Leaving the REST behind with GraphQL

By Nigel Hanlon

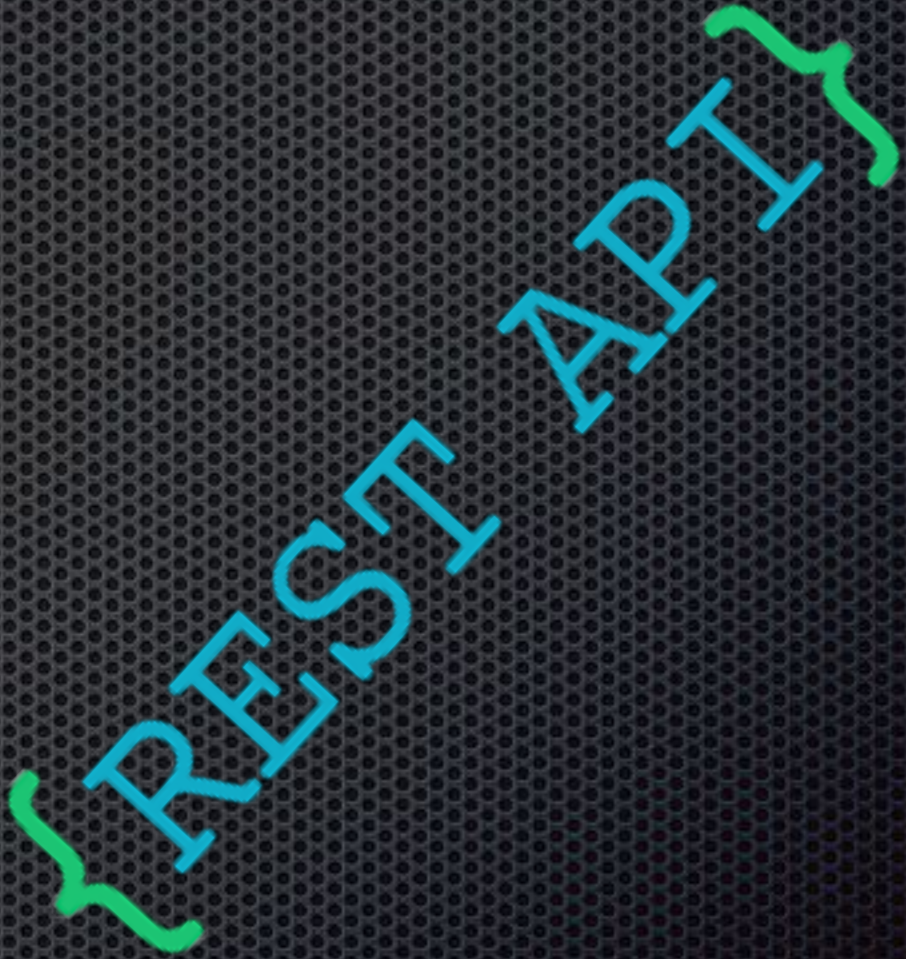
nearForm

Talk Outline

- Defining REST, its strengths and weaknesses
- Introduction to GraphQL
- Defining Schemas and Types
- Working with Resolvers
- Server side tools (getting to know Apollo and friends)
- Client side GraphQL
- Questions and Answers

Defining REST

- Stands for Representational state transfer
- An architecture for describing how clients and servers should exchange information.
- Proposed by Roy Fielding as part of his doctoral dissertation in 2000.



Six Constraints of REST

- Uniform Interface: Resources manipulated by defined methods
- Stateless: Resource state only changes in response to requests
- Cacheable: Clients can cache resources if defined cacheable.
- Client-Server: Both are replaceable so long as the interface is maintained.
- Layered System: Multiple layers may exist between client and server but should appear transparent.
- Code on Demand (Optional): A server may extend the client functionality with additional logic. (Think Java applets)

HTTP Methods for REST

Verb	CRUD	Description
POST	Create	Create a new resource
GET	Read	Get one or more resources
PUT	Update/Replace	Update a resource
PATCH	Update/Modify	Update by differences
DELETE	Delete	Delete a resource

Request Examples

HTTP	Description
POST /create_post	Create a new blog post
GET /post/1234	Get post with id 1234
PUT /post/1234	Update post with id 1234
PATCH /post/1234	Update post 1234 with these changes
DELETE /post/1234	Delete a post with id 1234

REST is Awesome!

