**GraphQL**

Contents

[1 Document Control 3](#_Toc92189050)

[1.1 Change Record 3](#_Toc92189051)

[1.2 Reviewer 3](#_Toc92189052)

[1.3 Approver 3](#_Toc92189053)

[2 Document Purpose 3](#_Toc92189054)

[3 What is GraphQL 3](#_Toc92189055)

[4 Why Use GraphQl 3](#_Toc92189056)

[5 Applications of GraphQL 4](#_Toc92189057)

[6 GraphQL Components 4](#_Toc92189058)

[6.1 Query 4](#_Toc92189059)

[6.1.1 Fields 4](#_Toc92189060)

[6.1.2 Arguments 5](#_Toc92189061)

[6.1.3 Variables 5](#_Toc92189062)

[6.1.4 Fragments 6](#_Toc92189063)

[6.1.5 Directives 6](#_Toc92189064)

[6.2 Resolver 7](#_Toc92189065)

[6.3 Schema 7](#_Toc92189066)

[7 Difference Between GraphQL and REST 7](#_Toc92189067)

[8 Disadvantages of GraphQL 7](#_Toc92189068)

# Document Control

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Change reference** |
| 01/04/2022 | Prajeesh T S | 1.1 | Initial version |

## Reviewer

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Approval/Review Date** |
| **Anoop Jose** | Staff Software Architect |  |

## Approver

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Approval/Review Date** |
| **Anoop Jose** | Staff Software Architect |  |

# Document Purpose

Purpose of the document is to describe GraphQL

# What is GraphQL

GraphQL is an application layer server-side technology which is developed by Facebook for executing queries with existing data. GraphQL can optimize RESTful API calls. It gives a declarative way of fetching and updating your data. GraphQL helps you to load data from server to client. It enables programmers to choose the types of requests they like to make.

# Why Use GraphQl

Following are the reasons of using GraphQL:

* It provides a human-readable query.
* In GraphQL, it is very easy to deal with many databases.
* It is suited for microservices and complex systems.
* You can fetch data with a single API call.
* It helps you with query batching and caching.
* You do not face, over, and under fetching problems.
* Tailoring requests to your needs.
* It helps you to discover the schema in the appropriate format.
* GraphQL automatically keeps documentation in sync with API changes.
* GraphQL fields are used in multiple queries that can be shared to a higher component level for reuse.

# Applications of GraphQL

Here are the important applications of GraphQL:

* It provides Relay and other client frameworks
* GraphQL helps you to improve the performance of the mobile app.
* It can reduce over fetching problem to lower server-side cloud service and decrease the client-side, network usage.
* It can be used when the client application has to specify which fields are needed in long query format.
* GraphQL can be fully utilized when you have to add functionality to your old or existing API.
* It is used when you have to simplify complex API.
* When you have to aggregate data from more than one place into one convenient API.
* You can use GraphQL as an abstraction on an existing API to specify response structure based on user needs.

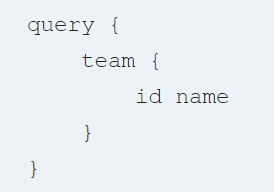
# GraphQL Components

## Query

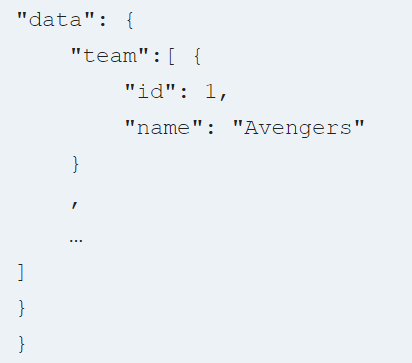
The Query is an API request made by the client machine application. It supports augments and points to arrays. Query is used to read or fetch values.

### Fields

A field simply indicates that we are asking the server for particular information. Following is a GraphQL example of a field in graphQL query.



In the above GraphQL example, we ask the server for the field called team and its subfields like id and name. The response will be follows



### Arguments

GraphQL queries allow us to pass in arguments into query fields and nested query objects. You can pass arguments to every field and every nested object in your query to further deepen your request and make multiple fetches. Arguments serve the same purpose as your traditional query parameters or URL segments in a REST API.

The query below for fetching a specific player’s kit details like shirt size or shoe size. First, we’ll have to specify that player by passing in an argument id to identify the player from the list of players, and then define the fields we want in the query payload:

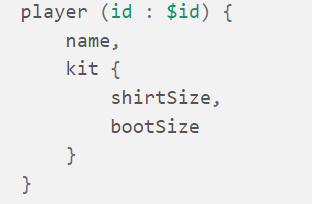


The query response will be as follows



### Variables

The arguments in the query should be dynamic, it should serve multiple values, variables are used to factor out dynamic values from queries and pass them as a separate dictionary.



### Fragments

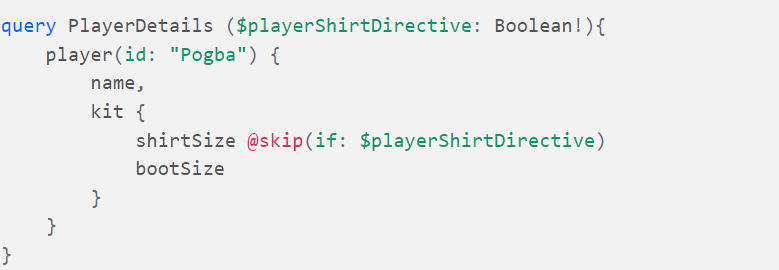
Fragments are piece of shared logic into a reusable.



### Directives

GraphQL directives provide us with a way to tell the server whether to include or skip a particular field when responding to our query. There are two built-in directives in GraphQL that helps us achieve that:

* @skip: for skipping a particular field when the value passed into it is true.
* @include: to include a particular field when the value passed into it is true.



This will now return the payload without the “shirtSize”



## Resolver

Resolvers provide the directions for converting GraphQL operation into data. They resolve the query to data by defining resolver functions.

## Schema

A GraphQL schema is the center of GraphQL implementation. It describes the functionality available to the clients which are connecting to it.

# Difference Between GraphQL and REST

|  |  |
| --- | --- |
| **GraphQL** | **REST** |
| It follows client-driven architecture | It follows server-driven architecture. |
| GraphQL can be organized in terms of a schema. | REST can be organized in terms of endpoints. |
| GraphQL is a growing community. | REST is a very large community. |
| The development speed in GraphQL is fast. | The development speed in REST is Slow. |
| The learning curve in GraphQL is difficult. | The learning curve in REST is moderate. |
| The identity is separated from how you fetch it. | The endpoint you call in REST is the identity of a particular object. |
| GraphQL provides high consistency across all platforms. | It is hard to get consistency across all platforms. |

# Disadvantages of GraphQL

Following are the GraphQL disadvantages:

* Young ecosystem
* Lack of resources on the backend part.
* Missing design pattern for a complex app.
* Performance issues with complex queries.
* Overkill for small applications
* GraphQL does not depend on the HTTP caching methods that enable storing request content.
* GraphQL does not understand files. Hence, a file uploading feature is not included in it.
* With GraphQL, be prepared to have a lot of pre-development education like learning the Schema Definition Language.