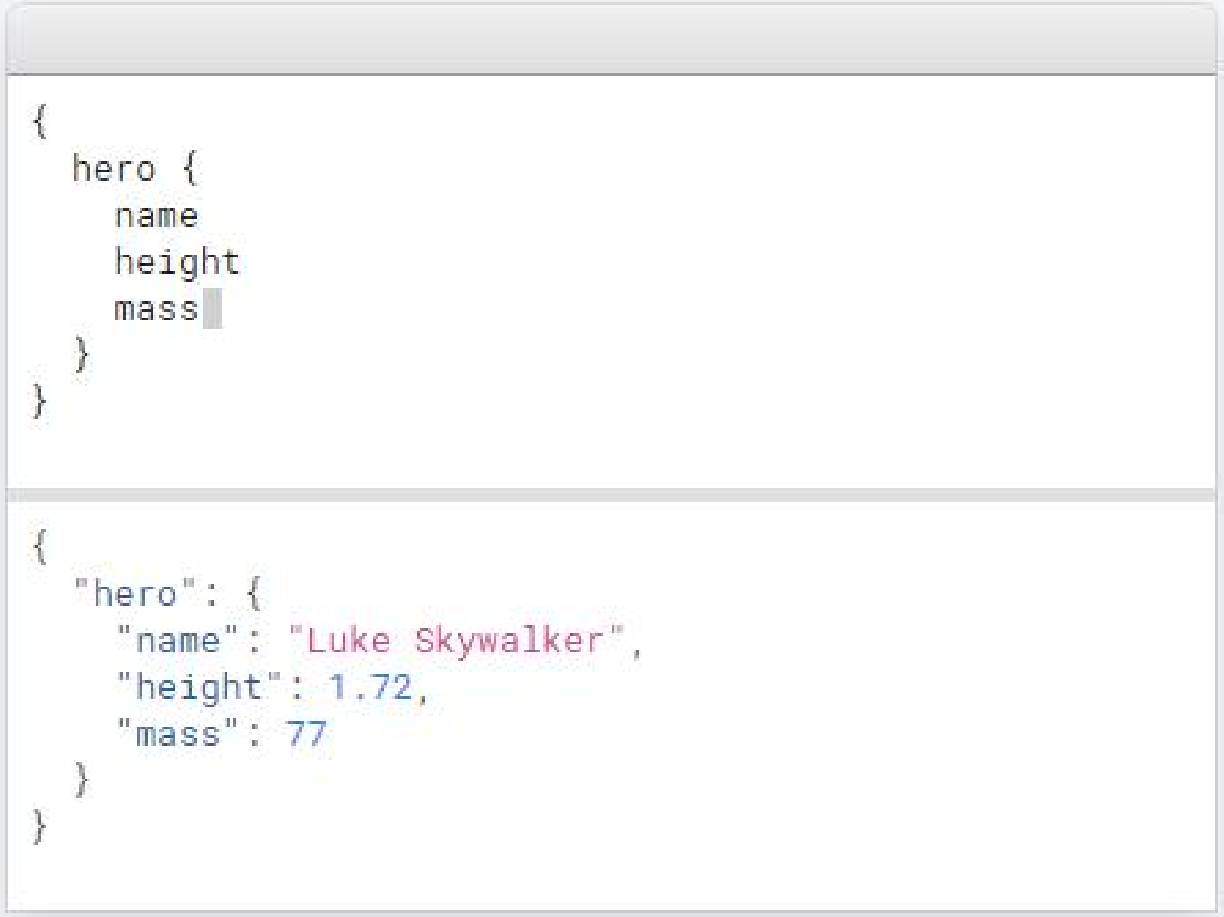
GraphQL



Main Topics

> Queries and Mutations
> Schemas and Types
> Validation
> Execution
> Introspection

Queries & Mutations

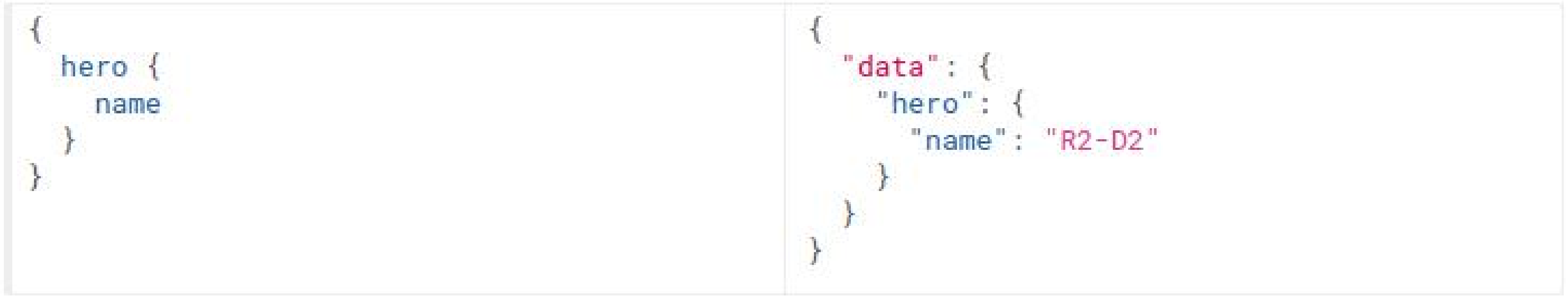


```
{
  hero {
    name
    height
    mass
  }
}
```

```
{
  "hero": {
    "name": "Luke Skywalker",
    "height": 1.72,
    "mass": 77
  }
}
```

Queries

"Ask for what you need, get exactly that"