- Freedom of implementation.
- Transport and language agnostic
- Learn once, use everywhere.
- Everything seems to have a RESTful API.

REST Sucks!

- Freedom of implementation.
- Modern app state is more complicated.
- Multiple endpoints needed for different resources.
- People often confuse REST with HTTP.

Introduction to GraphQL

What is GraphQL?

- GraphQL is a query language for APIs
- Taking your existing data, you can define types and fields to make a GraphQL service.
- No rigid endpoints, just query and receive exactly the data requested.
- GraphQL is named after the data structure it uses. Each field (node on the graph) resolves to a value.

GraphQL by Example

A Simple Blog

Post

One or more blog posts written by users with an account.

User

A user account that someone uses to write blog posts.

Comment

A comment written by a user on one or more posts.

A Simple Blog - REST API

Post

/posts
/posts/id

User

/users/id

Comment

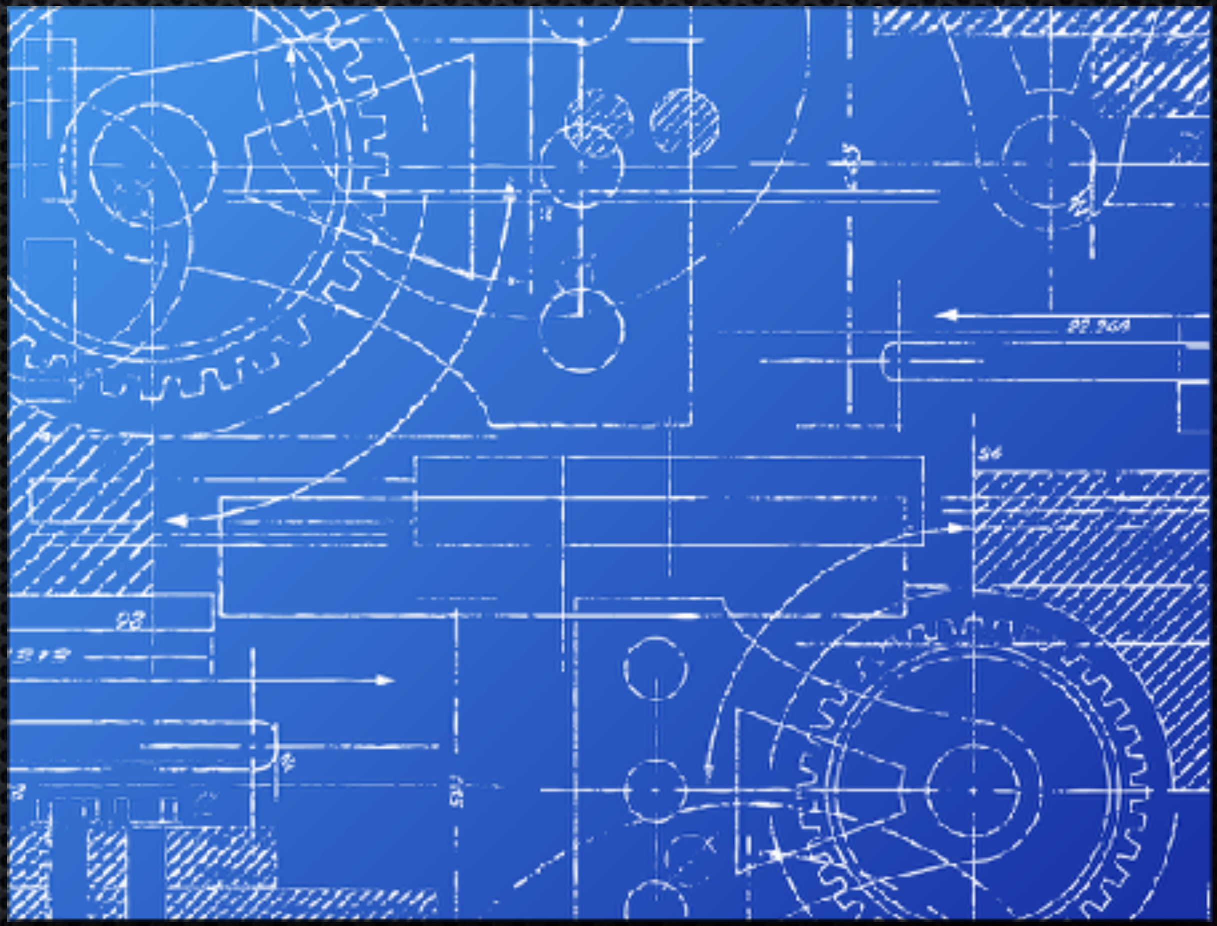
/comments/pid

Issues

- To display a page with a post, comments and author requires a minimum of 3 API calls.
- If I wanted to find all posts by a given user, which endpoint should we change?
- Adding additional functionality may require more endpoints or breaking API changes.

