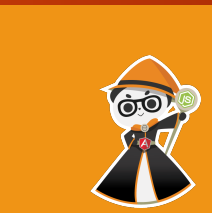
GraphQL & React

Tamas Piros

Hello 👋



Tamas Piros

Google Developer Expert

Director @ Full Stack Training

@tpiros

What is GraphQL?

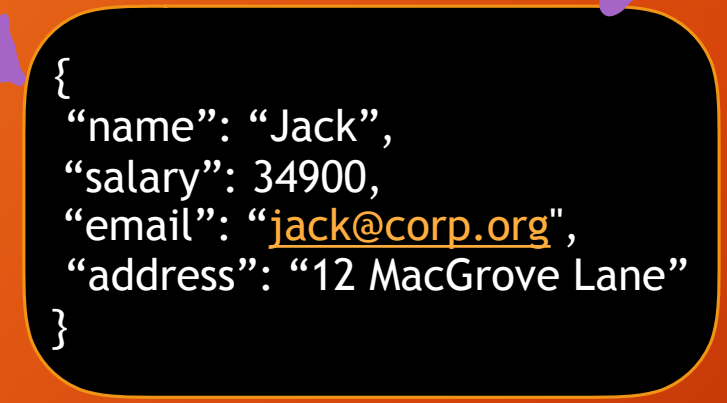
- Query language for an API
- REST vs GraphQL
  - Single request vs multiple requests
  - Get the data you need exactly
- Uses a type system
- Doesn’t necessarily rely on any database or storage

Let’s clarify a myth …

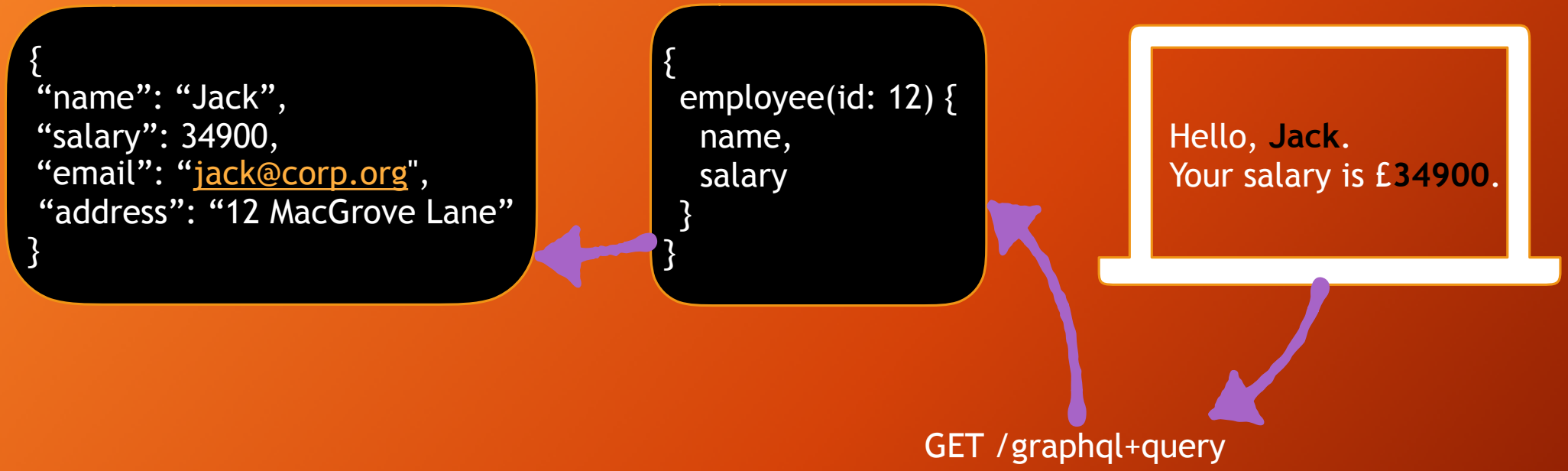
- GraphQL will replace REST - **NOT TRUE**
- REST is wrong / outdated - **NOT TRUE**