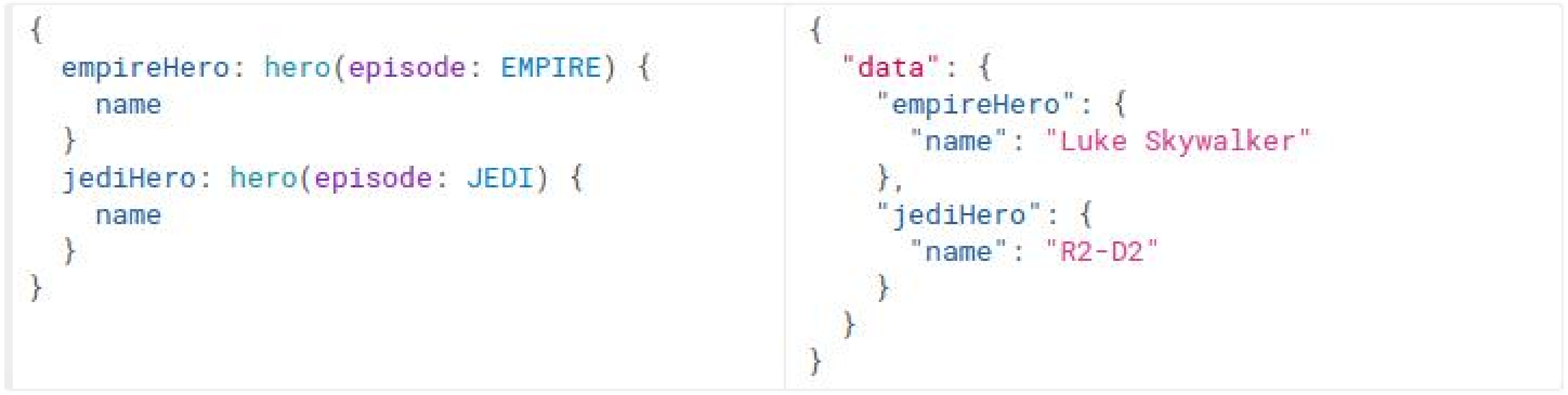


```
{
  hero {
    name
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2"
    }
  }
}
```

