GraphQL

An alternative to RESTful APIs

Introduction

➔ An API standard that provides a more efficient, powerful and flexible alternative to REST

➔ Enables *declarative data fetching* where a client can specify exactly what data it needs from an API

What Is GraphQL?

- ➔ A Query Language For APIs
- ➔ A long specification document that describes how a graphql server should behave
- ➔ Developed and open-sourced by *Facebook* in 2015
- ➔ Server libraries for Node.js environment
  - ○ GraphQL.js
  - ○ Apollo Server
  - ○ GraphQL-HTTP

Core Concepts

- ➔ Schema
- ➔ Schema Definition Language (SDL)
- ➔ Types & Fields
- ➔ Operations – Query, Mutation, Subscription
- ➔ Resolvers
- ➔ Introspection

Core Concepts

Schema

Specifies capabilities of the API, contract between the server and client.

```
type Person {
  name: String!
  age: Int!
  posts: [Post!]!
}

type Post {
  title: String!
  author: Person!
}
```

Core Concepts

Schema Definition Language (SDL)

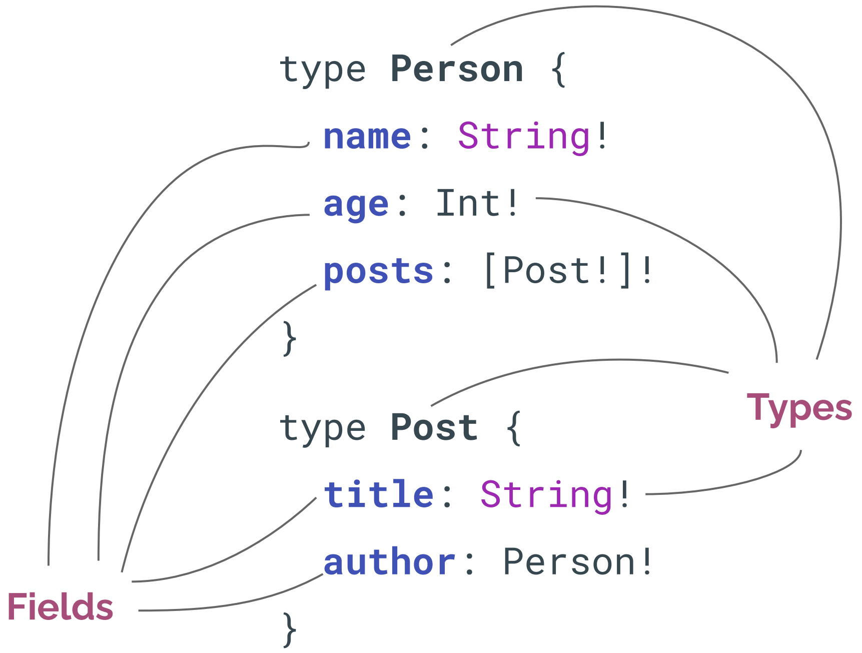
Syntax for writing schemas full-fledged with a type system.

```
type Person {
  name: String!
  age: Int!
}
```

Core Concepts

Types & Fields

Data that is requested for from a GraphQL server – scalar, list, object, custom scalar, non-null and interface



```
type Person {
  name: String!
  age: Int!
  posts: [Post!]!
}

type Post {
  title: String!
  author: Person!
}
```

Core Concepts

Query

An operation of structured request for data from a GraphQL API – C**R**UD

```
query {
    persons {
        name
        age
        posts {
            title
        }
    }
}
```

Core Concepts

Mutation

An operation for performing data changes on server – **C**R**UD**

```
mutation {
  createPerson(name: "Bob",
age: 36) {
    name
    age
  }
}
```

Core Concepts

Subscription

An operation for subscribing to an event and receiving real-time updates – continuous read

```
subscription {
  newPerson {
    name
    age
  }
}
```

Core Concepts

Resolvers

A function on a GraphQL server that is responsible for fetching the data for a single field

```
const allPersons = [
    { name: "Bob",age: 32 },
    { name: "Alice", age: 56 }
]

Query: {
    persons: () => allPersons
}
```

Core Concepts

Introspection

The ability for a client to ask a server for information about its schema, Discoverability of a GraphQL server's type system

When To Use

- ➔ Underfetching and overfetching
- ➔ Variety of different frontend frameworks and platforms
- ➔ Fast development & expectation for rapid feature development
- ➔ Increased mobile usage creates need for efficient data loading
- ➔ Slow loading times because of request waterfalls and/or overfetching

Demo

Q & A