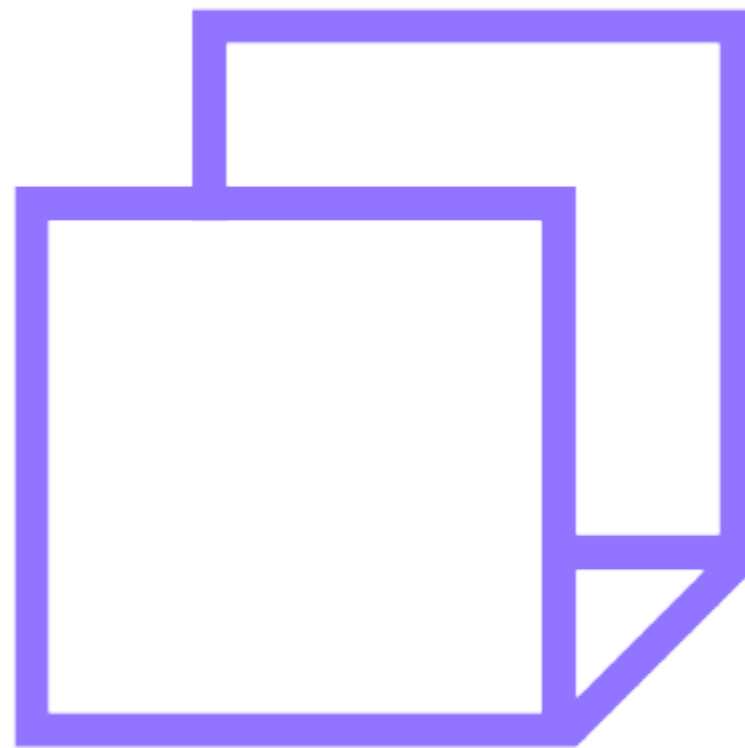


*Life is hard*  
and so is learning GraphQL

*@carolstran*



*meeshkan.com*





# global diversity CFP day

Are you a member of an underrepresented or marginalised group?

Have you always wanted to become a tech conference speaker?

Let 2020 be the year that you make that dream a reality!

Sign up for a workshop near you on Saturday, January 18th 2020.

[Organise a workshop](#)

[Celebrate Your Talk being Accepted 🎉](#)