Queries

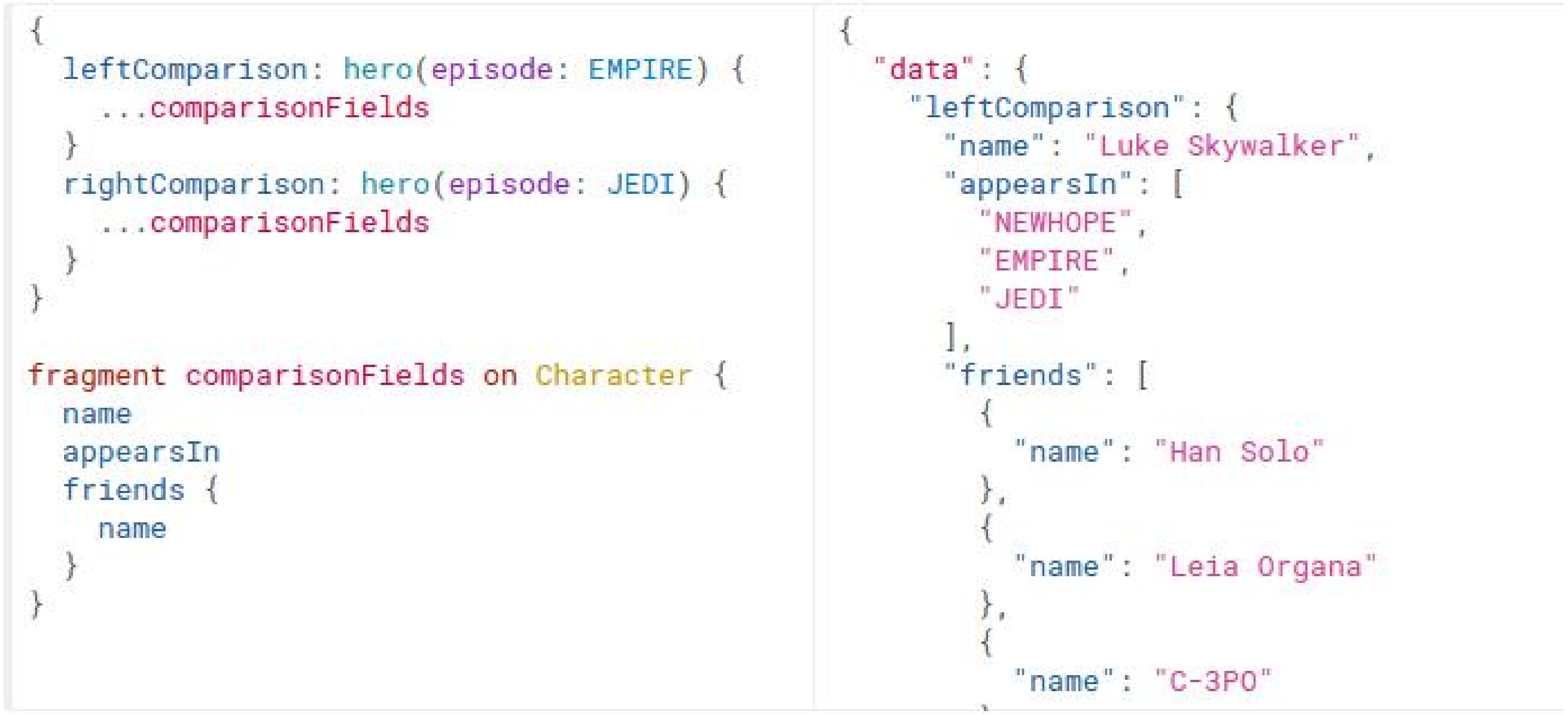
Aliases



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

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

Fragments

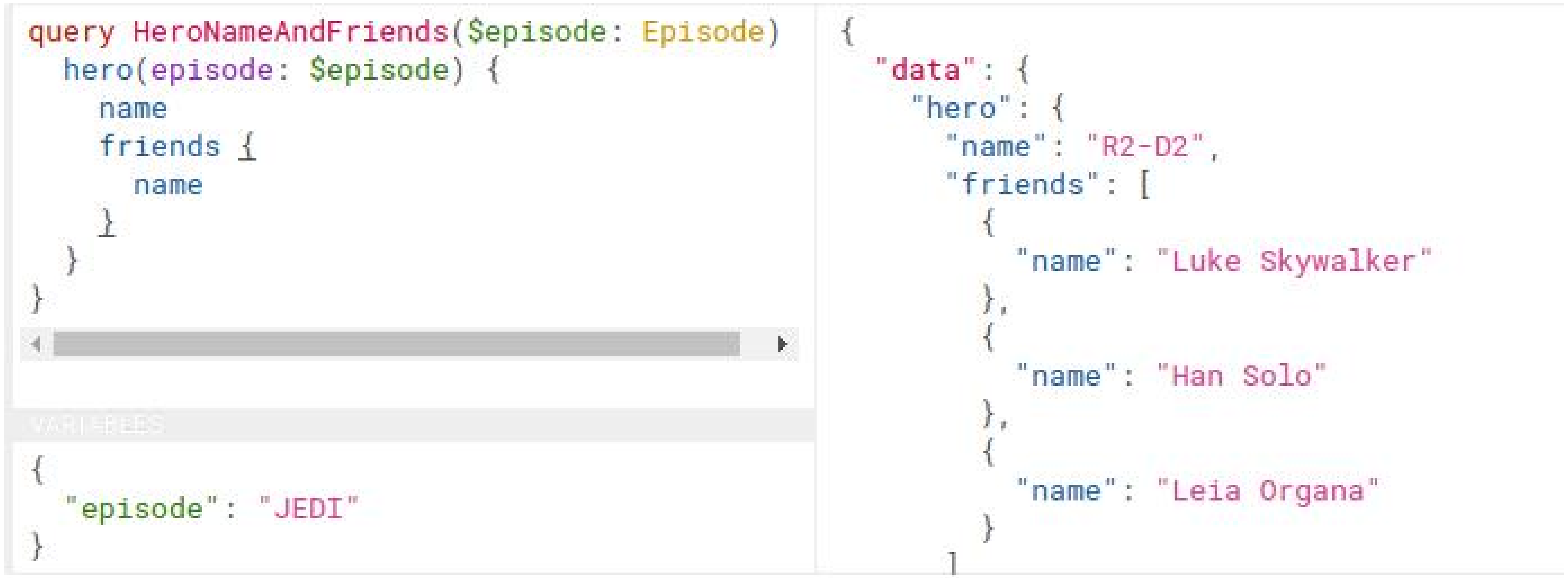


```
{
  leftComparison: hero(episode: EMPIRE) {
    ...comparisonFields
  }
  rightComparison: hero(episode: JEDI) {
    ...comparisonFields
  }
}

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

```
{
  "data": {
    "leftComparison": {
      "name": "Luke Skywalker",
      "appearsIn": [
        "NEWHOPE",
        "EMPIRE",
        "JEDI"
      ],
      "friends": [
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        },
        {
          "name": "C-3PO"
```

Variables



```
query HeroNameAndFriends($episode: Episode)
  hero(episode: $episode) {
    name
    friends {
      name
    }
  }
}
```

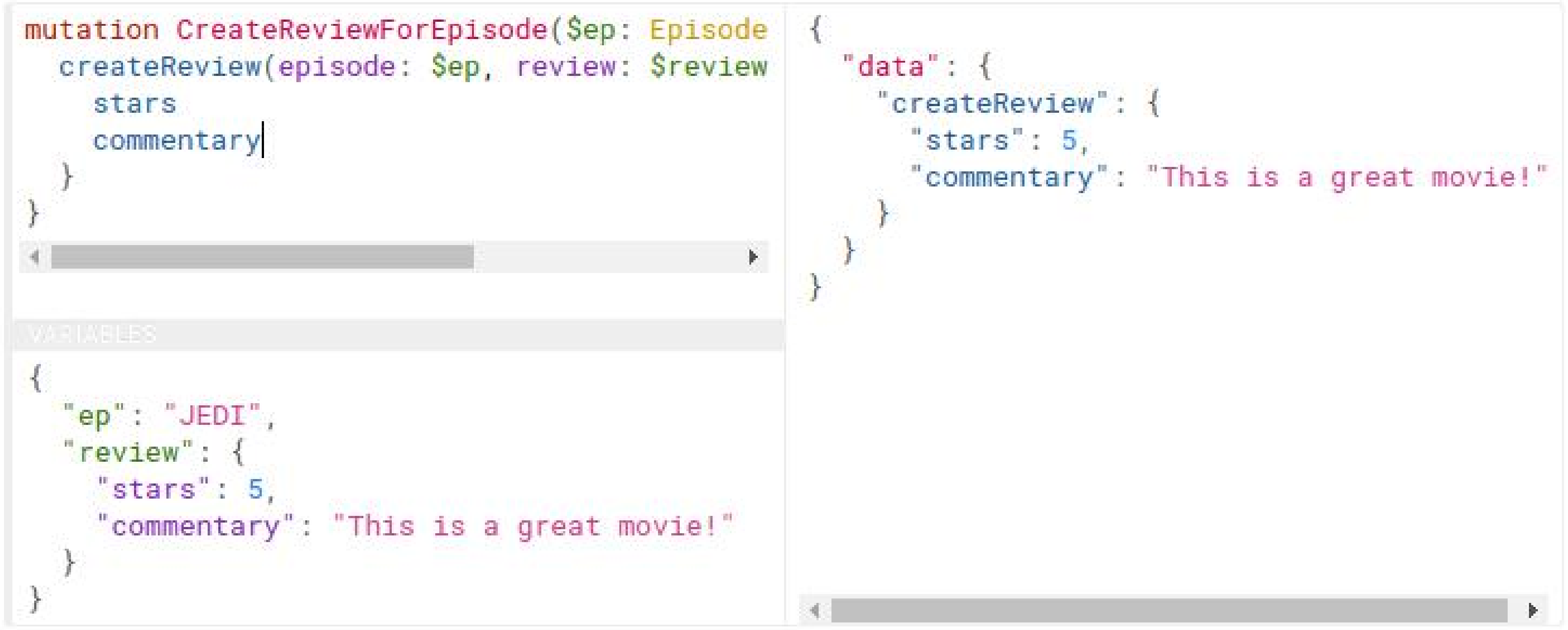
VARIABLES

