GraphQL for the REST of Us

Filip Stojanac

Agenda

API Pain

What is GraphQL?

Compare API Approaches

Why GraphQL?

GraphQL

- Documentation
- Queries
- Mutations
- Server
- Downsides

Let’s Talk API Pain

API Pain

Transformations

Over fetching

Under fetching

Long chains API calls

Waiting for API changes

Long API design convos

API proxy boilerplate maintenance

Lack of usage data

Documentation

Discovery

GraphQL

No transformations

No over fetching

No under fetching

No waiting for API modifications

No long chains API calls

No long debates on API design

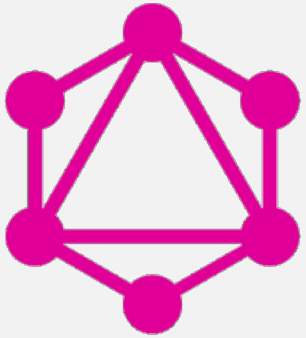
No API proxy boilerplate

No documentation worries

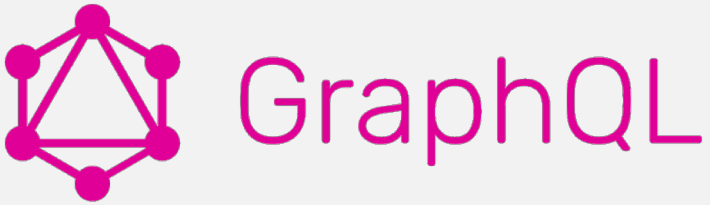
No discovery issues

What is GraphQL?

Created in 2012. Open Sourced in 2015

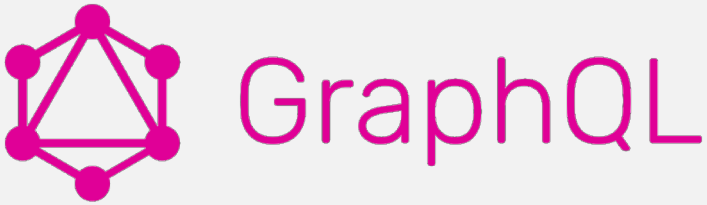


GraphQL



GraphQL is a specification

- Query language
- Type system
- Works with any language
- Uses a single HTTP endpoint
- Not tied to a specific DB technology



On the server

Describe the data structure.

On the client

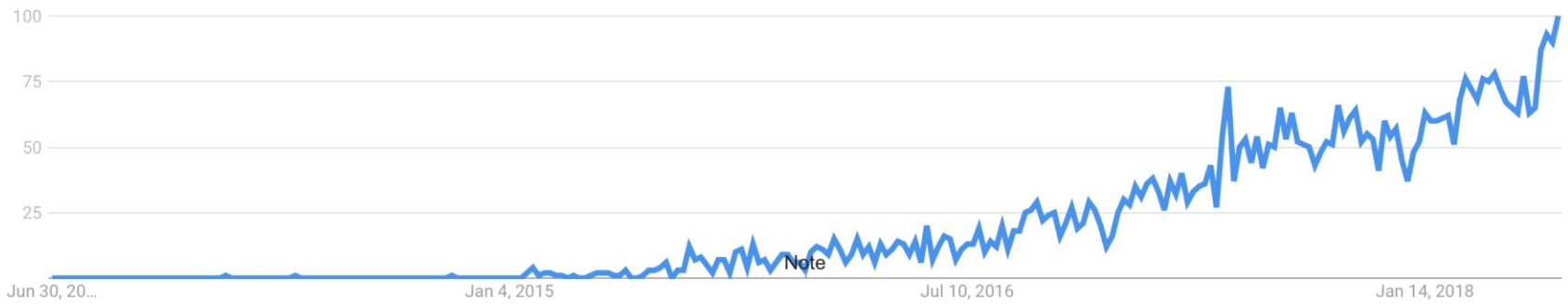
Request the ***exact*** data you want.