*@carolstran*

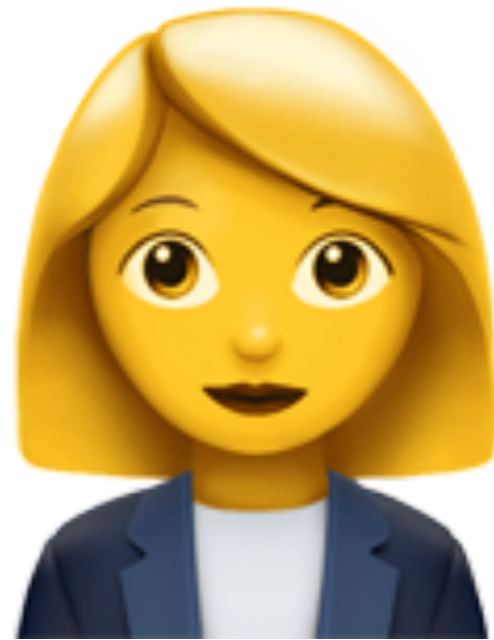


*@carolstran*



GRAPHCOOL

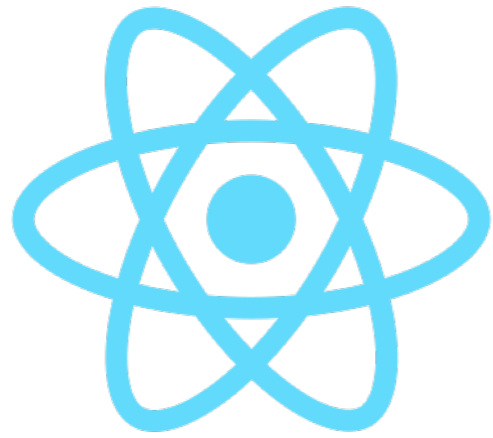


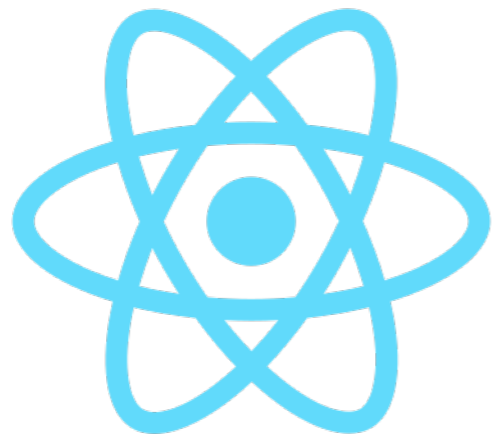


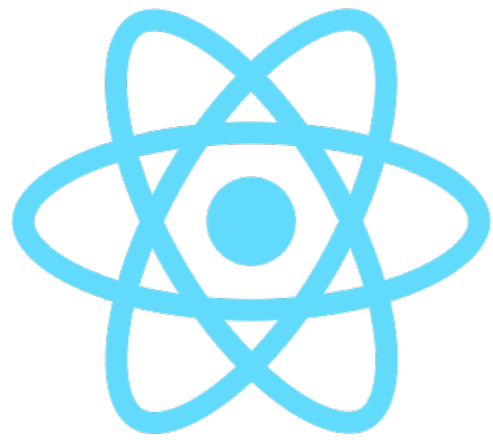
*@carolstran*



*@carolstran*

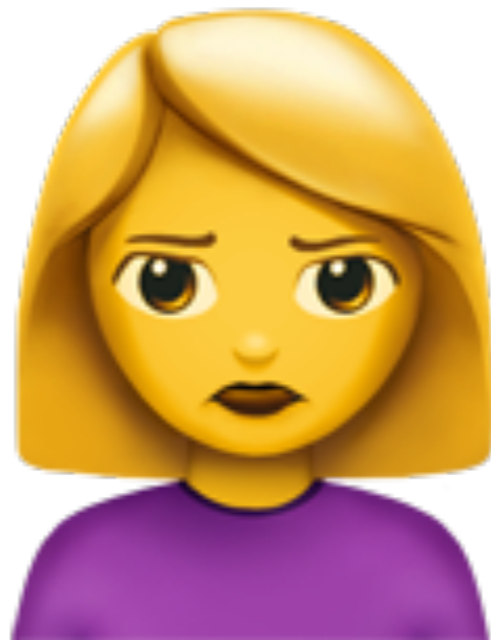








The Worst Cat

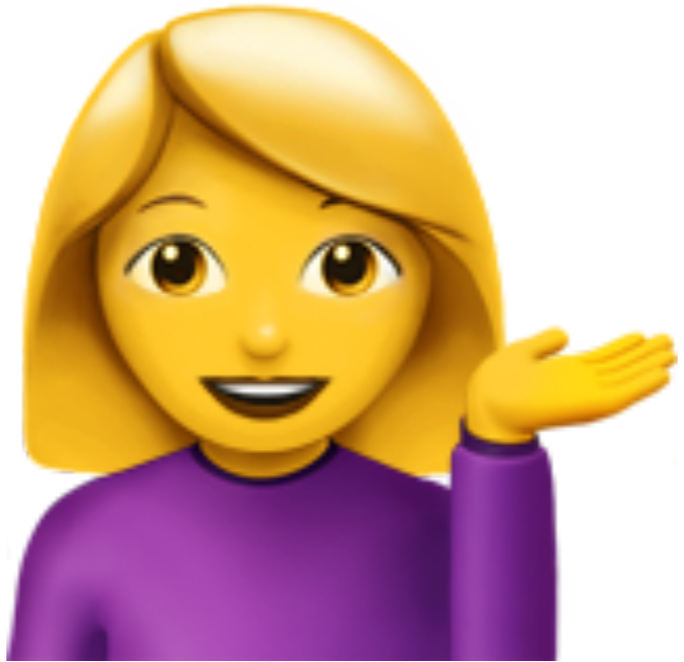


*@carolstran*



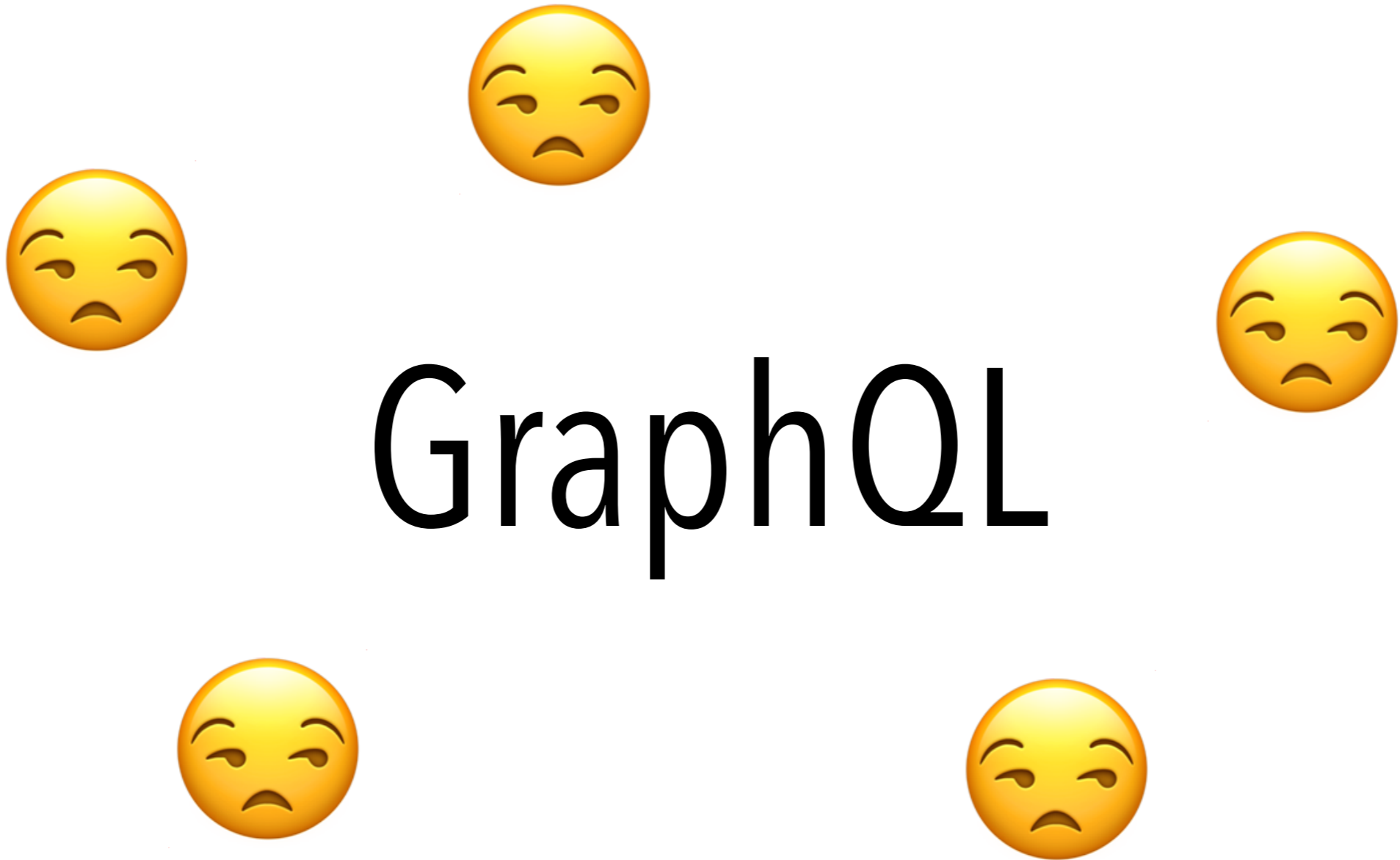
*@carolstran*





*@carolstran*

# GraphQL



# GraphQL



*@carolstran*



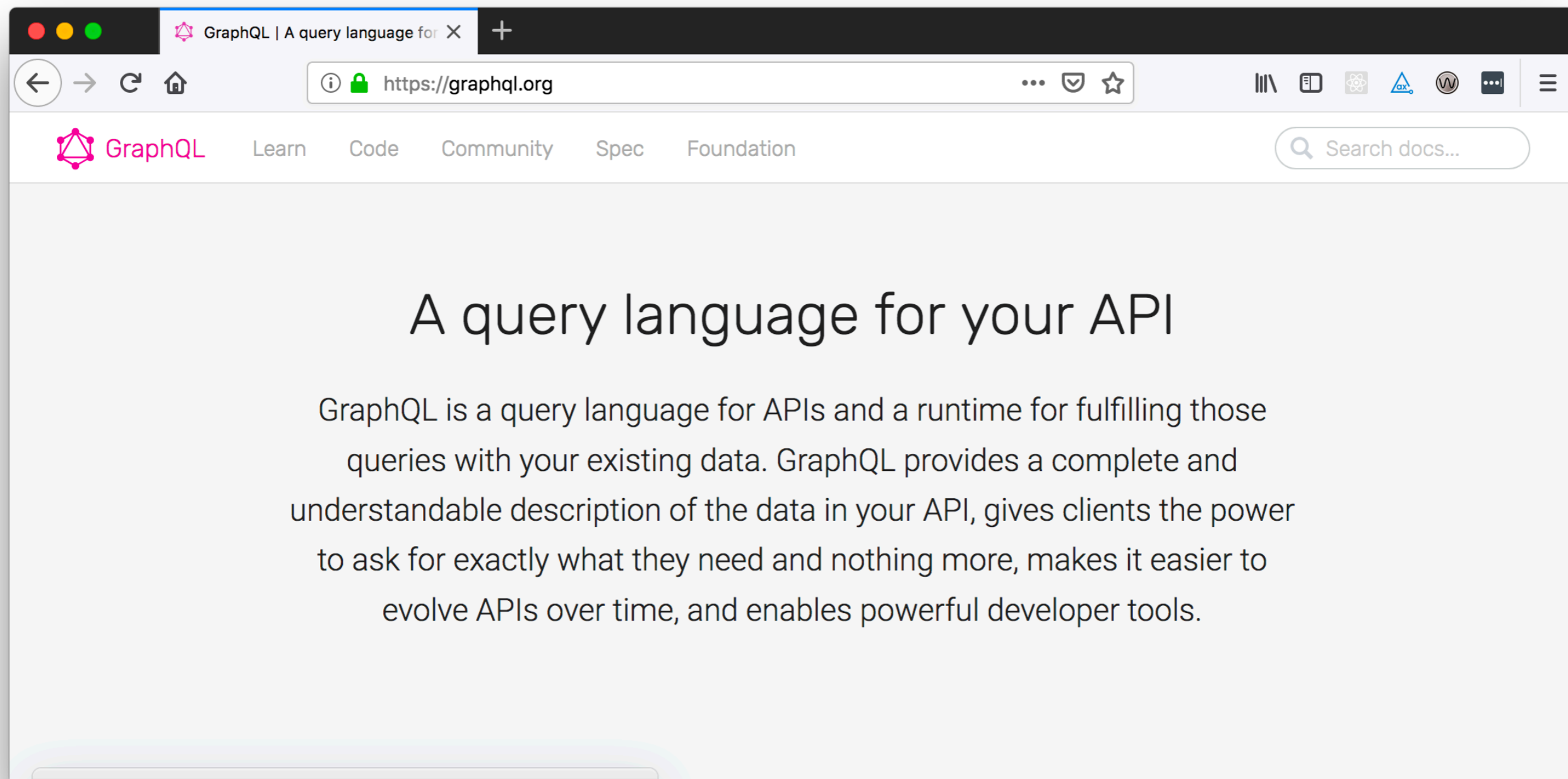
*@carolstran*



Go to GraphQL and  
"try it out"