```
{
  "episode": "JEDI"
}
```

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

Mutations

Just like in queries, if the mutation field returns an object type, you can ask for nested fields.



```
mutation CreateReviewForEpisode($ep: Episode
  createReview(episode: $ep, review: $review
    stars
    commentary
  }
}
```

VARIABLES

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

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

Mutations

Create

```
var MutationType = new GraphQLObjectType({
  name: 'GraphQL Mutations',
  description: 'These are the things we can change',
  fields: () => ({
    createArticle: {
      type: ArticleType,
      description: 'Create a new article',
      args: {
        article: { type: ArticleInputType }
      },
      resolve: (value, { article }) => {
        return
ArticleServices.createArticle(article);
      }
    }
  }),
});
```

Mutations

Update

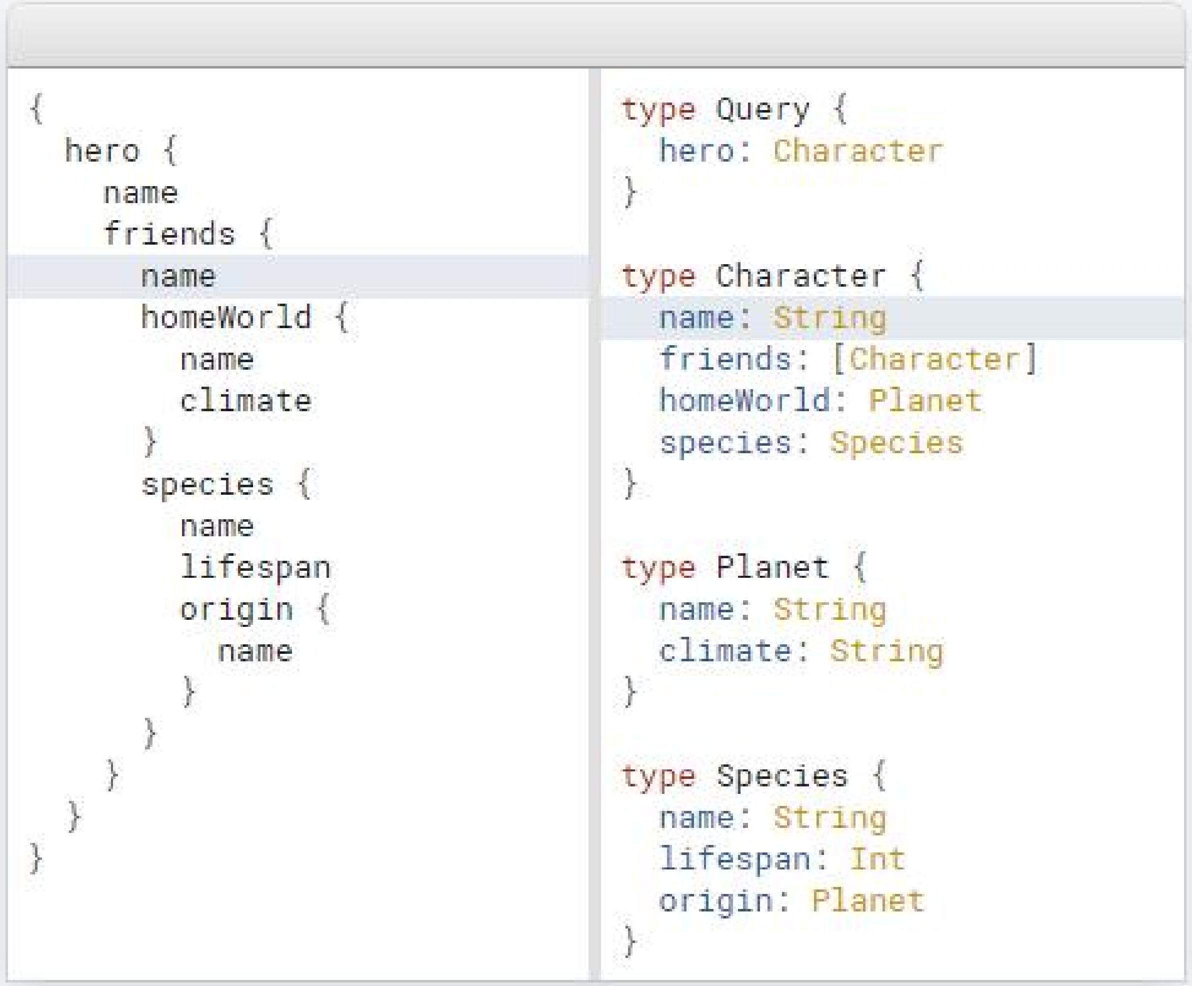
```
var MutationType = new GraphQLObjectType({
  name: 'GraphQL Mutations',
  description: 'These are the things we can change',
  fields: () => ({
    updateArticle: {
    type: ArticleType,
    description: 'Update an article article, and
optionally any related articles.',
    args: {
    article: { type: ArticleAttributesInputType }
  },
    resolve: (root, { article }) => {
    return
ArticleServices.createOrUpdateArticle(article);
  }
}
  }),
});
```

Mutations

Delete