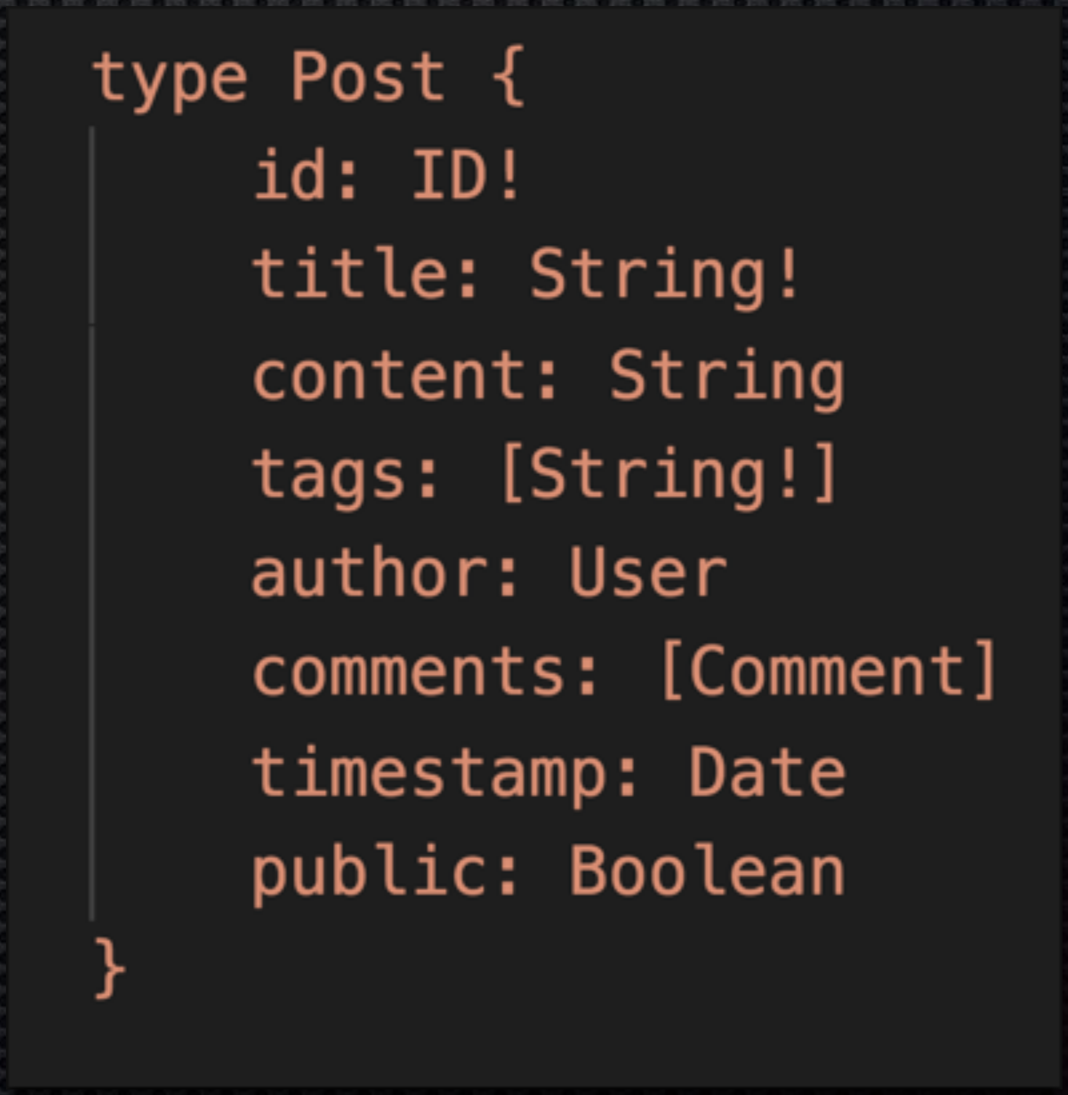
Defining Schemas and Types

- To support arbitrary queries, your API data needs to follow a schema.
- GraphQL defines its own simple syntax for schemas and queries.
- The schema defines the various data types and their fields which you can query in GraphQL.



Types and Fields

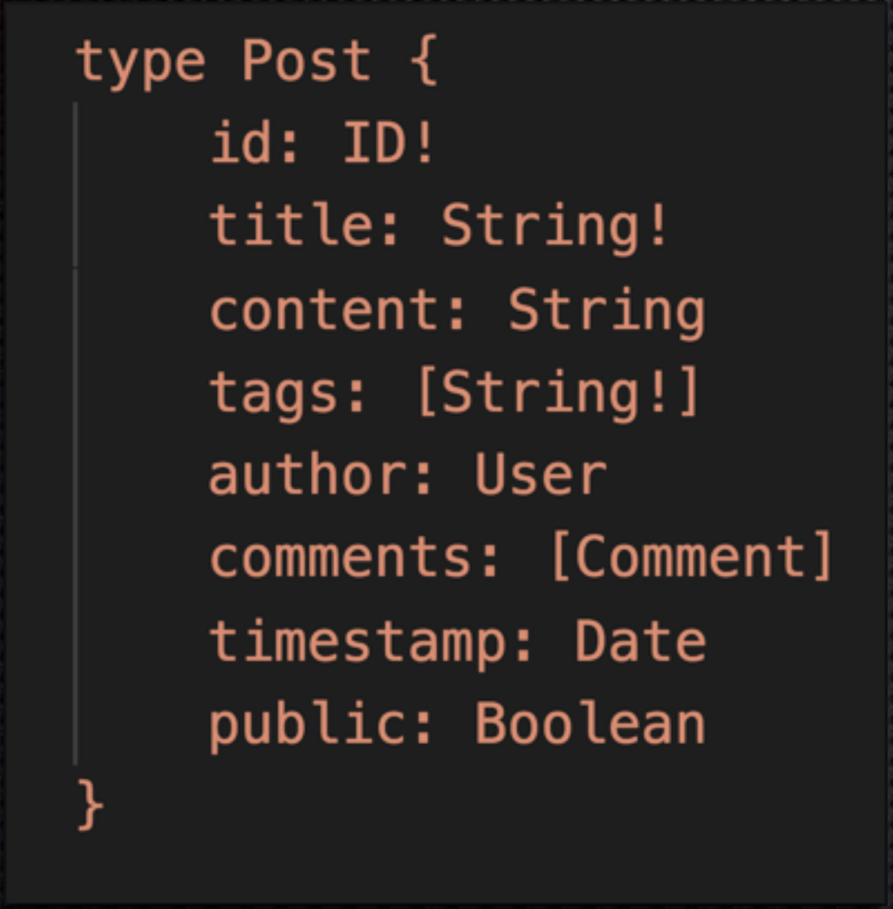
- All types have a given name and begin with the ‘type’ keyword.
- JSON like syntax.
- ! Denotes required fields
- [ ] denotes a list (array)
- Support for custom field types like Date



```
type Post {
    id: ID!
    title: String!
    content: String
    tags: [String!]
    author: User
    comments: [Comment]
    timestamp: Date
    public: Boolean
}
```

