# Intermidiate GraphQL

Sameer Manek

## What is GraphQL?

GraphQL is query language for a given API / server side runtime. It aims to generalize querying data in a requested format from single and / or multiple origins, by either writing the code based on the expected patterns OR to write a seperate layer that can communicate with the API(s) and get the data in given format.

## The fundamentals - queries

Unlike REST API standard, GraphQL has just one URL endpoint to handle all the requests. It only accepts POST requests and query is written in the body of the request (key is 'query'). Heres a basic query:

```
query {
    getUser (userId: 1234) {
        name,
        address,
        email,
        phone
   }
}
```

As you see, query here we are providing the user id, and in return we are asking for some (but not all) details / fields.

## The fundamentals - query results

The result that we get from graphql is a JSON block that is arranged in the same order as the request block, for eg:

```
{
      "name": "sameer manek",
      "address": "mars",
      "email": "sameermanek@hotmail.com",
      "phone": "9409662665"
 }
```

## The fundamentals - recursive queries

```
query {
    getUser (userId: 1234) {
        name,
        address {
            city,
            pincode,
            planet
        }
        email,
        phone
   }
}
```

## The fundamentals - recursive query result

```
{
      "name": "sameer manek",
      "address": {
              "city": "muskate",
              "pincode": "000001" ,
              "planet": "mars"
      },
      "email": "sameermanek@hotmail.com",
      "phone": "9409662665"
 }
```

## The fundamentals - Mutation

Mutation refers to actions that are related to either creating new or modifying the existing data. Again we write request body, this time with data that needs to be created in the arguments section. We often written a JSON response that can signify if the operation was successful or show errors(if any). Eg:

```
mutation {
    nUser(name: "elon musk", email: "elon@tesla.com",
        phone: "+41 8000090000") {
        status,
        response {
            userId
        }
    }
}
```

## Data Structures

GraphQL supports data types to address and validate requests.
Here as well, there is a sort of primitive and derived data types.
Here is a list of data types:

| Primitive Data Type | Derived Data Types |
|---|---|
| Integer | Arrays |
| Float | Unions |
| String | Enumerations |
| Boolean | Date* |
| ID | |

## Development Guide

Graphql aims to be platform independent technology. Its not limited to just JS or PHP, but is available for almost all languages that suit you. It aspires to become a standard, rather than a technology. To support this, apart from GraphQL's native libraries, we have frameworks that help us easily develop and deploy a GraphQL Runtime. For eg. Relay (by Facebook) and Apollo.

For now, lets stick to basics.

## Code Syntax

When coding, there 2 main components of a GraphQL request:

1. Schema (where the return types are composed)
2. Request Handling functions (I forgot the real name and am feeling too lazy to lookup.. so..)

Demo..

## Conclusion

REST served us well, but, with ever growing complexity of softwares and more computation available at cheaper costs, organizations and people are adopting better API standards, GraphQL just happened to be solving most of our problems with REST, and organization like NETFLIX and apple wouldn't open source their standards so..

I'd say, we are still far from the day where we have an API standard which removes requirement of active internet connection to access online data (:P) !!

Aham Brahmasami..