Work your way through  
the documentation

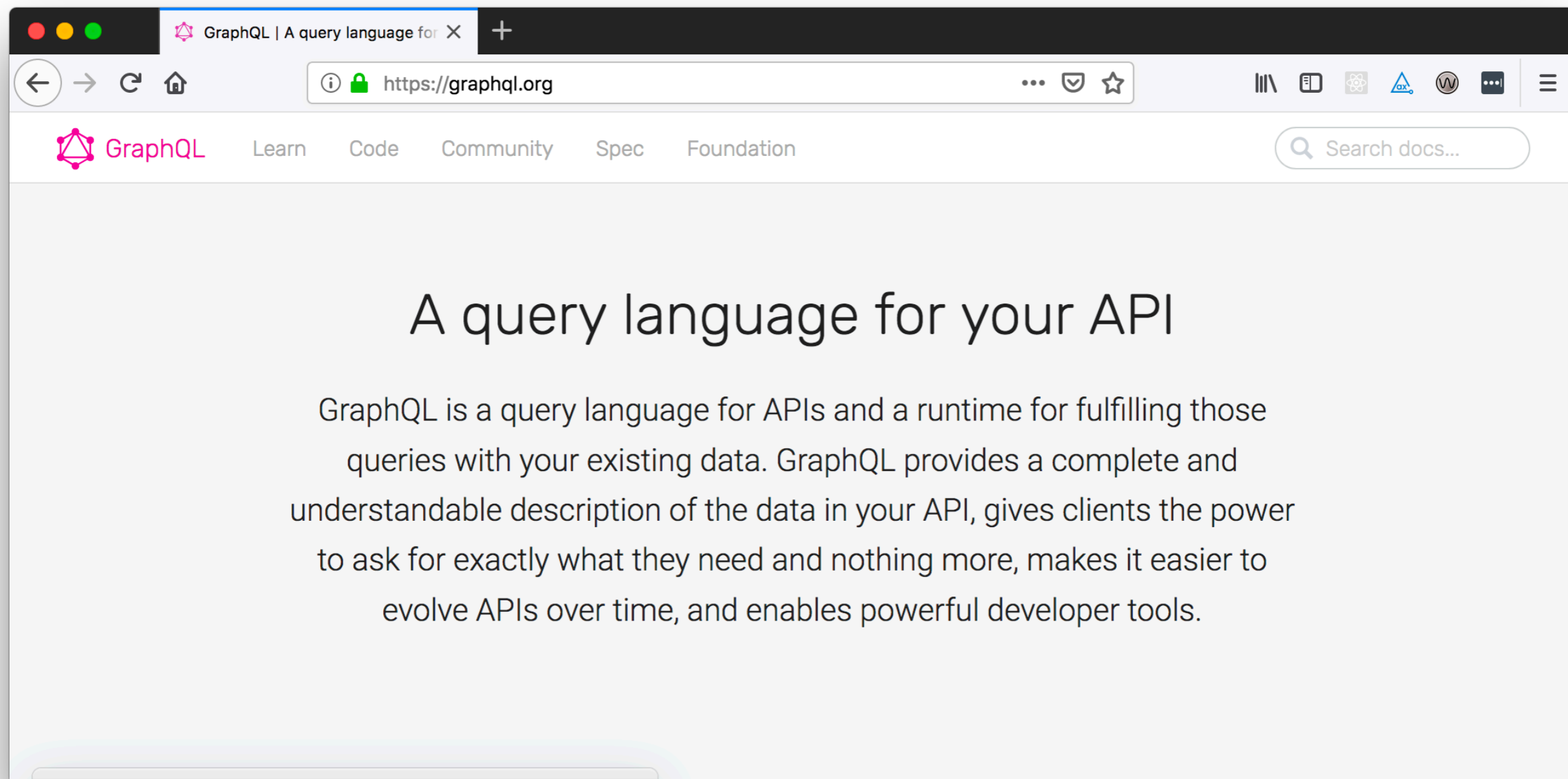


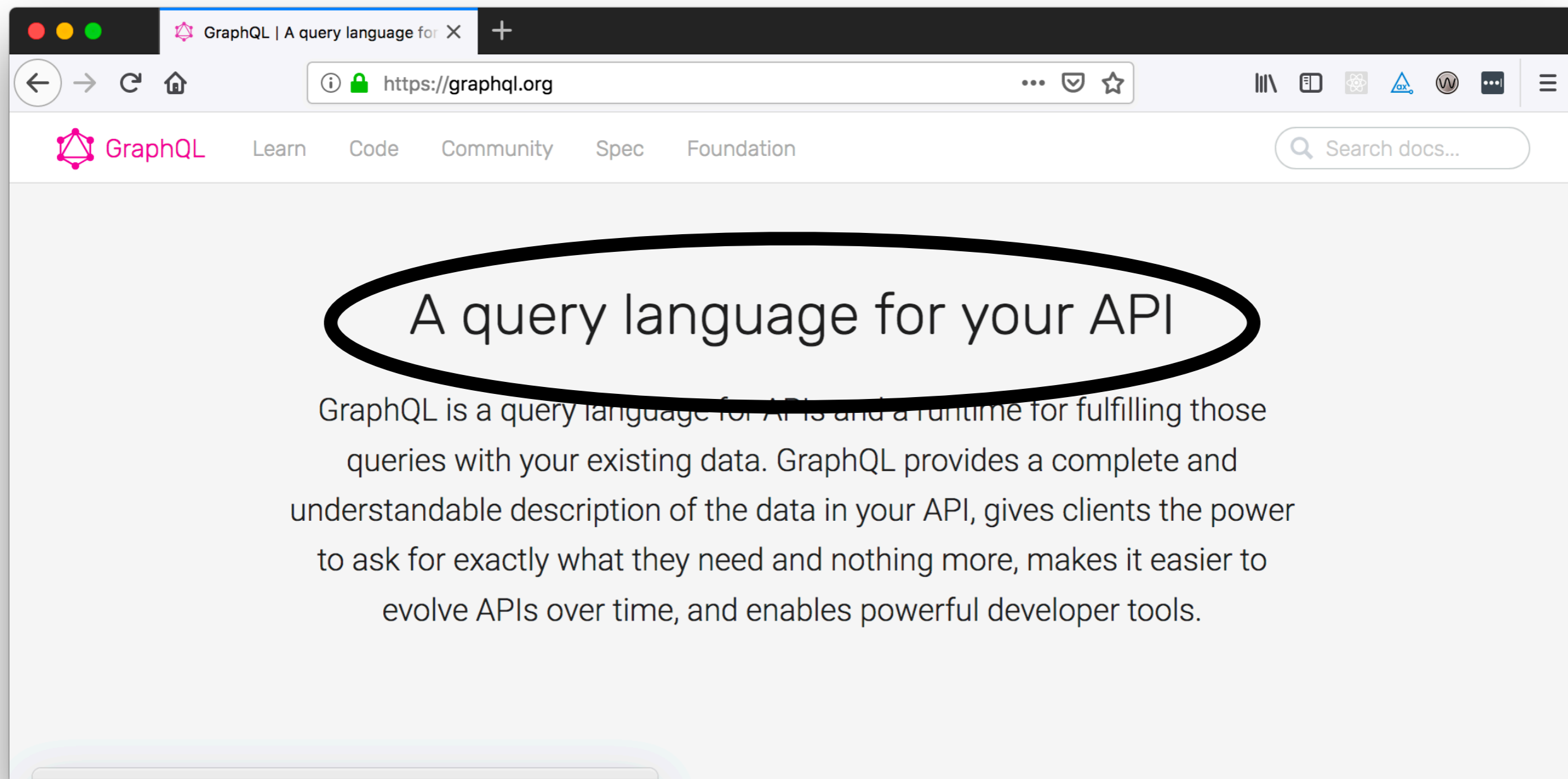


# GraphQL

# GraphQL

GraphQL





Introduction to GraphQL | GraphQL.org

https://graphql.org/learn/

GraphQL Learn Code Community Spec Foundation

Search docs...

# Introduction to GraphQL

Learn about GraphQL, how it works, and how to use it in this series of articles. Looking for documentation on how to build a GraphQL service? There are libraries to help you implement GraphQL in **many different languages**. For an in-depth learning experience with practical tutorials, visit the **How to GraphQL** fullstack tutorial website.

GraphQL is a query language for your API, and a server-side runtime for executing queries by using a type system you define for your data. GraphQL isn't tied to any specific database or storage engine and is instead backed by your existing code and data.

A GraphQL service is created by defining types and fields on those types, then providing functions for each field on each type. For example, a GraphQL service that tells us who the logged in user is (`me`) as well as that user's name might look something like this:

```
type Query {
  me: User
}

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

Along with functions for each field on each type:

- LEARN
  - Introduction
  - Queries and Mutations
    - Fields
    - Arguments
    - Aliases
    - Fragments
    - Operation Name
    - Variables
    - Directives
    - Mutations
    - Inline Fragments
  - Schemas and Types
    - Type System
    - Type Language
    - Object Types and Fields
    - Arguments
    - The Query and Mutation Types
    - Scalar Types
    - Enumeration Types
    - Lists and Non-Null
    - Interfaces
    - Union Types
    - Input Types
  - Validation
  - Execution

Introduction to GraphQL | GraphQL.org

https://graphql.org/learn/

GraphQL Learn Code Community Spec Foundation

# Introduction to GraphQL

Learn about GraphQL, how it works, and how to use it in this series of articles. Looking for documentation on how to build a GraphQL service? There are libraries to help you implement GraphQL in **many different languages**. For an in-depth learning experience with practical tutorials, visit the **How to GraphQL** fullstack tutorial website.

GraphQL is a query language for your API, and a server-side runtime for executing queries by using a type system you define for your data. GraphQL isn't tied to any specific database or storage engine and is instead backed by your existing code and data.

A GraphQL service is created by defining types and fields on those types, then providing functions for each field on each type. For example, a GraphQL service that tells us who the logged in user is (`me`) as well as that user's name might look something like this:

```
type Query {
  me: User
}

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

Along with functions for each field on each type:

LEARN

- Introduction
- Queries and Mutations
  - Fields
  - Arguments
  - Aliases
  - Fragments
  - Operation Name
