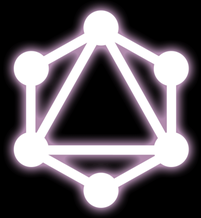


GraphQL
Clear and Performant

Argemiro Neto

- Data Engineer @Semios
- Bachelor and Master’s degree in CS
- 1 Year working with production level GraphQL API's

@argemiront

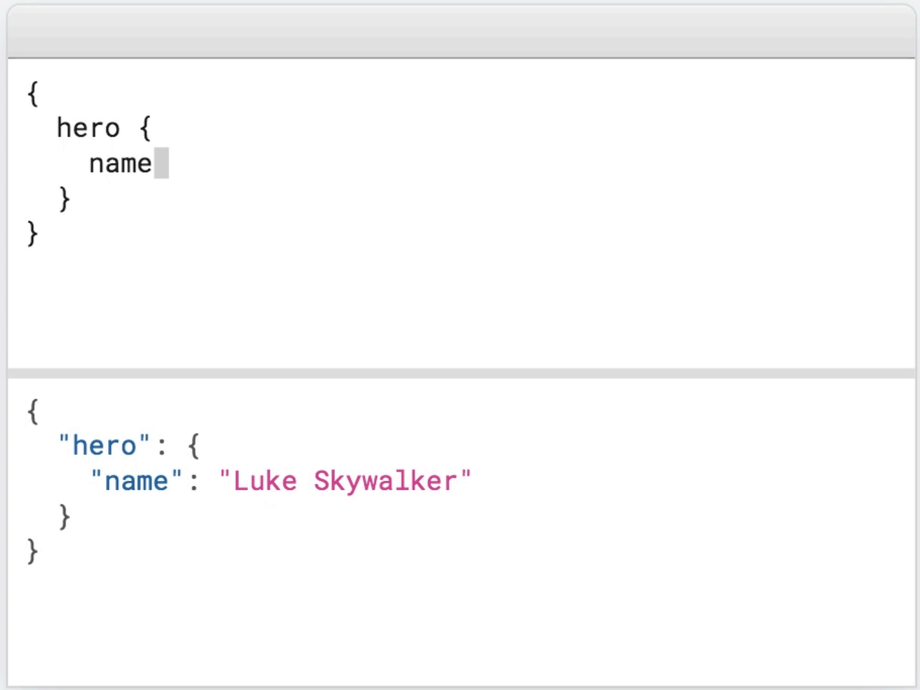
@argemironeto

@argemiront

- GraphQL Basics
- Apply middleware
- Use dataloaders to cache/batch queries
- Create schema directives

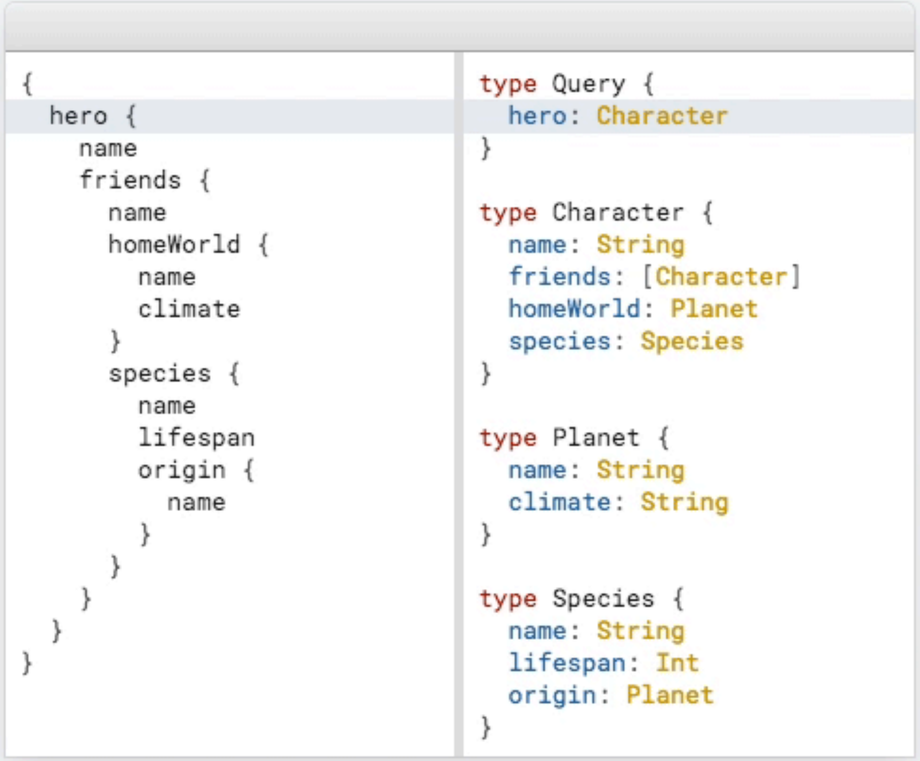
Presentation and code are available on GitHub

“GraphQL is a query language for APIs”



Query what you need

Define the data you want to fetch



```
{
  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
}
```

It’s based on types, not endpoints

Define the data you want to fetch

Demo time!

