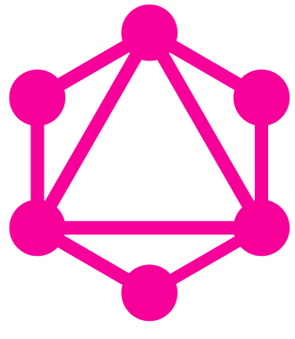


Почему GraphQL нужно использовать уже сейчас

Юрий Королев, AnjLab

Владимир 2016

REST API

Версионность REST API

- версия в URL
- версия в специальном HTTP заголовке запроса
- версия в HTTP заголовке Accept
- по версии клиента

Развитие API

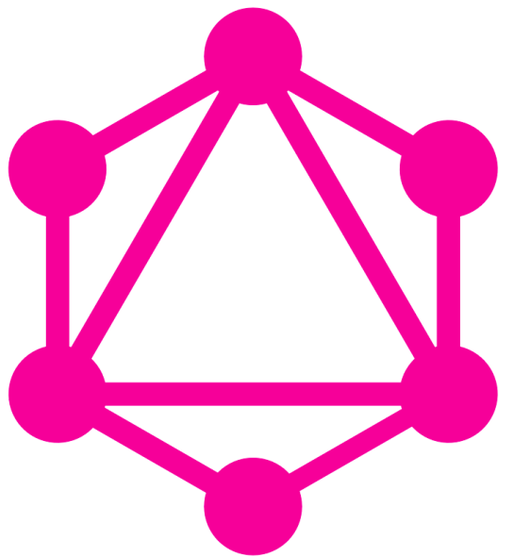
- Добавление новых полей в ответы сервера
- Добавление новых endpoint-ов
  - Часто композиция

Внедрение/отключение версий API

- тонкий клиент
- толстый закрытый клиент в закрытой системе
- толстый открытый клиент в закрытой системе
- открытая система

HTTP

- MQTT
- WebSockets
- и тд



GraphQL

GraphQL

“A query language for your API”

http://graphql.org

GraphQL

Queries Mutations Subscriptions

Objects Interfaces InputObjects Unions

Scalars Enums

4

Запрос

```
{
  viewer {
    login
  }
}
```

Ответ


```
{
  "data": {
    "viewer": {
      "login": "yury"
    }
  }
}
```

В запросах указывается структура ответа

Запрос

```
{
  viewer {
    login
    url
    avatarURL
  }
}
```

Ответ


```
{
  "data": {
    "viewer": {
      "login": "yury",
      "url": "https://github.com/yury",
      "avatarURL":
"https://avatars0.githubusercontent.com/u/5250?
v=3"
    }
  }
}
```

Поля могут принимать аргументы

Запрос

```
{
  viewer {
    login
    avatarURL(size: 10)
  }
}
```

Ответ

```
{
  "data": {
    "viewer": {
      "login": "yury",
      "avatarURL":
"https://avatars0.githubusercontent.com
/u/5250?v=3&s=10"
    }
  }
}
```

Именованные запросы и переменные

Запрос

```
query named($avatarSize: Int) {
  viewer {
    avatarURL(size: $avatarSize)
  }
}
```

QUERY VARIABLES

```
{
  "avatarSize": 10
}
```

Ответ


```
{
  "data": {
    "viewer": {
      "avatarURL":
"https://avatars0.githubusercontent.com/u
/5250?v=3&s=10"
    }
  }
}
```

Фрагменты

Запрос

```
{
  viewer {
    ... personalFields

    followers(first: 1) {
      edges {
        node {
          ... personalFields
        }
      }
    }
  }
}

fragment personalFields on User {
  name
  avatarURL
  company
}
```

Ответ

```
{
  "data": {
    "viewer": {
      "followers": {
        "edges": [
          {
            "node": {
              "name": "Andrew Nesbitt",
              "avatarURL":
"https://avatars1.githubusercontent.com/u
/1060?v=3",
              "company": "Dependency CI"
            }
          }
        ]
      },
      "name": "Yury Korolev",
      "avatarURL":
"https://avatars0.githubusercontent.com/u
/5250?v=3",
      "company": "anjlab.com"
    }
  }
}
```