  - Variables
  - Directives
  - Mutations
  - Inline Fragments
- Schemas and Types
  - Type System
  - Type Language
  - Object Types and Fields
  - Arguments
  - The Query and Mutation Types
  - Scalar Types
  - Enumeration Types
  - Lists and Non-Null
  - Interfaces
  - Union Types
  - Input Types
- Validation
- Execution



*@carolstran*





*@carolstran*

Anyone who encounters  
GraphQL, needs to be able  
to understand it





*@carolstran*

*The problems*

and what we can do about it

*@carolstran*

## *Problem*

Just because GraphQL is "self-documenting," doesn't mean you should stop writing docs

GraphQL API Explorer | GitHub | X

https://developer.github.com/v4/explorer/ 80%

GitHub Developer Docs Blog Forum Versions Search...

Signed in as carolstran. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your real, live, production data.

GraphiQL Prettify History

```
1 query {
2   viewer {
3     login
4     bio
5     location
6     isBountyHunter
7     createdAt
8     followers {
9       totalCount
10    }
11    following {
12      totalCount
13    }
14  }
15  anyPinnableItems
16 }
```

```
{
  "data": {
    "viewer": {
      "login": "carolstran",
      "bio": "",
      "location": "Berlin, Germany",
      "isBountyHunter": false,
      "createdAt": "2017-04-03T10:11:10Z",
      "followers": {
        "totalCount": 58
      },
      "following": {
        "totalCount": 25
      }
    }
  }
}
```

Schema Query x

Search Query...

The query root of GitHub's GraphQL interface.

**FIELDS**

- `codeOfConduct(key: String!): CodeOfConduct`  
Look up a code of conduct by its key
- `codesOfConduct: [CodeOfConduct]`  
Look up a code of conduct by its key
- `license(key: String!): License`  
Look up an open source license by its key
- `licenses: [License]!`  
Return a list of known open source licenses
- `marketplaceCategories(includeCategories: [String!] excludeEmpty: Boolean excludeSubcategories: Boolean): [MarketplaceCategory]!`  
Get alphabetically sorted list of Marketplace categories

© 2019 GitHub Inc. All rights reserved. Terms of service Privacy Security Support

*bit.ly/github-gql-api*

*@carolstran*

GraphQL API Explorer | GitHub

https://developer.github.com/v4/explorer/

GitHub Developer Docs Blog Forum Versions Search...

Signed in as carolstran. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your real, live, production data.

GraphiQL Prettify History

```
1 query {
2   viewer {
3     login
4     bio
5     location
6     isBountyHunter
7     createdAt
8     followers {
9       totalCount
10    }
11    following {
12      totalCount
13    }
14  }
15  anyPinnableItems
16 }
```

```
{
  "data": {
    "viewer": {
      "login": "carolstran",
      "bio": "",
      "location": "Berlin, Germany",
      "isBountyHunter": false,
      "createdAt": "2017-04-03T10:11:10Z",
      "followers": {
        "totalCount": 58
      },
      "following": {
        "totalCount": 25
      }
    }
  }
}
```

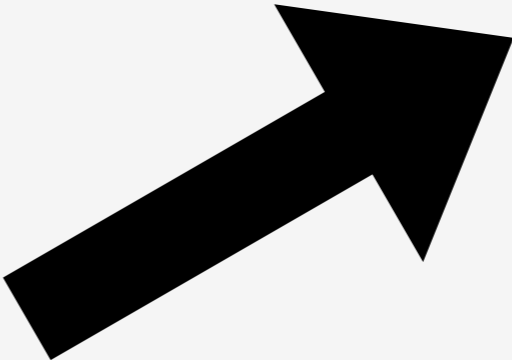
Schema Query x

Search Query...

The query root of GitHub's GraphQL interface.

