



FREE eBook

LEARNING graphql

Free unaffiliated eBook created from
Stack Overflow contributors.

#graphql

Table of Contents

About.....	1
Chapter 1: Getting started with graphql.....	2
Remarks.....	2
When should I use GraphQL?.....	2
Implementations.....	2
Client Libraries.....	2
Dev Tools.....	2
Versions.....	3
Examples.....	3
Server Installation & Implementation.....	3
GraphQL.js.....	3
Example Server.....	3
Server Middleware Alternatives.....	4
GraphQL Query Language (GQL).....	4
Query for All Film Titles from Star Wars Schema.....	4
Query for Planet Residents to Star Wars film "A New Hope".....	6
Schema Definition.....	8
Schema Object Type.....	8
Schema Query.....	8
Query Nick's Friend's Friend's Friends.....	8
Credits.....	10

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [graphql](#)

It is an unofficial and free graphql ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official graphql.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with graphql

Remarks

[GraphQL](#) is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

When should I use GraphQL?

GraphQL is intended to be the outward facing HTTP API for any kind of application but it is most powerful when the data is normalized with highly interconnected Collections|Tables|Nodes. The GraphQL Query Language (GQL) is designed to project interconnected data in a very intuitive and flexible way.

Implementations

GraphQL itself is a [Spec](#) and is [implemented](#) by a number of different programming languages. These are the most popular supported languages

- [GraphQL.js](#)
- [graphql-ruby](#)
- [Graphene \(Python\)](#)
- [Sangria \(Scala\)](#)
- [graphql-java](#)
- [graphql-clj \(Clojure\)](#)
- [graphql-go](#)
- [graphql-php](#)
- [graphql-dotnet \(C# / .Net\)](#)
- [absinthe \(Elixir\)](#)

Client Libraries

Running queries from the client side can be done with any HTTP client but client side libraries can be very beneficial.

- [Relay](#)
- [Apollo](#)
- [Lokka](#)

Dev Tools

- [graphiql](#) /'grafək(ə)l/ - An interactive in-browser GraphQL IDE.
- [libgraphqlparser](#) - A GraphQL query language parser in C++ with C and C++ APIs.
- [GraphQL Language Service](#) - An interface for building GraphQL language services for IDEs (diagnostics, autocomplete etc).

Versions

Version	Release Date
v0.5.0	2016-04-08
v0.6.0	2016-05-10
v0.6.1	2016-07-07
v0.6.2	2016-07-21
v0.7.0	2016-08-26
v0.7.1	2016-09-29
v0.7.2	2016-10-10
v0.8.0	2016-11-10
v0.8.1	2016-11-11
v0.8.2	2016-11-16

Examples

Server Installation & Implementation

GraphQL.js

[GraphQL.js](#) is a JavaScript reference implementation for GraphQL. You can install it via npm:

- Initialize npm in your project if you have not done so already: `npm init`
- Install GraphQL.js from npm: `npm install --save graphql`

Example Server

```
var { graphql, buildSchema } = require('graphql');
```

```
// Construct a schema, using GraphQL schema language
var schema = buildSchema(`
  type Query {
    hello: String
  }
`);

// The root provides a resolver function for each API endpoint
var root = {
  hello: () => {
    return 'Hello world!';
  },
};

// Run the GraphQL query '{ hello }' and print out the response
graphql(schema, '{ hello }', root).then((response) => {
  console.log(response);
});
```

Server Middleware Alternatives

- [Express GraphQL Middleware](#)
- [Hapi GraphQL Middleware](#)
- [Koa GraphQL Middleware](#)

GraphQL Query Language (GQL)

Instead of defining URL endpoints for each data resource, in GraphQL you define a single endpoint that accepts GraphQL queries. Unlike traditional database query languages, GraphQL Query Language (GQL) is a projection of data returned by a root level query. The GraphQL schema will define the data model and root level queries and mutations. GQL executes those queries and defines what data to return.

Query for All Film Titles from Star Wars Schema

```
query{
  allFilms{
    films{
      title
    }
  }
}
```

Query for Planet Residents to Star Wars film "A New Hope"


```
query{
  film(id: "ZmlsbXM6MQ=="){
    id
    title,
    planetConnection{
      planets{
        name
        residentConnection{
          residents{
            name
          }
        }
      }
    }
  }
}
```

GraphQL Query

Notice how the root query `film()` takes in the id parameter of the film "A New Hope" and the projection from GQL returns the title, planets connected to the film, and then resident names of the planets. Check out the [Star Wars GraphQL](#) site to experiment with querying in GraphQL.

Schema Definition

In [GraphQL](#) the Schema defines the root execution [queries and mutations](#) as well as the [types](#) for your data.

Schema Object Type

The `Person` type has two fields, one is a standard Scalar type and the other represents a relationship to a list of other Person types that are 'friends'. Linking other types is what makes GraphQL so powerful. Now in GraphQL Query Language (GQL), the client can traverse the friends graph without any additional code or advanced querying.

```
type Person {
  id: ID
  name: String
  friends: [Person]
}
```

Schema Query

The `person` query looks up a single person by it's id. This is the entry point to your data for clients using GQL.

```
type Query {
  person(id: ID!): Person
}
```

Query Nick's Friend's Friend's Friends

Now that we have a Person type and a person root query we can look up a person and as many degrees of separation we want of the person's friends network.

```
query {
  person(id: 'nick'){
    id
    name
    friends{
      id
      name
      friends{
        id
        name
        friends{
```

```
        id
        name
      }
    }
  }
}
```

Read **Getting started with graphql** online: <https://riptutorial.com/graphql/topic/7649/getting-started-with-graphql>

Credits

S. No	Chapters	Contributors
1	Getting started with graphql	Community , jengeb , Nikordaris