ALIASES

Запрос

```
{
  user: viewer {
    id
    login
    company
  }
  repos: viewer {
    repositories {
      totalCount
    }
  }
}
```

Ответ

```
{
  "data": {
    "user": {
      "id": "MDQ6VXNlcjUyNTA=",
      "login": "yury",
      "company": "anjlab.com"
    },
    "repos": {
      "repositories": {
        "totalCount": 100
      }
    }
  }
}
```

Mutations

```graphql
mutation ($a: UpdateProjectInput!, $b: UpdateProjectInput!){
  op1: updateProject(input: $a) {
    project {
      name
    }
  }
  op2: updateProject(input: $b) {
    project {
      name
    }
  }
}
```

QUERY VARIABLES

```json
{
  "a": {
    "projectId": "MDc6UHJvamVjdDM1NQ==",
    "name": "planning 1"
  },
  "b": {
    "projectId": "MDc6UHJvamVjdDM1NQ==",
    "name": "planning 2"
  }
}
```

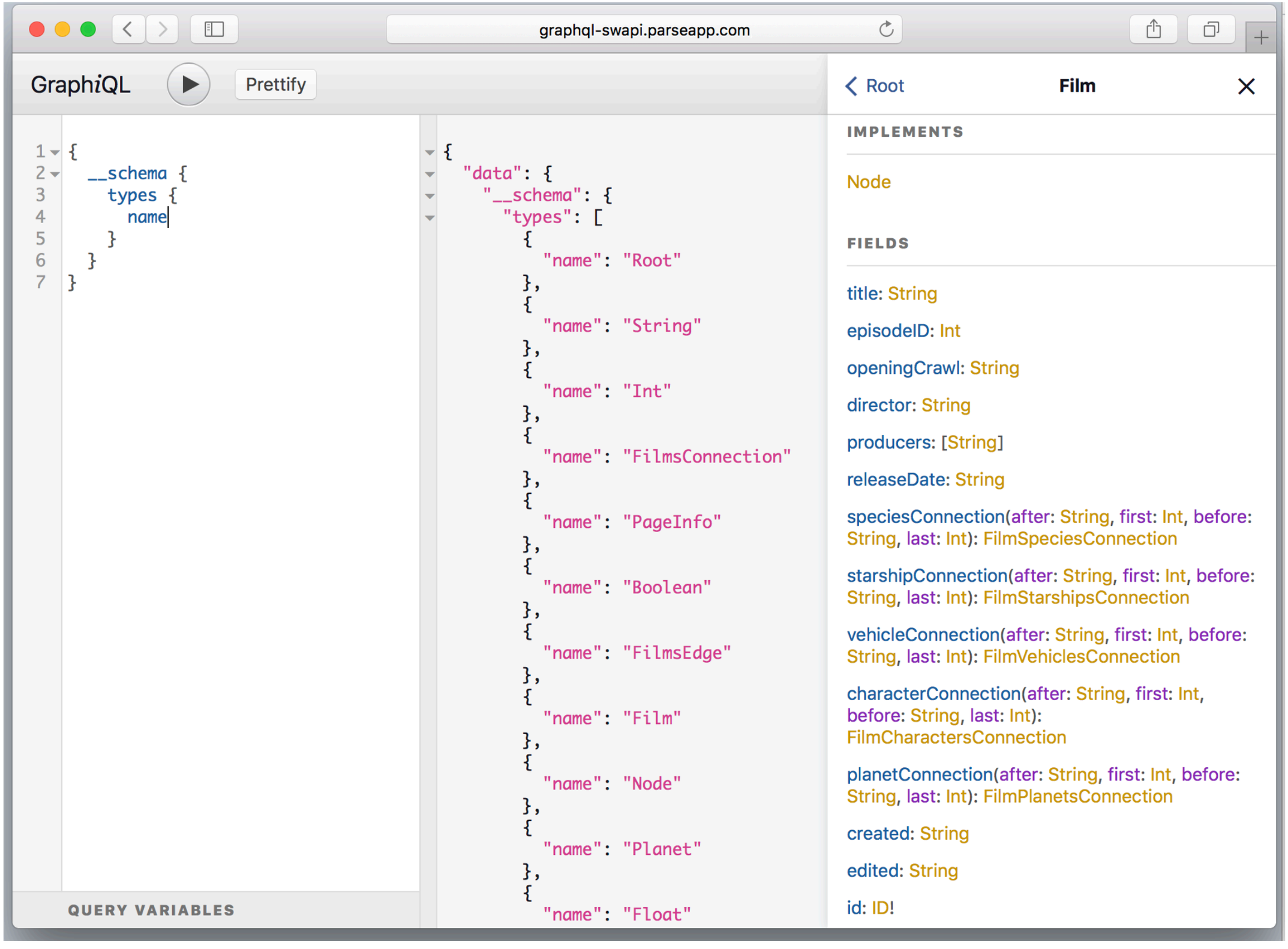
```json
{
  "data": {
    "op1": {
      "project": {
        "name": "planning 1"
      }
    },
    "op2": {
      "project": {
        "name": "planning 2"
      }
    }
  }
}
```