**FIELDS**

- `codeOfConduct(key: String!): CodeOfConduct`  
Look up a code of conduct by its key
- `codesOfConduct: [CodeOfConduct]`  
Look up a code of conduct by its key
- `license(key: String!): License`  
Look up an open source license by its key
- `licenses: [License]!`  
Return a list of known open source licenses
- `marketplaceCategories(includeCategories: [String!] excludeEmpty: Boolean excludeSubcategories: Boolean): [MarketplaceCategory]!`  
Get alphabetically sorted list of Marketplace categories





“This type of documentation is great after you understand the domain area and the business cases and you’ve built some confidence making queries and mutations...

– Andrew Johnston

...but without more detailed documentation that covers the conceptual materials, you're not going to get there."

– Andrew Johnston

“Documenting API endpoints explains how individual tools work, explaining how to use those tools together is a whole other area of documentation effort.”

– Chris Ward

The screenshot shows a web browser window with the URL <https://www.gatsbyjs.org/docs/graphql/>. The page title is "Querying data with GraphQL". The navigation bar includes "Gatsby", "Docs", "Tutorial", "Plugins", "Features", "Blog", "Showcase", and "Contributing". A search bar and social media icons are also present. The left sidebar lists various guides, with "Querying Your Data with GraphQL" selected. The main content area features the heading "Querying data with GraphQL" and an introductory paragraph explaining that GraphQL is used to declaratively express data needs. It includes a code block for a GraphQL query and a corresponding JSON response.

Watch now: "Making Gatsby Even Greater With Themes — Better, Faster, Flexible-er".

Gatsby Docs Tutorial Plugins Features Blog Showcase Contributing

Expand All

Awesome Gatsby resources

**GUIDES**

- Preparing Your Environment
- Deploying & Hosting
- Custom Configuration
- Images, Files, and Video
- Sourcing Content and Data
- Querying Your Data with GraphQL**
- Why Gatsby Uses GraphQL
- Understanding GraphQL Syntax
- Introducing GraphQL
- Creating and Modifying Pages
- Querying Data in Pages with GraphQL

## Querying data with GraphQL

When building with Gatsby, you access your data through a query language named [GraphQL](#). GraphQL allows you to declaratively express your data needs. This is done with **queries**, queries are the representation of the data you need. A query looks like this:

```
GRAPHQL
{
  site {
    siteMetadata {
      title
    }
  }
}
```

Which returns this:

```
JSON
{
  "site": {
    "siteMetadata": {
      "title": "A Gatsby site!"
    }
  }
}
```

*Problem*

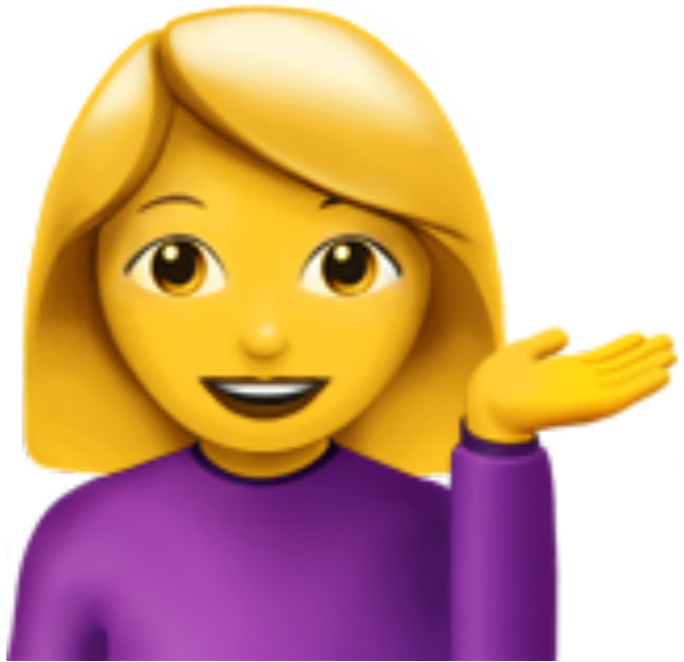
GraphQL is dominated by  
the JavaScript community

*@carolstran*



*[bit.ly/graphql-def](https://bit.ly/graphql-def)*

*@carolstran*



*@carolstran*

JavaScript

C/C++

Rust

TypeScript

Swift

Go

Ruby

.NET

Scala

PHP

Erlang

Clojure

Python

Elixir

R

Java

Haskell

(and more)



Introduction to GraphQL | GraphQL.org

https://graphql.org/learn/

GraphQL Learn Code Community Spec Foundation

Search docs...

# Introduction to GraphQL

Learn about GraphQL, how it works, and how to use it in this series of articles. Looking for documentation on how to build a GraphQL service? There are libraries to help you implement GraphQL in [many different languages](#). For an in-depth learning experience with practical tutorials, visit the [How to GraphQL](#) fullstack tutorial website.

GraphQL is a query language for your API, and a server-side runtime for executing queries by using a type system you define for your data. GraphQL isn't tied to any specific database or storage engine and is instead backed by your existing code and data.