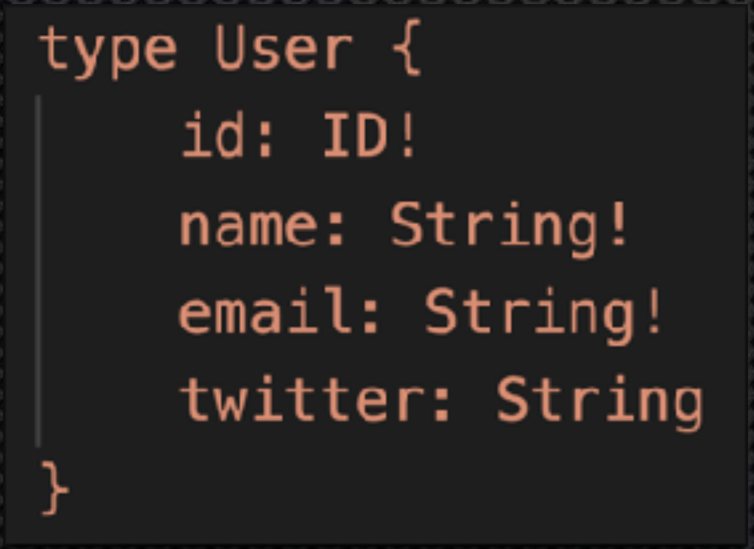
A Simple Blog - Objects

Post



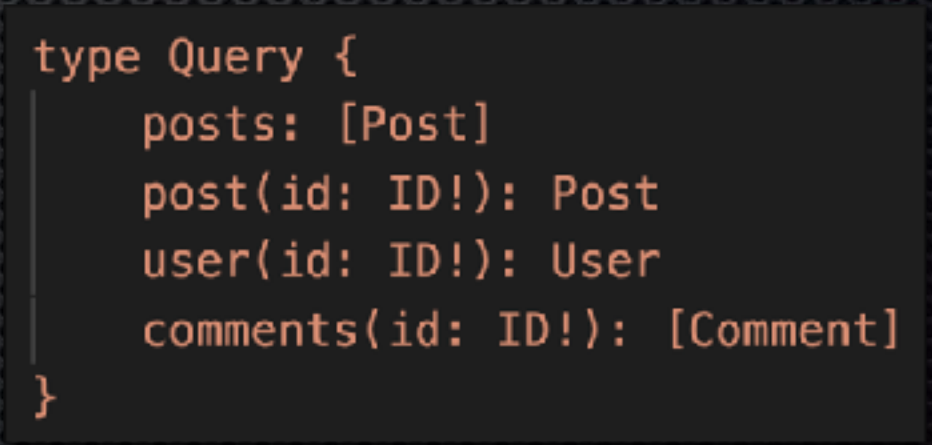
```
type Post {
    id: ID!
    title: String!
    content: String
    tags: [String!]
    author: User
    comments: [Comment]
    timestamp: Date
    public: Boolean
}
```

User



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

Comment



```
type Query {
    posts: [Post]
    post(id: ID!): Post
    user(id: ID!): User
    comments(id: ID!): [Comment]
}
```

A Simple Blog - Resolvers

- GraphQL lets you freely source your data from anywhere.
- You can use Mongo, MySQL, PostgreSQL, Elastic Search, REST API, Carrier Pigeon etc…
- Resolvers match your data to your schema. They are the glue between GraphQL and data stores.

Demo time

Client Side GraphQL

- You can simply POST GraphQL query strings to the endpoint.
- You can use a tool like Relay (https://facebook.github.io/relay/) developed by Facebook
- There is also the Apollo Client (http://dev.apollodata.com/react/)
- Google and NPM for more.

Thank you.

Twitter: @nigelhanlon
Github: https://github.com/nigelhanlon
Email: nigel@mapwolf.net