- Both world have pros and cons
  - REST - data overload / over query - but good server side caching
  - GraphQL - precise data queries - complicated server side caching

GET /api/employees/12

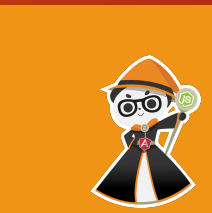


Traditional, API based approach



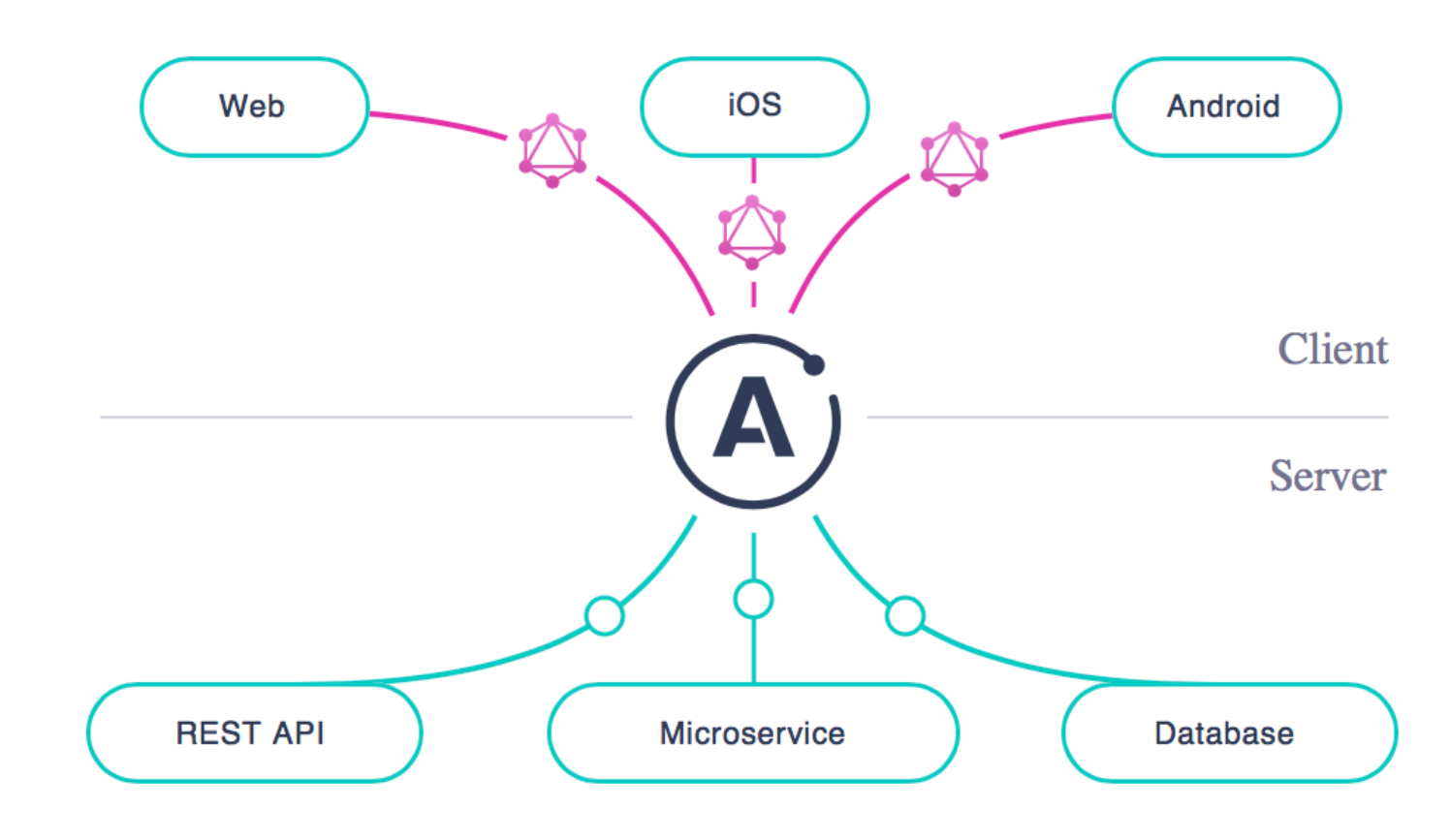
GraphQL approach

GraphQL approach



- Served over HTTP/S
- Two approaches
  - Build GraphQL on top of a database
  - Build GraphQL on top of an existing API
- Data returned has “data” and “errors” properties

“The magical connector” (Apollo)



Key terms

- Schema
  - Type Definitions
  - Mandatory “Query” type
  - Fields & nested fields
  - “How data looks like”
- Resolver
  - JavaScript objects (“resolver map”)
  - One resolver per query type
  - “What queries return”

Schema

- Also referred to as Type Definitions (type defs)
- Specifies the available data for reading/writing
- Has a mandatory Query type
  - Defines what to query for
- Works using fields and nested fields
- Uses SDL - Schema Definition Language
  - Schema doesn't care where the data is coming from

Schemas and Types

- Field values use scalars - String, Int, Boolean etc
- Type specifies the type for a given field
- Custom types can be added (like User)
- Exclamation mark: non-nullable field
- ID - internal identifier for advanced usage (e.g. caching)

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

```
type User {
  id: ID
  name: String!
  email: String
}
```

Schemas and Types

- Schema specifies the “structure”
- Type specifies the type for a given field

```
type Employee {
  id: Int,
  name: String,
  email: String
}
```

```
type Department {
  id: Int,
  name: String,
  employees: [Employee]
}
```

```
{
  department {
    name,
    employees {
      name,
      salary
    }
  }
}
```

Resolver

- Function that defines where the data is coming from
- Resolver Map - a JavaScript object of resolvers
- Each top level Query type needs a resolver

- Works with 4 parameters: object, arguments, context, info
  - Object - contains the results on the parent
  - Args - “get employee with id 12”
  - Context - an object shared by all resolvers - useful for authentication!
  - Info - rarely used, contains execution information

Query

- Returns data based on the Schema and the Resolver

```
{
 users {
   name,
   email
 }
}
```

```
{
  “data”: {
    “users”: [{
      “name”: “John”,
      “email”: “john@example.com"
    }, {
      “name”: “Sue”,
      “email”: “sue@example.com”
    }]
  }
}
```

Types

```
type Query {
    employees: [Employee!]
    employee(id: Int!): Employee
  }

type Employee {
  name: String!,
  salary: Int
}
```