GraphQL
Topic

+ Compare

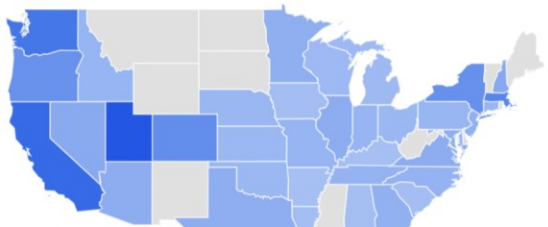
United States | Past 5 years | All categories | Web Search

Interest over time



Interest by subregion

Subregion

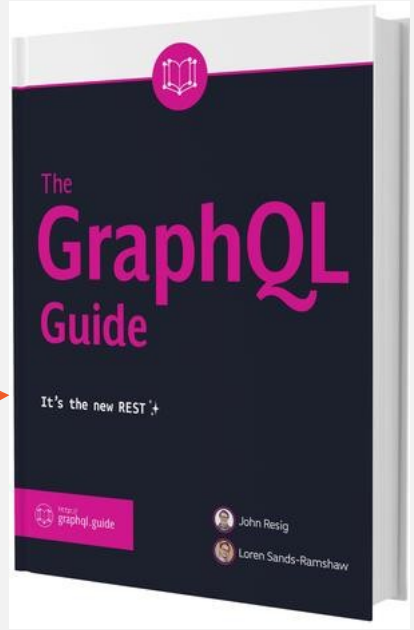


1	Utah	100
2	California	83
3	Washington	72

GraphQL is the true successor to REST APIs.

John Resig

Subtitle: It's the new REST



Let’s Compare API Approaches

Let’s Compare API Approaches

Get first 10 blog posts.
Include author and company.

REST: Get 10 Posts w/ Author & Company

/post?limit=10
/author/1
/author/2
/author/3
/author/4
/author/5
…
/company/1
/company/2
/company/3
/company/4
/company/5
…

21 requests
Request waterfall (need author before company)
Over-fetching (sparse fieldsets?)
Manually concat on client

REST Include: Get 10 Posts w/ Author & Company

/post?limit=10&include=author.company

1 request
No standard
Rare
Over-fetching
Desired shape?

Who has an API that supports this?

GraphQL: Get 10 Posts w/ Author & Company

```
query {
   posts(first:10) {
      id
      title
      author {  id
         firstName
         company
            {  id

            companyNa
         } me
      }
   }
}
```

1 request
Desired properties
Desired relationships
Desired shape
Desired property names

Why GraphQL?

1 Get *Exactly* What You Want

Custom Client-side Queries

No separate web and mobile APIs!

🎉

UI code and data requirements, side-by-side

Each component declares the data it needs

Heirarchical queries

Joins

Filtering

Sorting

Apps change fast

GraphQL lets the clients declare what they need.

2 Full Specification

No more REST arguments

GraphQL

Prerelease	Working Draft	Tue, Nov 20, 2018	
Latest Release	June 2018	Sun, Jun 10, 2018	Release Notes
	October 2016	Mon, Oct 31, 2016	Release Notes
	April 2016	Thu, Apr 7, 2016	Release Notes
	October 2015	Thu, Oct 1, 2015	Release Notes
	July 2015	Thu, Jul 2, 2015	Release Notes

facebook.github.io/graphql/


REST denominations

Versioning

Header

URL

None (HATEOAS)

HTTP Verbs

GET

POST

PUT?

DELETE?

PATCH?