A GraphQL service is created by defining types and fields on those types, then providing functions for each field on each type. For example, a GraphQL service that tells us who the logged in user is (`me`) as well as that user's name might look something like this:

```
type Query {
  me: User
}

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

Along with functions for each field on each type:

- LEARN
  - [Introduction](#)
  - [Queries and Mutations](#)
    - Fields
    - Arguments
    - Aliases
    - Fragments
    - Operation Name
    - Variables
    - Directives
    - Mutations
    - Inline Fragments
  - [Schemas and Types](#)
    - Type System
    - Type Language
    - Object Types and Fields
    - Arguments
    - The Query and Mutation Types
    - Scalar Types
    - Enumeration Types
    - Lists and Non-Null
    - Interfaces
    - Union Types
    - Input Types
  - [Validation](#)
  - [Execution](#)

The image shows a browser window displaying the GraphQL.org/learn page. The browser's address bar shows the URL https://graphql.org/learn/. The page has a navigation bar with links for GraphQL, Learn, Code, Community, Spec, and Foundation, along with a search bar. The main heading is "Introduction to GraphQL". A paragraph of introductory text is highlighted with a thick black oval. Below this, there is a paragraph explaining GraphQL as a query language, followed by a paragraph about creating a GraphQL service. A code block shows a GraphQL schema for a Query and User type. The right sidebar contains a "LEARN" section with a list of topics including Introduction, Queries and Mutations, Fields, Arguments, Aliases, Fragments, Operation Name, Variables, Directives, Mutations, Inline Fragments, Schemas and Types, Type System, Type Language, Object Types and Fields, Arguments, The Query and Mutation Types, Scalar Types, Enumeration Types, Lists and Non-Null, Interfaces, Union Types, Input Types, Validation, and Execution.

Introduction to GraphQL

Learn about GraphQL, how it works, and how to use it in this series of articles. Looking for documentation on how to build a GraphQL service? There are libraries to help you implement GraphQL in **many different languages**. For an in-depth learning experience with practical tutorials, visit the **How to GraphQL** fullstack tutorial website.

GraphQL is a query language for your API, and a server-side runtime for executing queries by using a type system you define for your data. GraphQL isn't tied to any specific database or storage engine and is instead backed by your existing code and data.

A GraphQL service is created by defining types and fields on those types, then providing functions for each field on each type. For example, a GraphQL service that tells us who the logged in user is (`me`) as well as that user's name might look something like this:

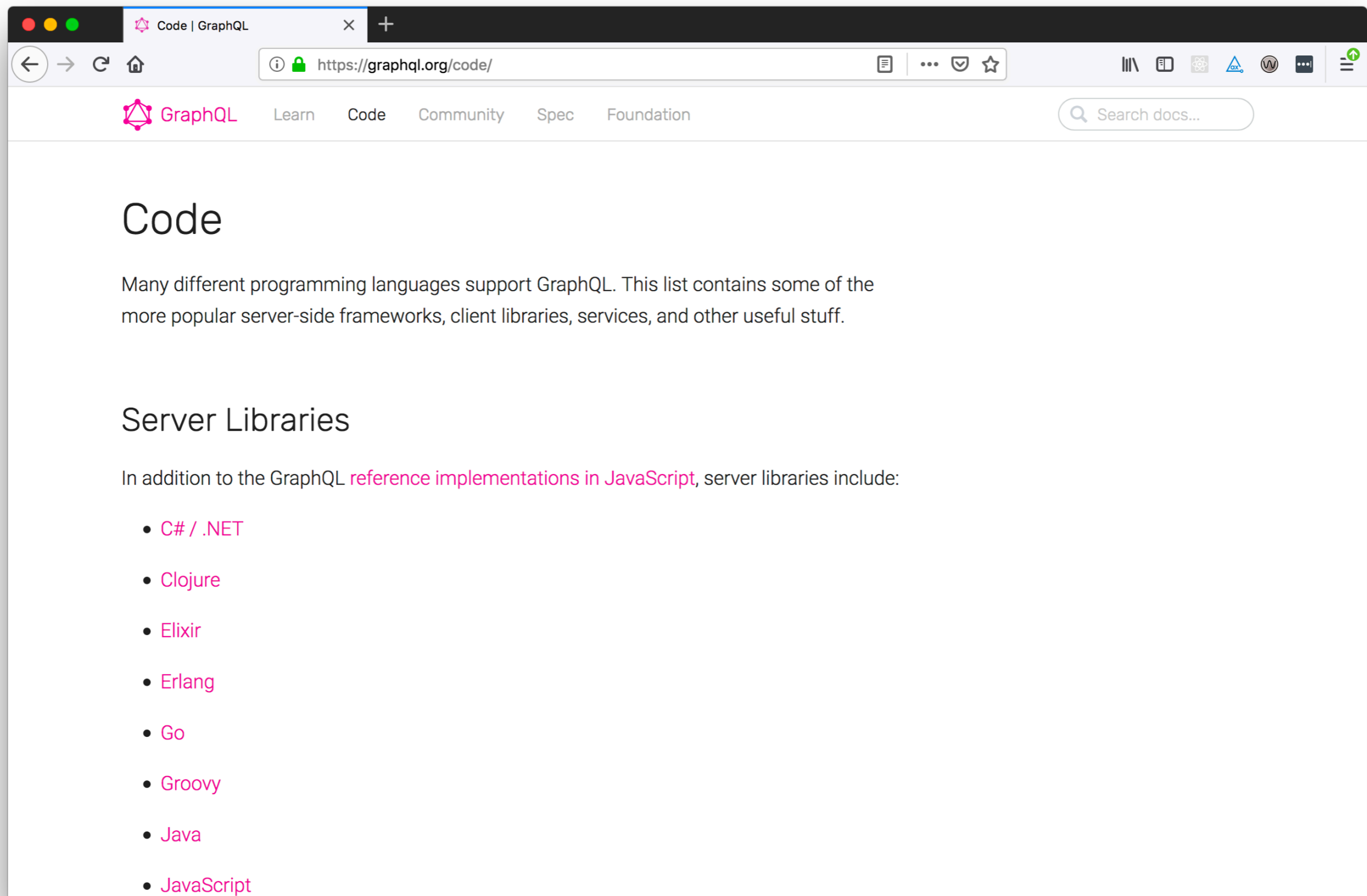
```
type Query {
  me: User
}

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

Along with functions for each field on each type:

LEARN

- Introduction
- Queries and Mutations
- Fields
- Arguments
- Aliases
- Fragments
- Operation Name
- Variables
- Directives
- Mutations
- Inline Fragments
- Schemas and Types
- Type System
- Type Language
- Object Types and Fields
- Arguments
- The Query and Mutation Types
- Scalar Types
- Enumeration Types
- Lists and Non-Null
- Interfaces
- Union Types
- Input Types
- Validation
- Execution