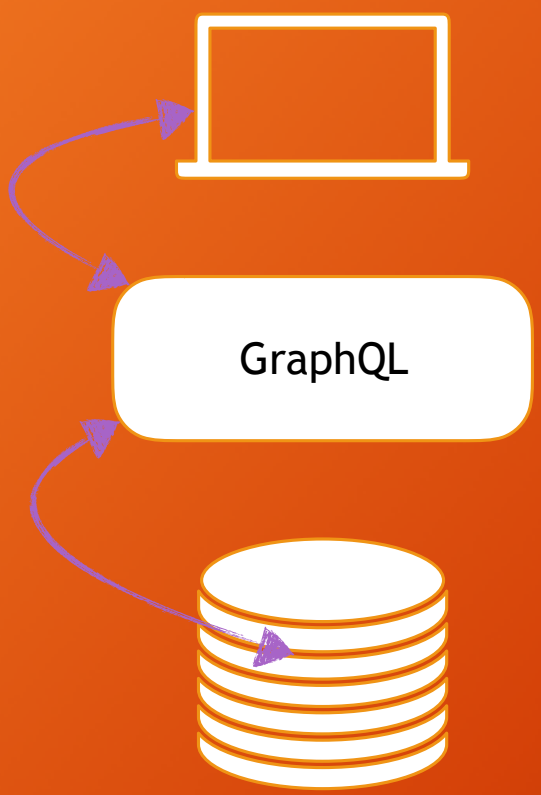
List

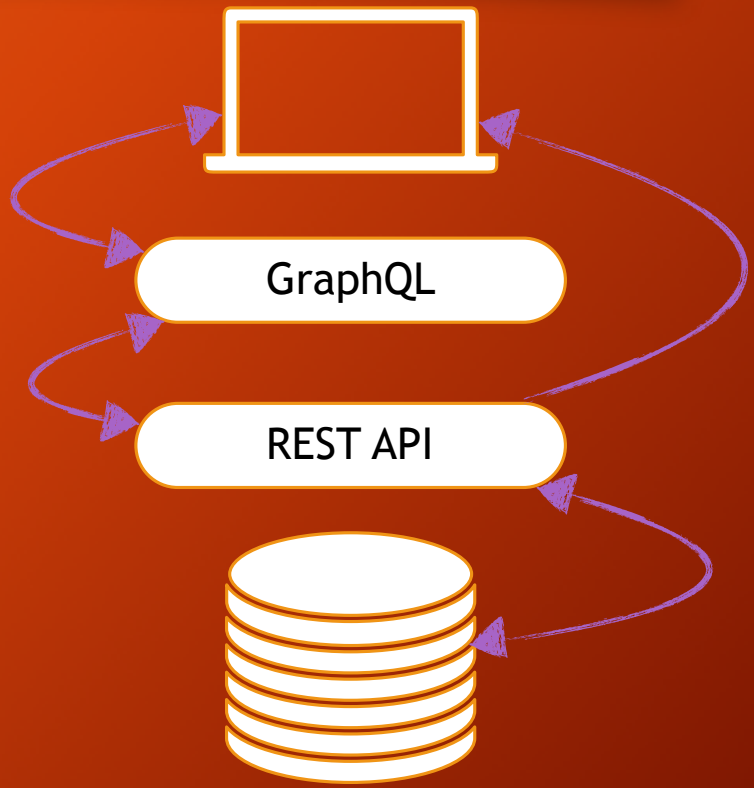
Non-Null

Resolver

```
Query: {
  employees: () => {
    return [{ name: 'Joe', salary: 15000 }, { name : 'Sue', salary: 15500
});
  },
  employee: () => {
    return { name: 'Joe', salary: 15000 });
  }
}
```

GraphQL Architecture



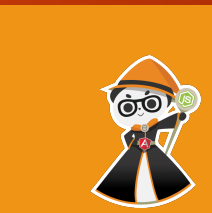


JavaScript ecosystem

- Multiple JS based implementations
  - relay
  - Apollo
  - FetchQL
  - express-graphql
  - graphql-sequelize
  - More on https://github.com/chentsulin/awesome-graphql#lib-js
- GraphQL exists outside of the JS ecosystem as well

Let’s get started!

Technology



- React for the frontend
- Apollo for the GraphQL implementation

What is Apollo?

- “The Apollo Platform brings together the best practices, design patterns, developer tools, and agile workflows for building and operating a GraphQL API on top of your existing codebase.”

- Offers GraphQL as a platform
- Client & Server side libraries
  - React, Angular, Vue etc

Technology

- Apollo Client for React
  - “Plugin and use” model
  - Out of box (client-side) caching
- Apollo Server for GraphQL
  - Ready to use
  - Define schemas, queries, mutations, custom data types etc

Two apps

- Basic React App
- List resources
- Add pagination
- Authentication
- Custom scalar
- Image manipulation via Cloudinary

Resources

- React & GraphQL at Stripe
- (A very gentle) Introduction to GraphQL
- Managing images via GraphQL
- Gatsby and GraphQL
- Apollo GraphQL's website
- Link to the material on GitHub
- GraphQL Course

Thank you

@tpiros