Filters

Single filter

- /todos?filter=key%3Dvalue
- /todos?filterKey=key&filterValue=value

Multiple filters

- /todos?filterKeys=key1%2Ckey2&filterValues=val1%2Cval2

%3D: Equals sign
%2C: Comma

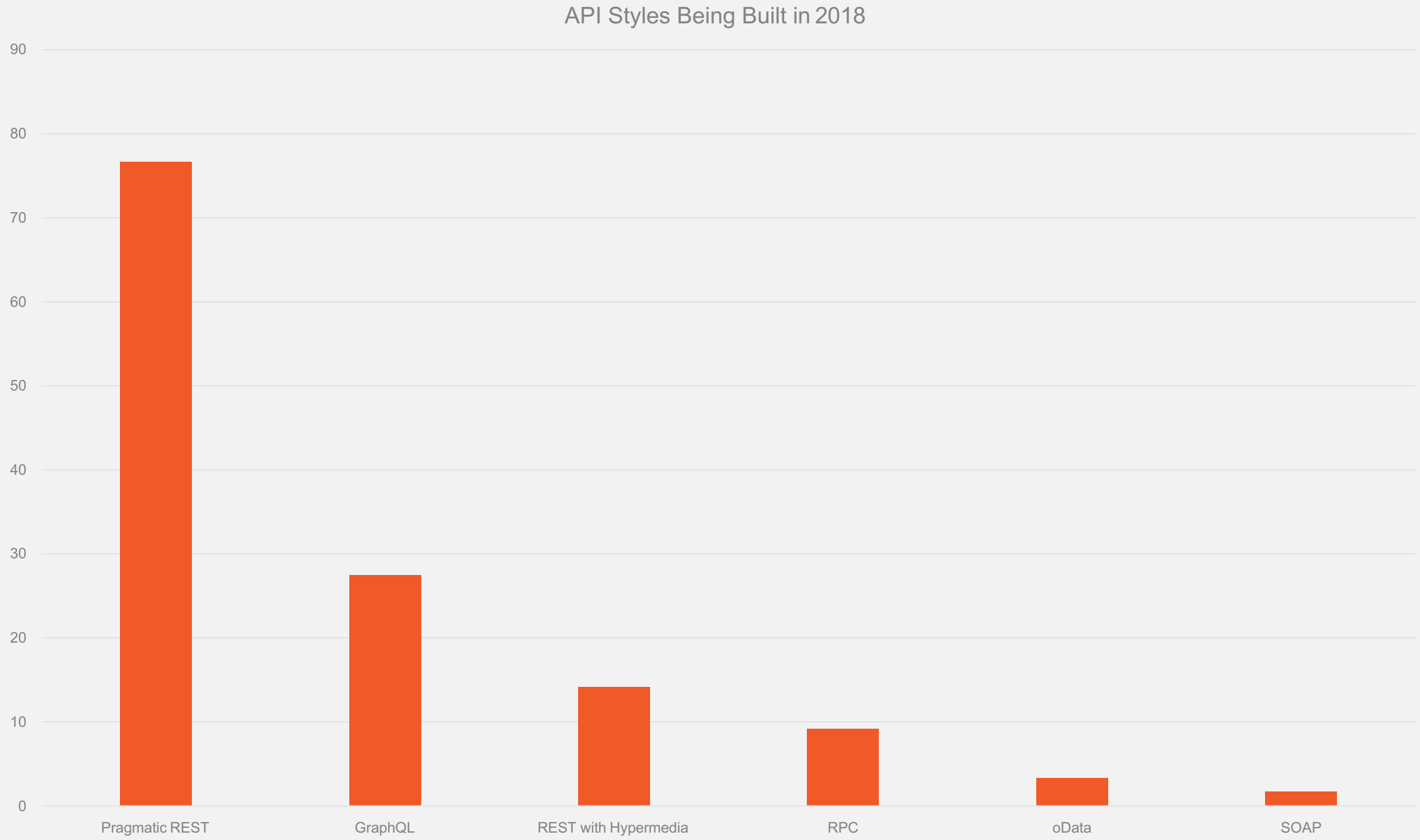
Pagination

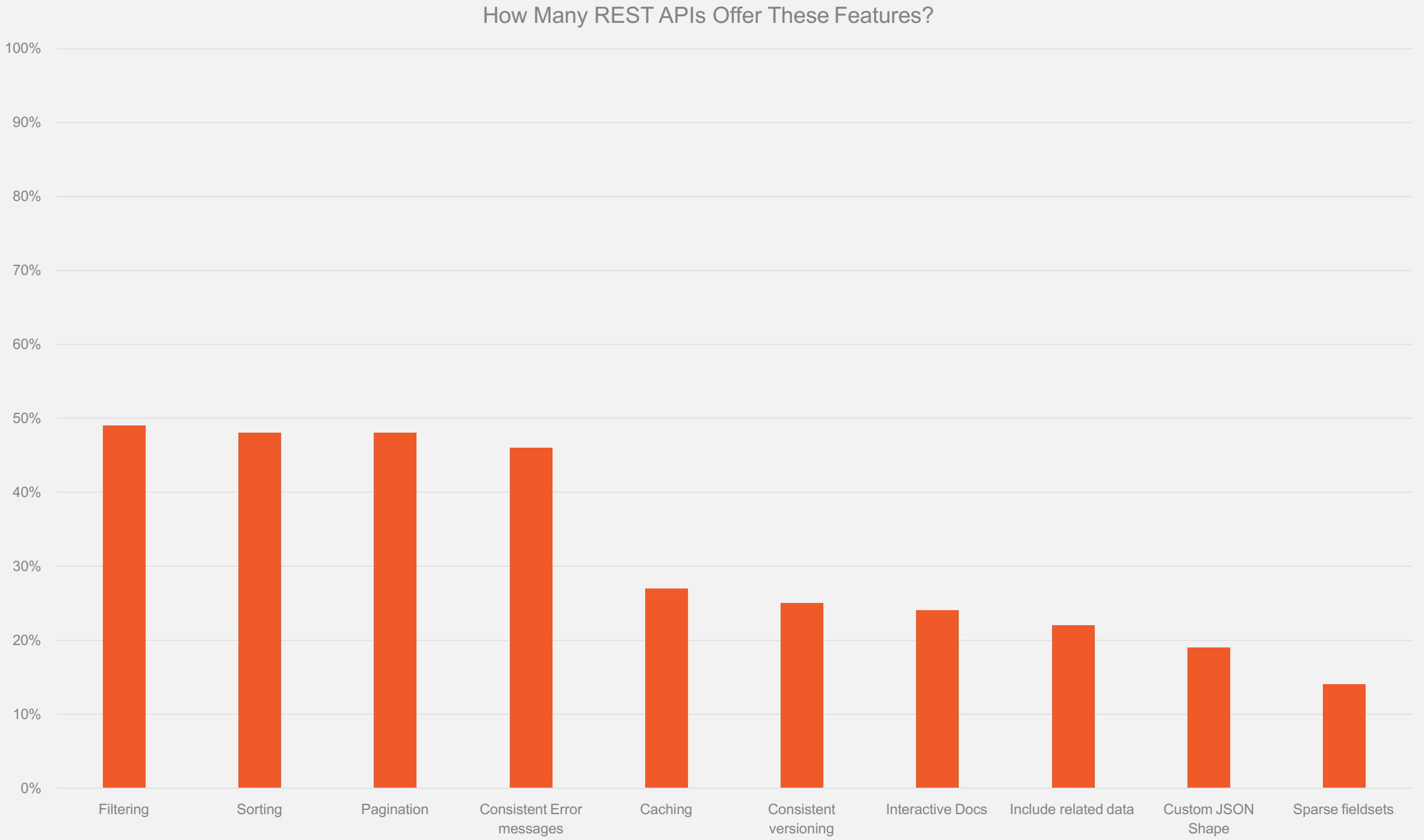
/todos?limit=10&offset=20