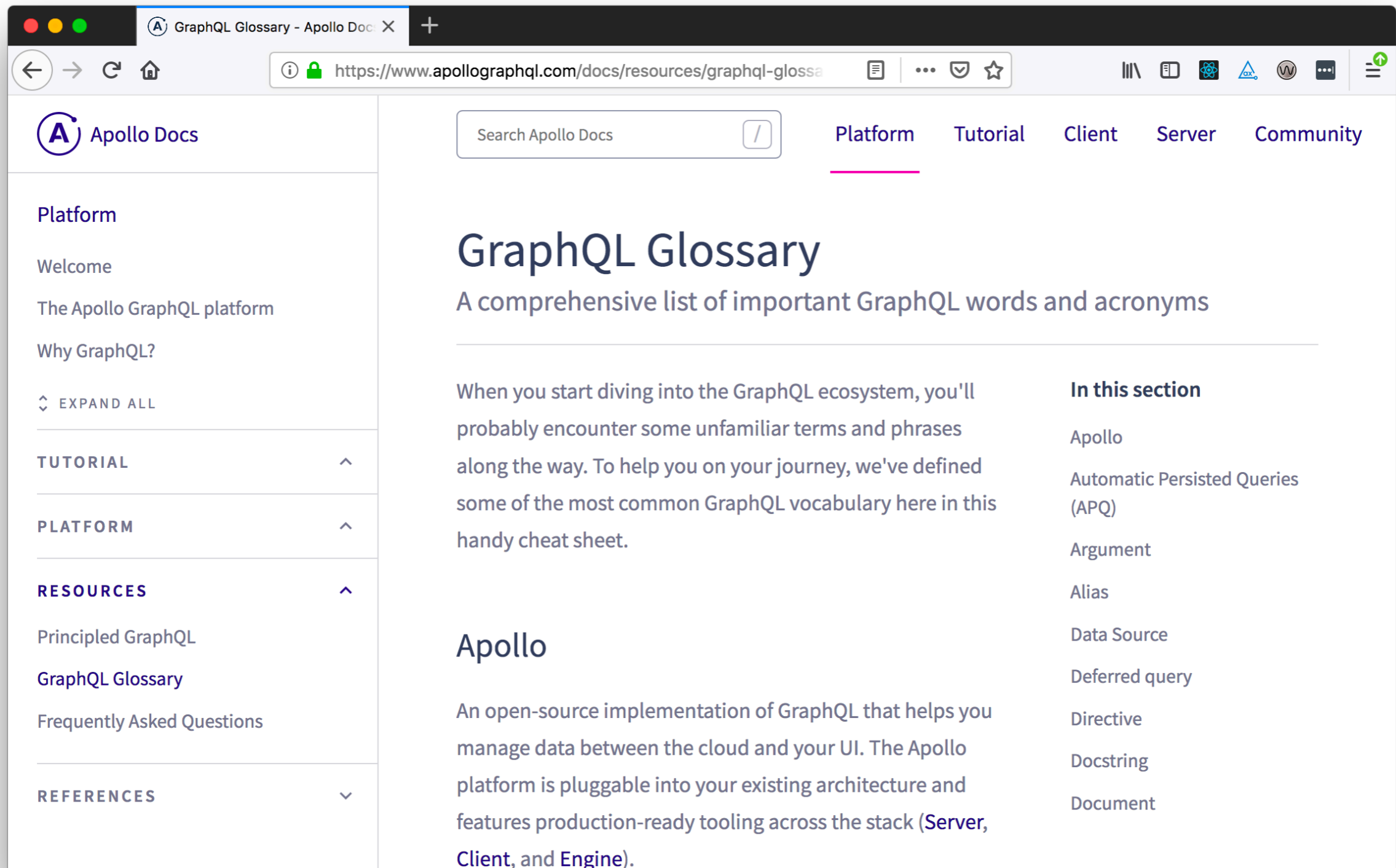
*Problem*

Assumed knowledge  
and specific terminology

*@carolstran*

Everyone is a beginner  
at some point

Listing pre-reqs for a tutorial  
will help prevent headaches



*bit.ly/graphql-glossary*

*@carolstran*

The image shows a browser window with the URL `https://developers.facebook.com/docs/graph-api/overview`. The page header includes the Facebook logo, navigation links for Docs, Tools, and Support, a search bar, and a Log In button. The main content area is titled "Overview" and includes a sidebar with links to Graph API, Overview, Rate Limits, Using the Graph API, FAQ, Reference, Webhooks, Advanced, Changelog, and Server-Sent Events. The main text explains that the Graph API is the primary way to get data into and out of the Facebook platform. It is an HTTP-based API used for querying data, posting stories, managing ads, and uploading photos. The section "The Basics" defines nodes as individual objects (User, Photo, Page, Comment), edges as connections between objects, and fields as data about an object. It concludes by stating that nodes are used for specific objects, edges for collections, and fields for data about objects or collections.

facebook for developers Docs Tools Support Search developers.facebook.com Log In

Graph API

Overview

Rate Limits

Using the Graph API

FAQ

Reference

Webhooks

Advanced

Changelog

Server-Sent Events

On This Page

# Overview

The Graph API is the primary way to get data into and out of the Facebook platform. It's an HTTP-based API that apps can use to programmatically query data, post new stories, manage ads, upload photos, and perform a wide variety of other tasks.

## The Basics

The Graph API is named after the idea of a "social graph" — a representation of the information on Facebook. It's composed of:

- **nodes** — basically individual objects, such as a User, a Photo, a Page, or a Comment
- **edges** — connections between a collection of objects and a single object, such as Photos on a Page or Comments on a Photo
- **fields** — data about an object, such as a User's birthday, or a Page's name

Typically you use nodes to get data about a specific object, use edges to get collections of objects on a single object, and use fields to get data about a single object or each object in a collection.

*bit.ly/fb-graphapi*

*@carolstran*



# *Quick wins*

some short documentation tips

*@carolstran*

*Quick win*

Be confident with your  
definitions

*@carolstran*

*Quick win*

There's a time and  
place for analogies

*@carolstran*



*Quick win*

There's a time and  
place for analogies

(and that place is Twitter)

*@carolstran*

## GraphQL & Rest: A burger comparison

`https://your-api.com/burger/`



```
query getBurger {  
  burger {  
    bun  
    patty  
    bun  
    lettuce  
  }  
}
```



*Quick win*

Don't integrate tools or  
services without explanation

*@carolstran*

*Quick win*

Be transparent about  
any downfalls

*@carolstran*



# *More resources*

for learning GraphQL and beyond

*@carolstran*

*Resource*

HowToGraphQL.com

*@carolstran*

How to GraphQL - The Fullstack X

https://www.howtographql.com

HOW TO GRAPHQL Search tutorials...

GraphQL Frontend Backend

Tickets are now available for GraphQL Conf! ↗

# The Fullstack Tutorial for GraphQL

The free and open-source tutorial to learn all around GraphQL to go from zero to production.

WATCH OVERVIEW

Start with Introduction

GraphQL Fundamentals 42 MIN TOTAL

Introduction 5 MIN

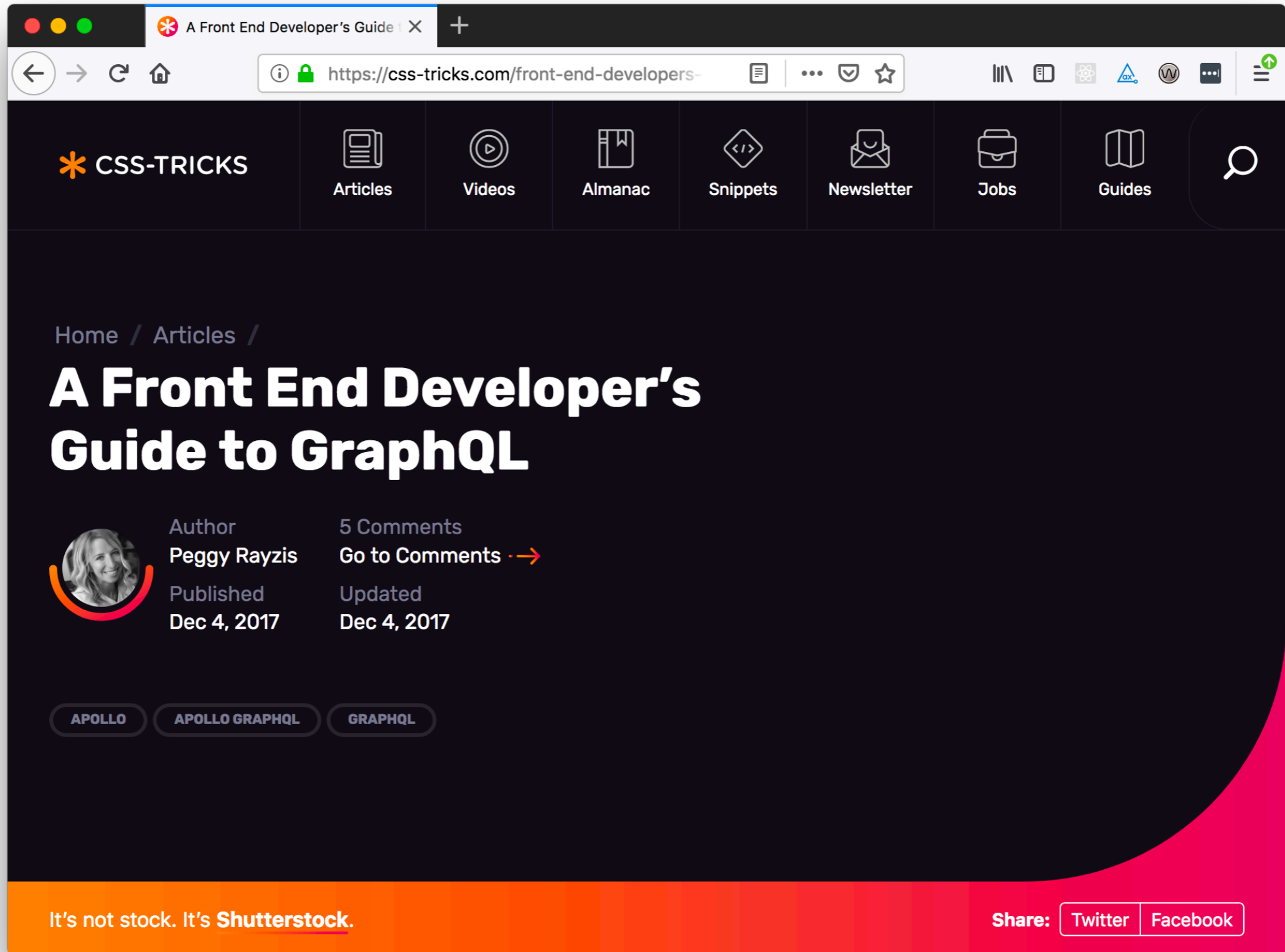
GraphQL is the better REST 12 MIN

*Resource*

# A Frontend Developer's Guide to GraphQL

*[github.com/peggyrayzis/guide-to-graphql](https://github.com/peggyrayzis/guide-to-graphql)*

*@carolstran*



*css-tricks.com/front-end-developers-guide-graphql/*

*@carolstran*

*Resource*

# GitHub's GraphQL API

*[bit.ly/github-gql-api](https://bit.ly/github-gql-api)*

*@carolstran*

GraphQL API Explorer | GitHub | X

https://developer.github.com/v4/explorer/ 80%

GitHub Developer Docs Blog Forum Versions Search...