```
var MutationType = new GraphQLObjectType({
  name: 'GraphQL Mutations',
  description: 'These are the things we can change',
  fields: () => ({
    deleteArticle: {
      type: ArticleType,
      description: 'Delete an article with id and
return the article that was deleted.',
      args: {
        id: { type: new GraphQLNonNull(GraphQLInt) }
      },
      resolve: (value, { id }) => {
        return ArticleServices.delete(id);
      }
    }
  }),
});
```

Schemas & Types



```
{
  hero {
    name
    friends {
      name
      homeWorld {
        name
        climate
      }
      species {
        name
        lifespan
        origin {
          name
        }
      }
    }
  }
}
```

```
type Query {
  hero: Character
}

type Character {
  name: String
  friends: [Character]
  homeWorld: Planet
  species: Species
}

type Planet {
  name: String
  climate: String
}

type Species {
  name: String
  lifespan: Int
  origin: Planet
}
```

Schemas & Types

GraphQL type system to **describes** what data can be queried. Using any **backend** framework or programming language.

```
const PersonType = new GraphQLObjectType({
  name: 'Person',
  description: 'Somebody that you used to know',
  fields: () => ({
    id: globalIdField('Person'),
    firstName: {
      type: GraphQLString,
      resolve: (person) => person.first_name,
    },
    lastName: {
      type: GraphQLString,
      resolve: (person) => person.last_name,
    },
    email: {
      type: GraphQLString
    },
    username: {