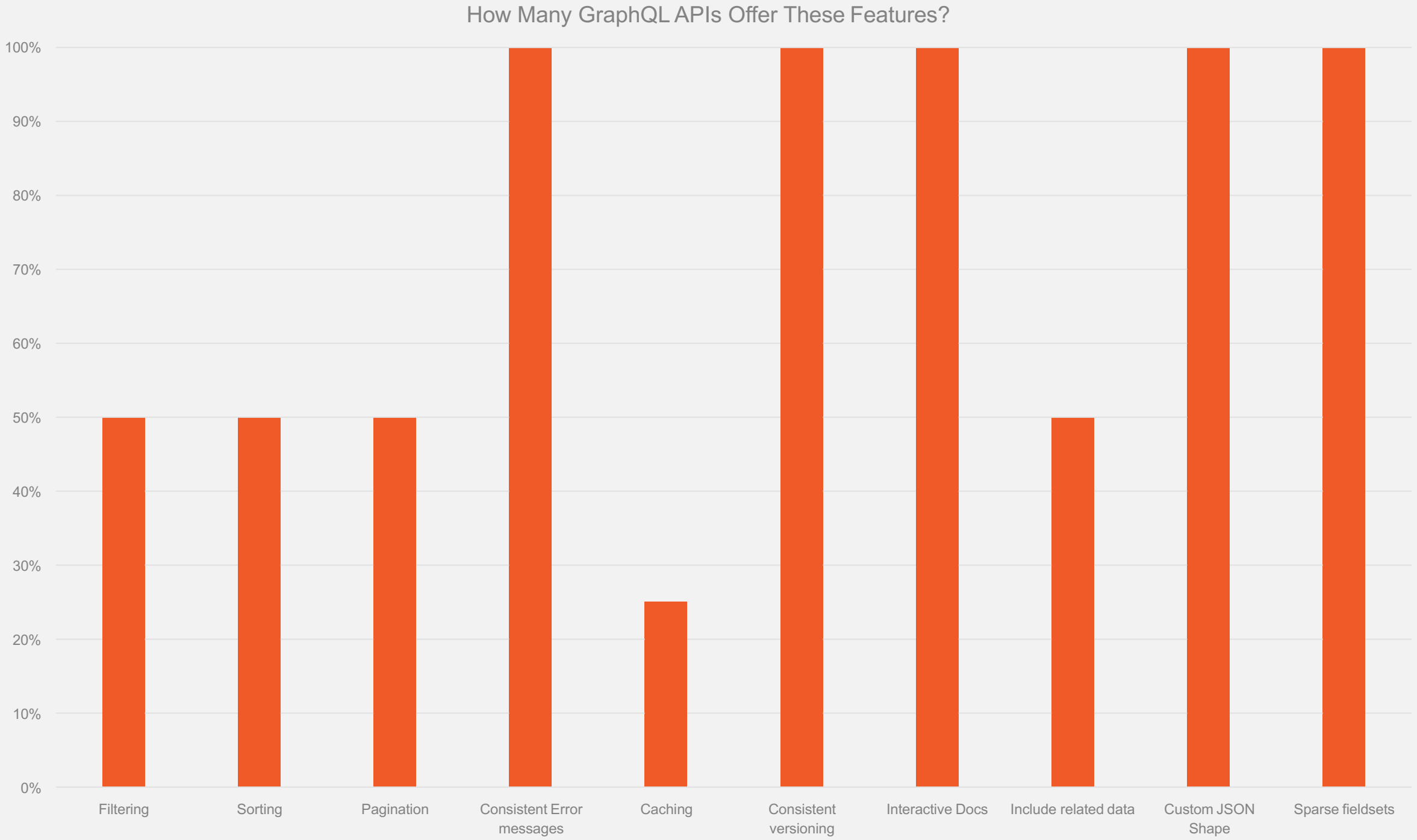
/todos?count=10&skip=20

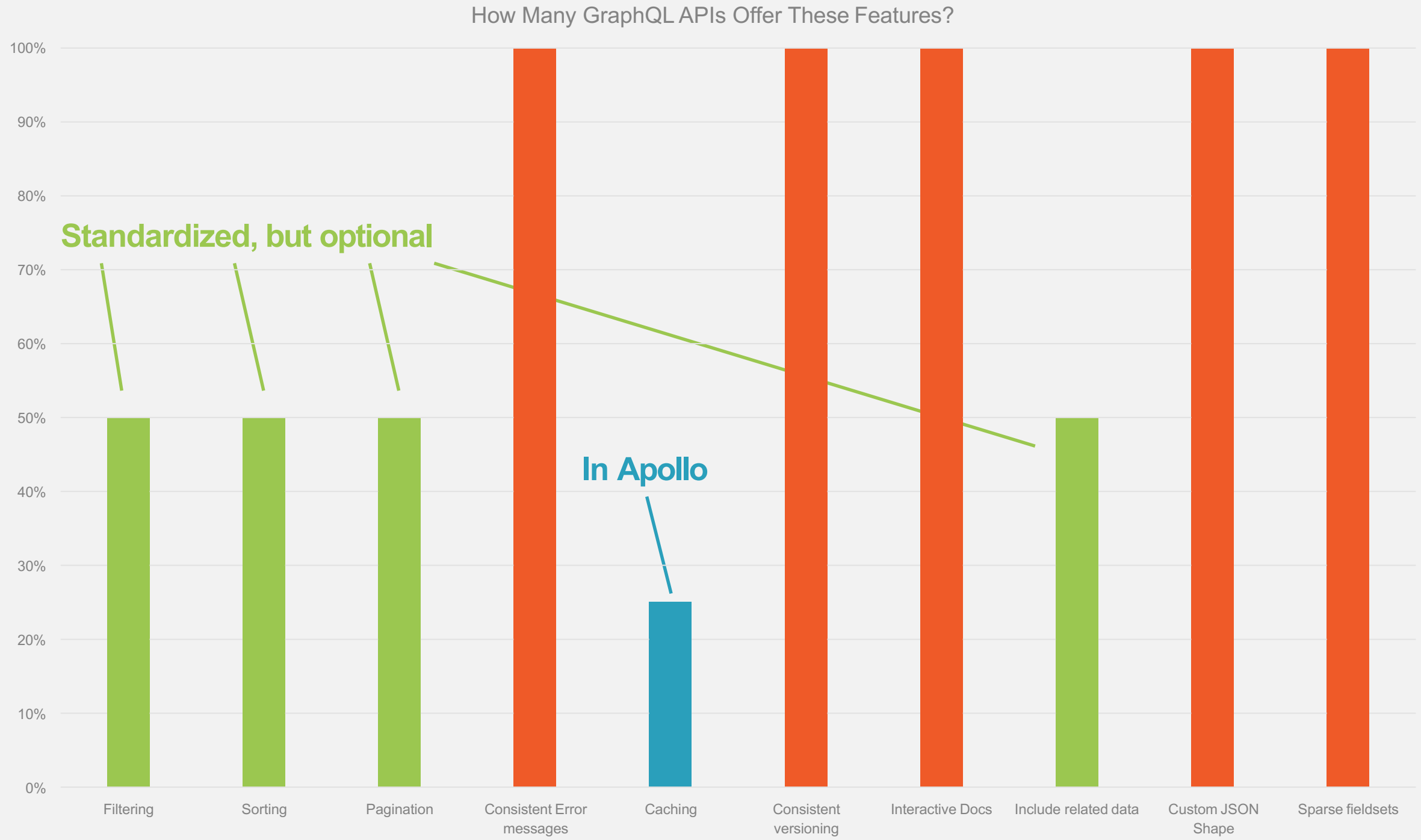
/todos?pg=1

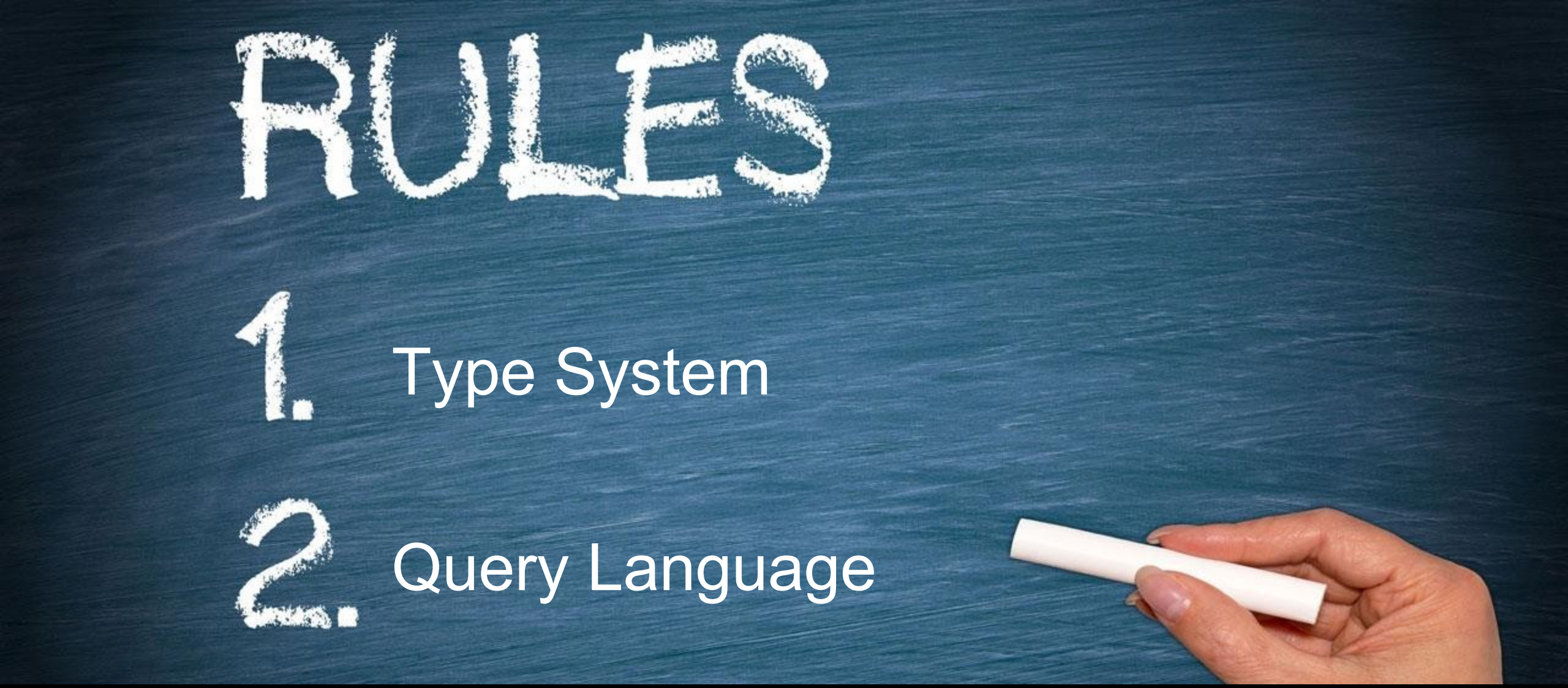
/todos?page=1


API Styles Being Built in 2018
90
80
70
60
50
40
30
20
10
0
Pragmatic REST
GraphQL
REST with Hypermedia
RPC
oData
SOAP