GitHub GraphQL API Signed in as carolstran. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your real, live, production data.

GraphiQL Prettify History

```

1 query {
2   viewer {
3     login
4     bio
5     location
6     isBountyHunter
7     createdAt
8     followers {
9       totalCount
10    }
11    following {
12      totalCount
13    }
14  }
15  anyPinnableItems
16 }

```

```

{
  "data": {
    "viewer": {
      "login": "carolstran",
      "bio": "",
      "location": "Berlin, Germany",
      "isBountyHunter": false,
      "createdAt": "2017-04-03T10:11:10Z",
      "followers": {
        "totalCount": 58
      },
      "following": {
        "totalCount": 25
      }
    }
  }
}

```

Schema Query

Search Query...

The query root of GitHub's GraphQL interface.

**FIELDS**

**codeOfConduct(key: String!): CodeOfConduct**  
Look up a code of conduct by its key

**codesOfConduct: [CodeOfConduct]**  
Look up a code of conduct by its key

**license(key: String!): License**  
Look up an open source license by its key

**licenses: [License]!**  
Return a list of known open source licenses

**marketplaceCategories( includeCategories: [String!] excludeEmpty: Boolean excludeSubcategories: Boolean ): [MarketplaceCategory]!**  
Get alphabetically sorted list of Marketplace categories

© 2019 GitHub Inc. All rights reserved. Terms of service Privacy Security Support

*bit.ly/github-gql-api*

*@carolstran*

*Resource*

# Community Resources on official GraphQL docs

*[graphql.org/community](https://graphql.org/community)*

*@carolstran*



The image shows a browser window with the URL <https://graphql.org/community/>. The page features the GraphQL logo and navigation links for Learn, Code, Community, Spec, and Foundation. The main heading is "Community Resources".

## Community Resources

**Stack Overflow**

Many members of the community use Stack Overflow to ask and answer questions. [Read through the existing questions tagged with graphql or ask your own!](#)

**Facebook Group**

Join the [GraphQL Facebook Group](#) sharing and discovering new content. The GraphQL Facebook group is the preferred venue for announcements and broader discussion.

**Twitter**

Use the [#graphql hashtag](#) on Twitter to join the conversation.

**COMMUNITY**

- [Community Resources](#)
- [Blogs](#)
- [Videos](#)
- [Upcoming Events](#)
- [Upcoming Events](#)
- [Meetups](#)

*[graphql.org/community](https://graphql.org/community)*

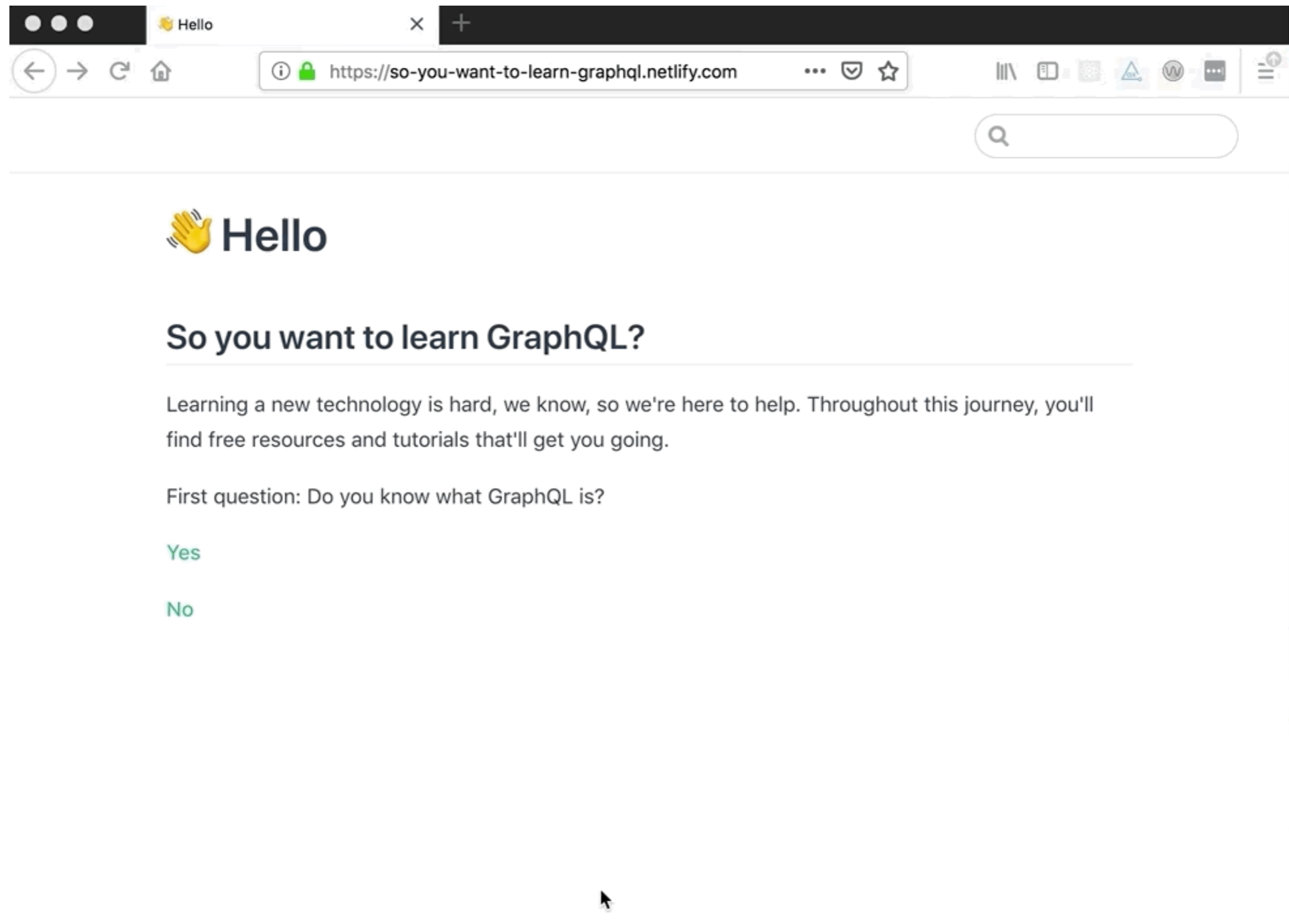
*@carolstran*

*Resource*

<https://so-you-want-to-learn-graphql.netlify.com/>

*github.com/carolstran/so-you-want-to-learn-graphql*

*@carolstran*



*One last thing*

(but actually, I promise)

*@carolstran*

*“Once you understand monads, you immediately become incapable of explaining them to anyone else” Lady Monadgreen’s curse ~ Gilad Bracha (used famously by Douglas Crockford)*

Please don't let GraphQL  
become the new monads