      type: GraphQLString
    },
    friends: {
      type: new GraphQLList(PersonType),
      resolve: person => personLoader.loadMany(person.friends),
    },
```

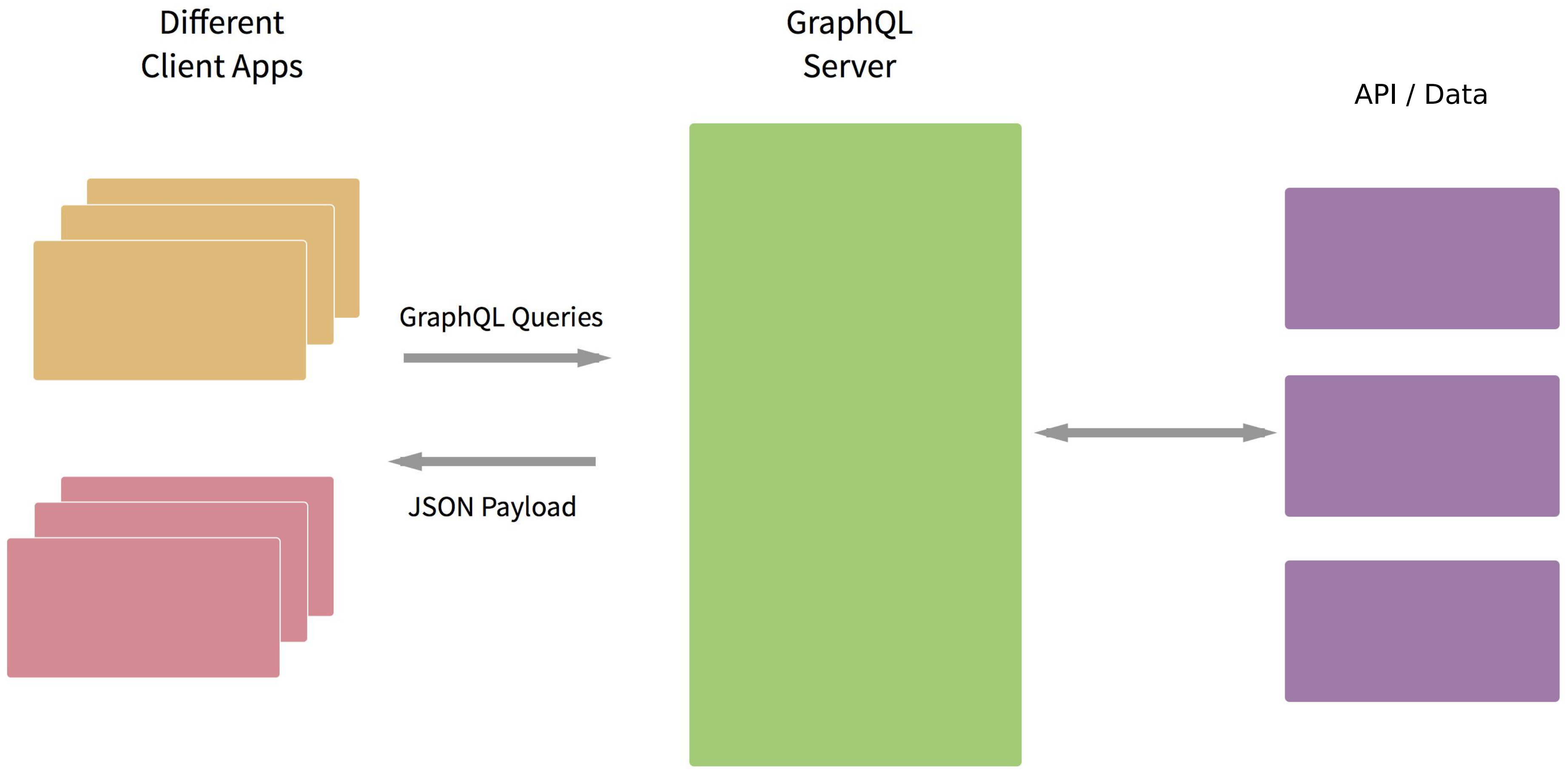
Best Practices

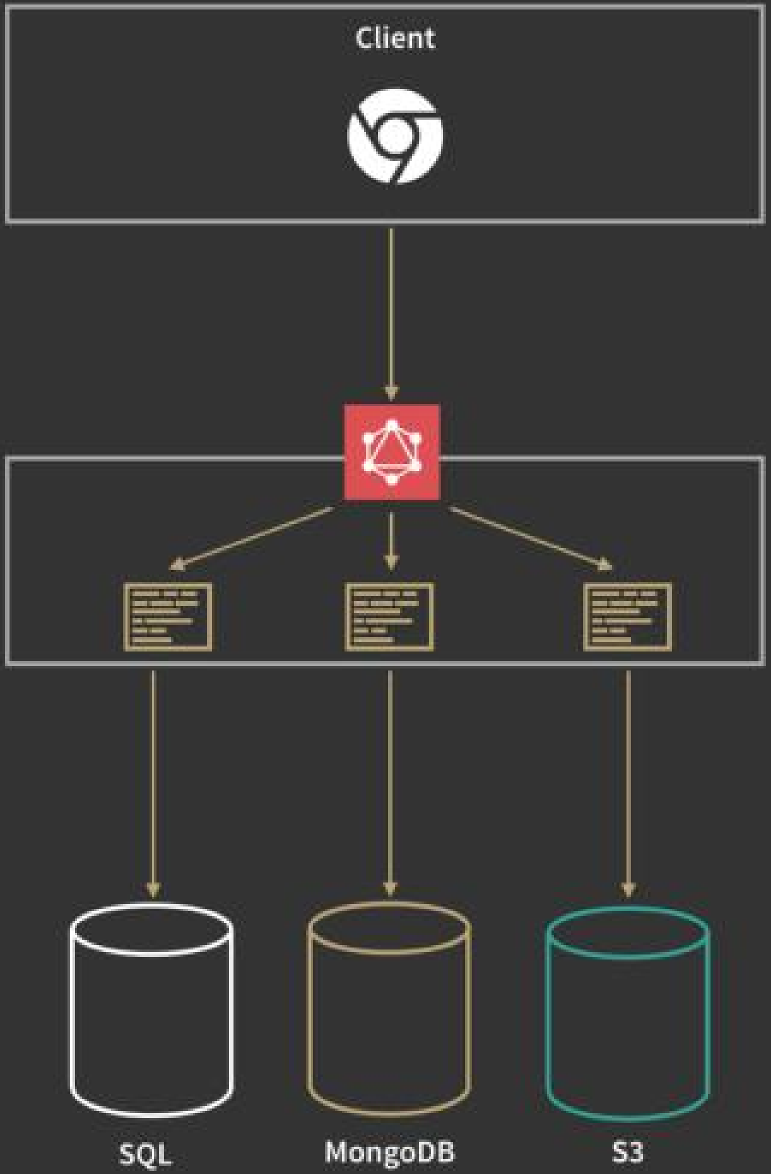
> Thinking in Graphs
> Serving over HTTP
> Authorization
> Pagination
> Caching

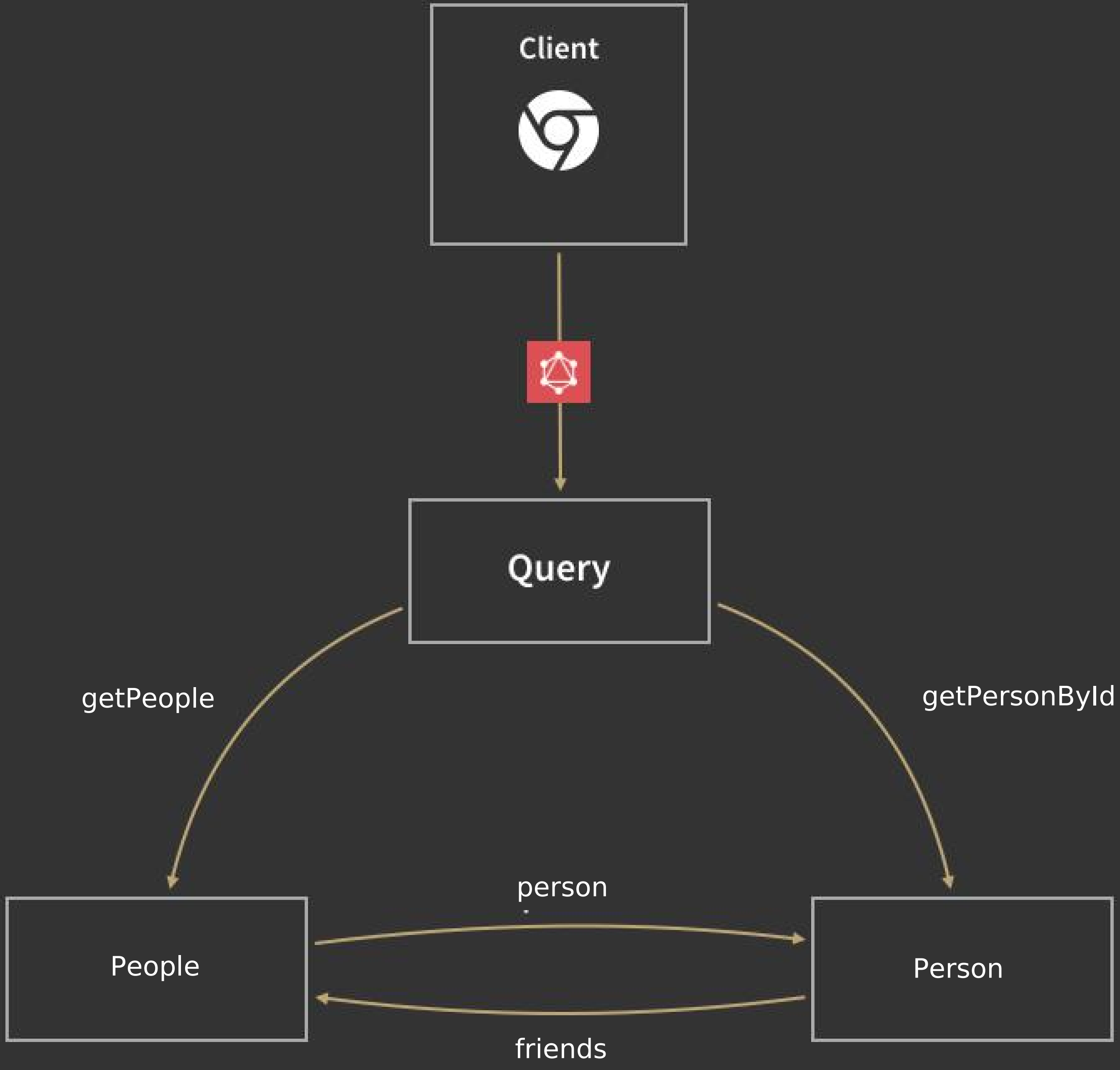
Best Practices

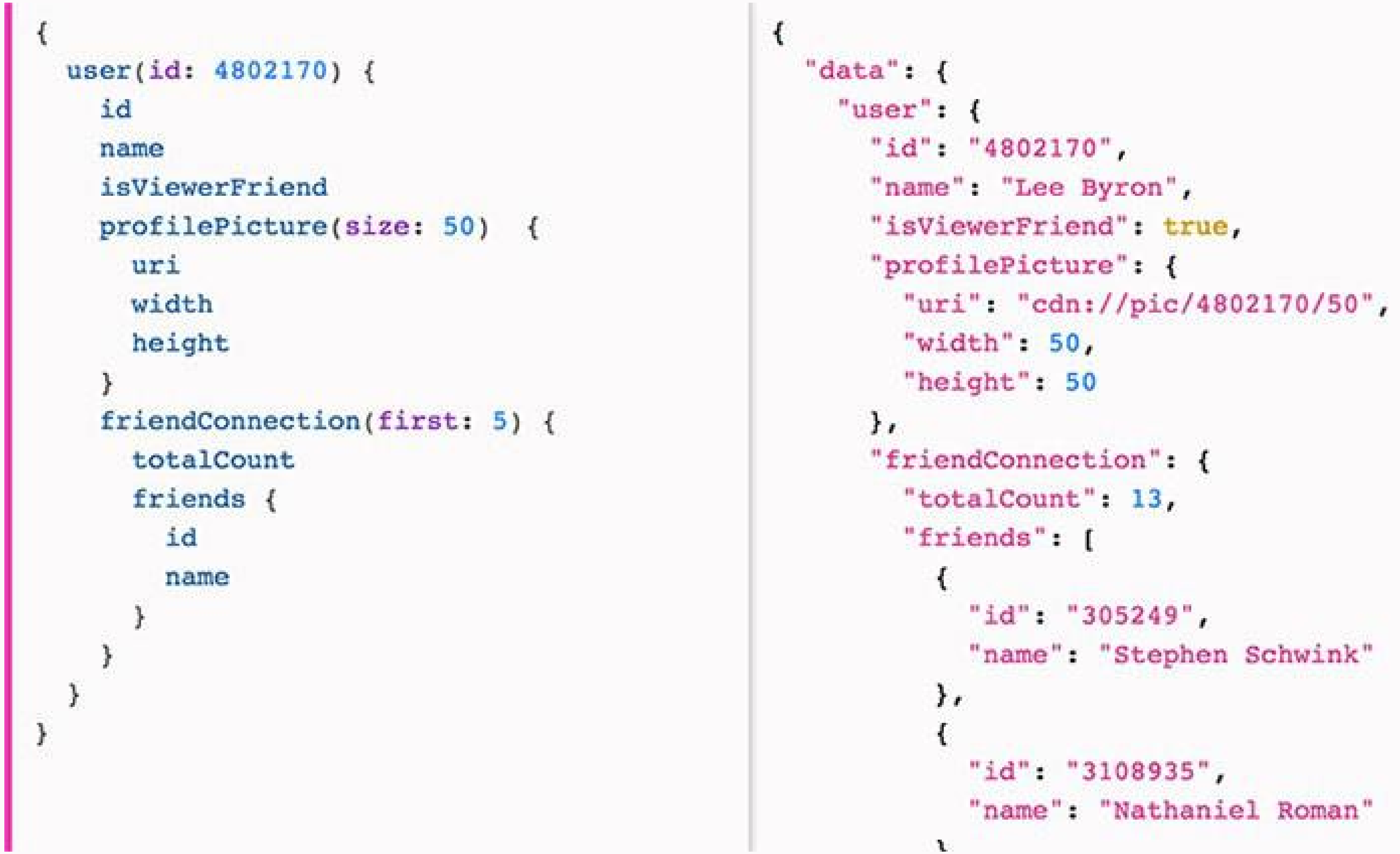
> Thinking in Graphs
> Serving over HTTP
> Authorization
> Pagination
> Caching

How does it works?


Different Client Apps
GraphQL Queries
JSON Payload
GraphQL Server
API / Data


Client
SQL
MongoDB
S3

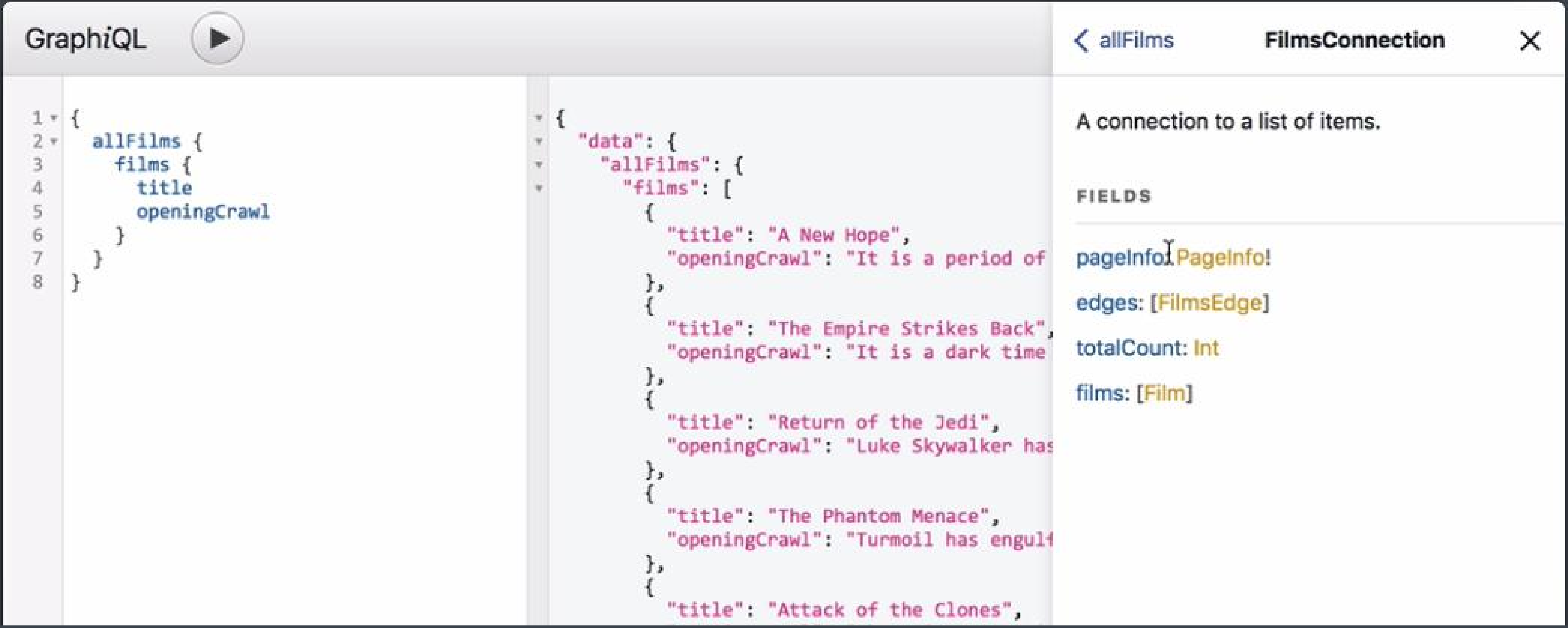

Client
Query
getPeople
getPersonById
person
People
Person
friends