How Many REST APIs Offer These Features?
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Filtering
Sorting
Pagination
Consistent Error messages
Caching
Consistent versioning
Interactive Docs
Include related data
Custom JSON Shape
Sparse fieldsets


How Many GraphQL APIs Offer These Features?
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Filtering
Sorting
Pagination
Consistent Error messages
Caching
Consistent versioning
Interactive Docs
Include related data
Custom JSON Shape
Sparse fieldsets


How Many GraphQL APIs Offer These Features?
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Standardized, but optional
In Apollo
Filtering
Sorting
Pagination
Consistent Error messages
Caching
Consistent versioning
Interactive Docs
Include related data
Custom JSON Shape
Sparse fieldsets



GraphQL

Versioning

Standardized. Deprecate fields.

HTTP Verbs

Standardized.

Filters

Standardized.

Include child data

Standardized.

Errors

Standardized.

GraphQL removes the arguing or confusion about what is “the most RESTful way to do something”, as it has a spec and example implementations.

Phil Sturgeon

REST vs GraphQL: Summary

REST	GraphQL
Sparse fieldsets	Declare fields in query
Schema optional (JSON Schema, MSON),	Schema required
Various versioning approaches	Deprecate fields and monitor use
Include data via relations	Declare desired data in query
Easy caching (though sparse = harder)	Caching = few hits, more complex
Various URL approaches	Standardized query language
Swagger, but no doc standard	GraphiQL / GraphQL Playground

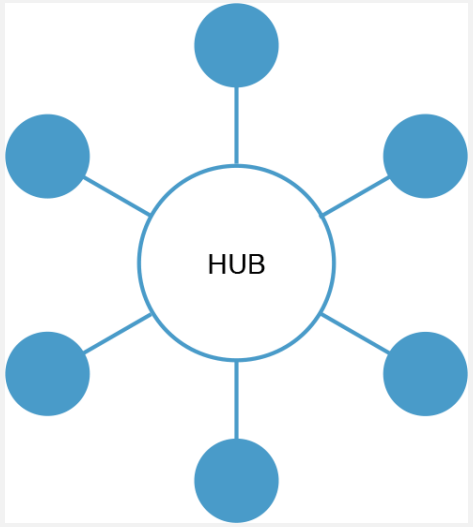
3

Tech Stack Agnostic

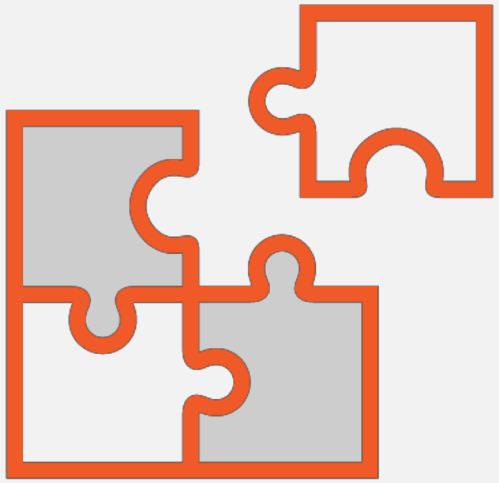
Over 24 tech stacks supported

4 One Endpoint

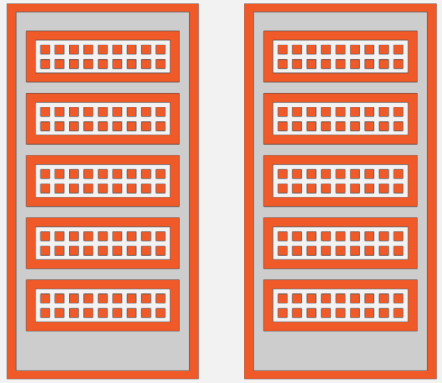
One Endpoint



No discovery issues



Stitch Schemas



Can still call separate APIs behind the scenes

No separate web and mobile URLs

5

Strongly Typed

Catch typos and errors ***before*** deployment

6 Incremental Adoption

Existing API?