Multiple Fields

Query vs Mutation

Параллельно vs Последовательно

Tooling


graphql-swapi.parseapp.com
GraphiQL
Prettify
{
__schema {
types {
name
}
}
}
{
"data": {
"__schema": {
"types": [
{
"name": "Root"
},
{
"name": "String"
},
{
"name": "Int"
},
{
"name": "FilmsConnection"
},
{
"name": "PageInfo"
},
{
"name": "Boolean"
},
{
"name": "FilmsEdge"
},
{
"name": "Film"
},
{
"name": "Node"
},
{
"name": "Planet"
},
{
"name": "Float"
QUERY VARIABLES
Root
Film
IMPLEMENTS
Node
FIELDS
title: String
episodeID: Int
openingCrawl: String
director: String
producers: [String]
releaseDate: String
speciesConnection(after: String, first: Int, before: String, last: Int): FilmSpeciesConnection
starshipConnection(after: String, first: Int, before: String, last: Int): FilmStarshipsConnection
vehicleConnection(after: String, first: Int, before: String, last: Int): FilmVehiclesConnection
characterConnection(after: String, first: Int, before: String, last: Int): FilmCharactersConnection
planetConnection(after: String, first: Int, before: String, last: Int): FilmPlanetsConnection
created: String
edited: String
id: ID!

Кто пользуется

Реализации

- Серверные библиотеки
  - GraphQL.js (node)
  - express-graphql (node)
  - Apollo Server (node)
  - graphql-ruby (ruby)
  - Graphene (python)
  - Sangria (Scala)
  - graphql-java (Java)
  - graphql-go
  - graphql-php
  - graphql-dotnet
  - …….

Примеры использования

```
var query = '{ hello }';

graphql(schema, query).then(result => {
  // Prints
  // {
  //   data: { hello: "world" }
  // }
  console.log(result);

});
```

Пример сервера

```ruby
# Declare a type...
PostType = GraphQL::ObjectType.define do
  name "Post"
  description "A blog post"

  field :id, !types.ID
  field :title, !types.String
  field :body, !types.String
  field :comments, types[!CommentType]
end
```

Пример сервера

```
QueryType = GraphQL::ObjectType.define do
  name "Query"
  description "The query root of this schema"

  field :post do
    type PostType
    argument :id, !types.ID
    resolve -> (obj, args, ctx) { Post.find(args["id"]) }
  end
end
```

Пример сервера

```
Schema = GraphQL::Schema.define do
  query QueryType
  max_depth 8
end

result_hash = Schema.execute(query_string)
# {
#   "data" => {
#     "post" => {
#       "id" => 1,
#       "title" => "GraphQL is nice"
#     }
#   }
# }
```

Relay, Apollo

- COLOCATION
- BATCHING

Спасибо за внимание.