The engine decides what should return to the client

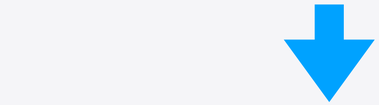
```
me: (_, args, ctx, info) => {
  return {
    id: '1',
    name: 'Argemiro Neto',
    friends: [],
    activities: {},
  };
},
```

```
query {
  me {
    name
    activities {
      hiking
    }
  }
}
```

```
{
  "data": {
    "me": {
      "name": "Argemiro Neto",
      "activities": {
        "hiking": [
          "Garibaldi Lake",
          "Diez Vistas",
          "Stanley Park"
        ]
      }
    }
  }
}
```

How did the empty object was filled??

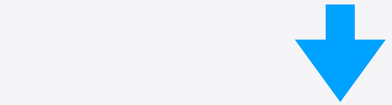
```
me: (_, args, ctx, info) => {
  return {
    id: '1',
    name: 'Argemiro Neto',
    friends: [],
    activities: {}
  };
},
```



```
query {
  me {
    name
    activities {
      hiking
    }
  }
}
```

```
{
  "data": {
    "me": {
      "name": "Argemiro Neto",
      "activities": {
        "hiking": [
          "Garibaldi Lake",
          "Diez Vistas",
          "Stanley Park"
        ]
      }
    }
  }
```

```
me: (_, args, ctx, info) => {
  return {
    id: '1',
    name: 'Argemiro Neto',
    friends: [],
    activities: {}
  };
},
```



Resolvers are traversed Pre-Order!

```
Interests: {
  hiking: (parent, args, ctx, info) => {
    return [
      'Garibaldi Lake',
      'Diez Vistas',
      'Stanley Park'
    ];
  },
```

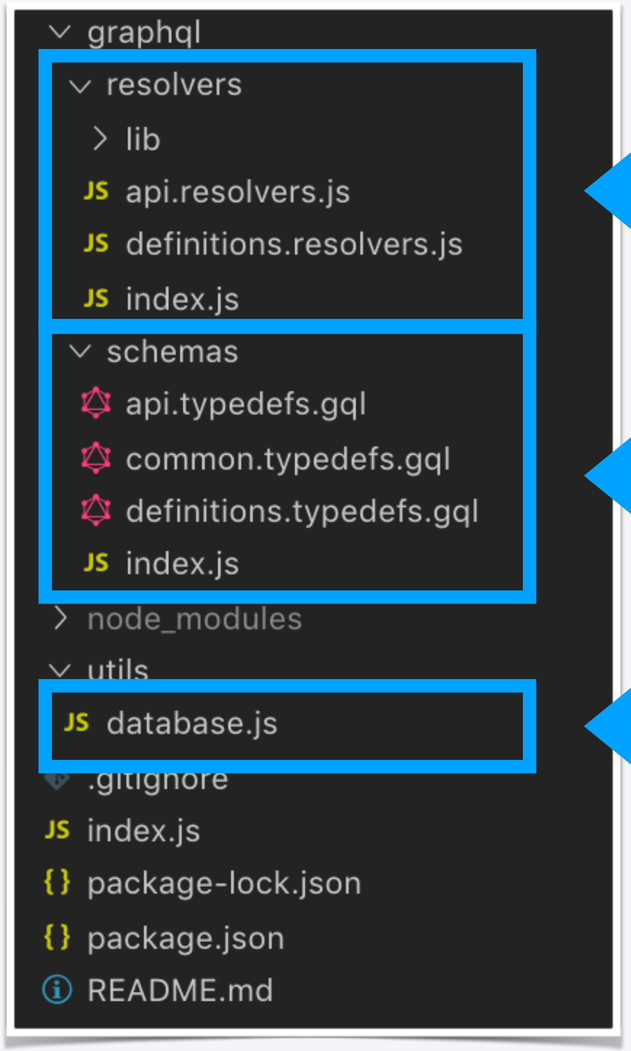
```
query {
  me {
    name
    activities {
      hiking
    }
  }
}
```

```
{
  "data": {
    "me": {
      "name": "Argemiro Neto",
      "activities": {
        "hiking": [
          "Garibaldi Lake",
          "Diez Vistas",
          "Stanley Park"
        ]
      }
    }
  }
}
```

index.js

```
const { ApolloServer, gql } = require('apollo-server');

const typeDefs = gql`
  type Query {
    """
    Fetches a person's data and interests
    """
    me: Person
  }

  """
  A Person
  """
  type Person {
    id: ID!
    name: String!
    activities: Interests
    friends: [Person]!
  }

  type Interests {
    hiking: [String]!
    concert: [String]!
    watchList: [String]!
  }
`;

const resolvers = {
  Query: {
    me: (_, args, ctx, info) => {
      return {
        id: '1',
        name: 'Argemiro Neto',
        friends: [],
        activities: {},
      };
    },
  },
  Interests: {
    hiking: (parent, args, ctx, info) => {
```

Define a better folder's structure…



Resolvers added by the **resolvers** pattern

Types added by the **typedefs** pattern

Simulate a database

Demo time!

```
const Query = {
  me: (_, args, ctx, info) => {
    return lib.person.getByID('p1');
  },
};
```

Apply middleware to request and resolvers…

- Use the Request Context (authentication)
- Apply a middleware to all resolvers
- Apply a middleware to a specific resolver
- Create an extension to control the GraphQL request/response flow
- Make the request performant with dataloaders

Demo time!

Additional computation with directives…

- Add the directive to the schema
- Extend the ***SchemaDirectiveVisitor*** class
- Override the desired visitor method

Demo time!

Thank you!

Argemiro Neto

Presentation code*

@argemironeto

@argemiront

* **https://github.com/argemiront/vanjs-graphql.git**

Argemiro Neto