```
{
  user(id: 4802170) {
    id
    name
    isViewerFriend
    profilePicture(size: 50)  {
      uri
      width
      height
    }
    friendConnection(first: 5) {
      totalCount
      friends {
        id
        name
      }
    }
  }
}
```

```
{
  "data": {
    "user": {
      "id": "4802170",
      "name": "Lee Byron",
      "isViewerFriend": true,
      "profilePicture": {
        "uri": "cdn://pic/4802170/50",
        "width": 50,
        "height": 50
      },
      "friendConnection": {
        "totalCount": 13,
        "friends": [
          {
            "id": "305249",
            "name": "Stephen Schwink"
          },
          {
            "id": "3108935",
            "name": "Nathaniel Roman"
```

GraphiQL - Cool Tool



```
{
  allFilms {
    films {
      title
      openingCrawl
    }
  }
}
```

```
{
  "data": {
    "allFilms": {
      "films": [
        {
          "title": "A New Hope",
          "openingCrawl": "It is a period of
        },
        {
          "title": "The Empire Strikes Back",
          "openingCrawl": "It is a dark time
        },
        {
          "title": "Return of the Jedi",
          "openingCrawl": "Luke Skywalker has
        },
        {
          "title": "The Phantom Menace",
          "openingCrawl": "Turmoil has engulf
        },
        {
          "title": "Attack of the Clones",
```

A connection to a list of items.

FIELDS

pageInfo: PageInfo!

edges: [FilmsEdge]

totalCount: Int

films: [Film]

Who’s using GraphQL?

The growing GraphQL ecosystem

GraphQL-JS Apollo stack Sangria Reindex GraphQL-ruby

Graphene Relay GraphQL hub Graphcool and more!

DEMO

GitHub

https://github.com/saladinid/techtalk-graphql

Conclusion

> GraphQL is query language
> GraphQL is not RESTful API killer
> GraphQL available for many Tech.
> Many supports
> For dynamic & high scalability App

Questions

LEARN MORE

- GraphQL Introduction
- Exploring GraphQL video
- Zero GraphQL video
- Fullstack React & GraphQL
- Your First GraphQL Server
- Intro To GraphQL

Thank You