- Replace one call at a time
- Wrap it with GraphQL

7

Generated Docs

GraphiQL, Playground…

Durability

GraphQL Field Deprecation

```
‘author_name’ => [

‘type’ => Type::string(),
   ‘deprecationReason’ =>     ‘Deprecated.    Use author    field’,
  ],
```

Analytics

Know what data is used

Deprecate what isn’t used

Detailed performance info

9

Community


Microsoft

ticketmaster®


Instagram


STARBUCKS
COFFEE


coursera
SERVERLESS
Product Hunt
yelp
METEOR


PayPal


intuit


ATLASSIAN

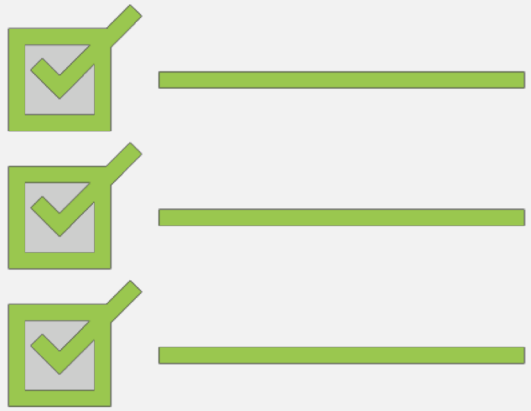
GraphQL Server

Server Steps

Parse

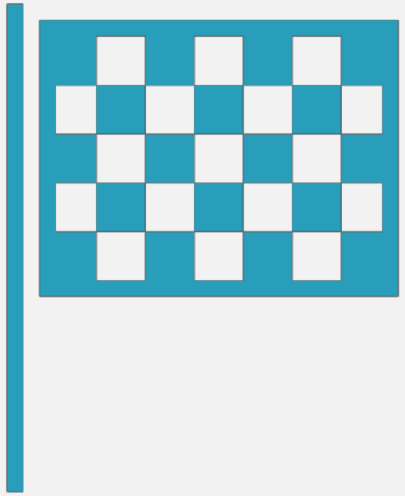
Convert to AST

Bad syntax? Stops.



Validate

Compare query to schema



Execute

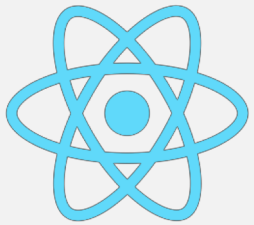
Run the query

No client-side lib needed.

Just copy/paste queries from GraphiQL and use fetch!

GraphQL Clients

GraphQL Client Libraries



React

Angular

Vue

iOS

Android

Relay

Apollo

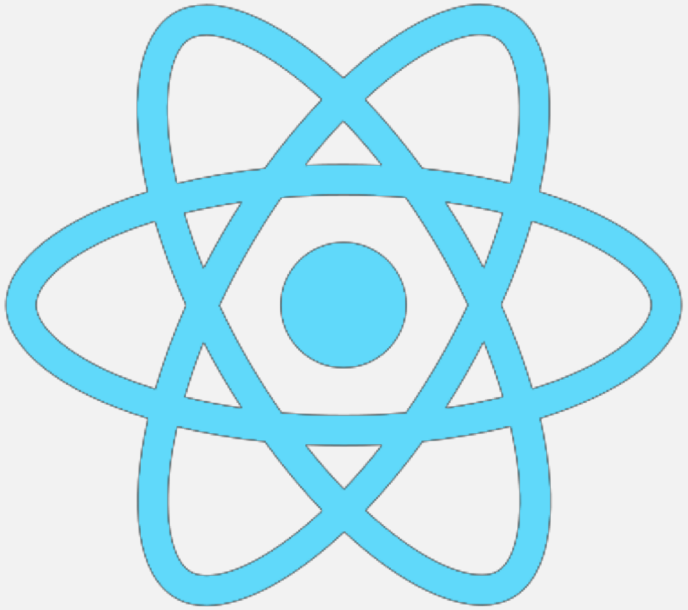
Urql

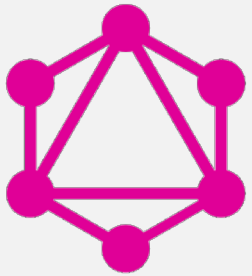
graphql-request

Popular, easy to learn, framework agnostic

Learn this…

Before this…





GraphQL

Learn this…

Before this.

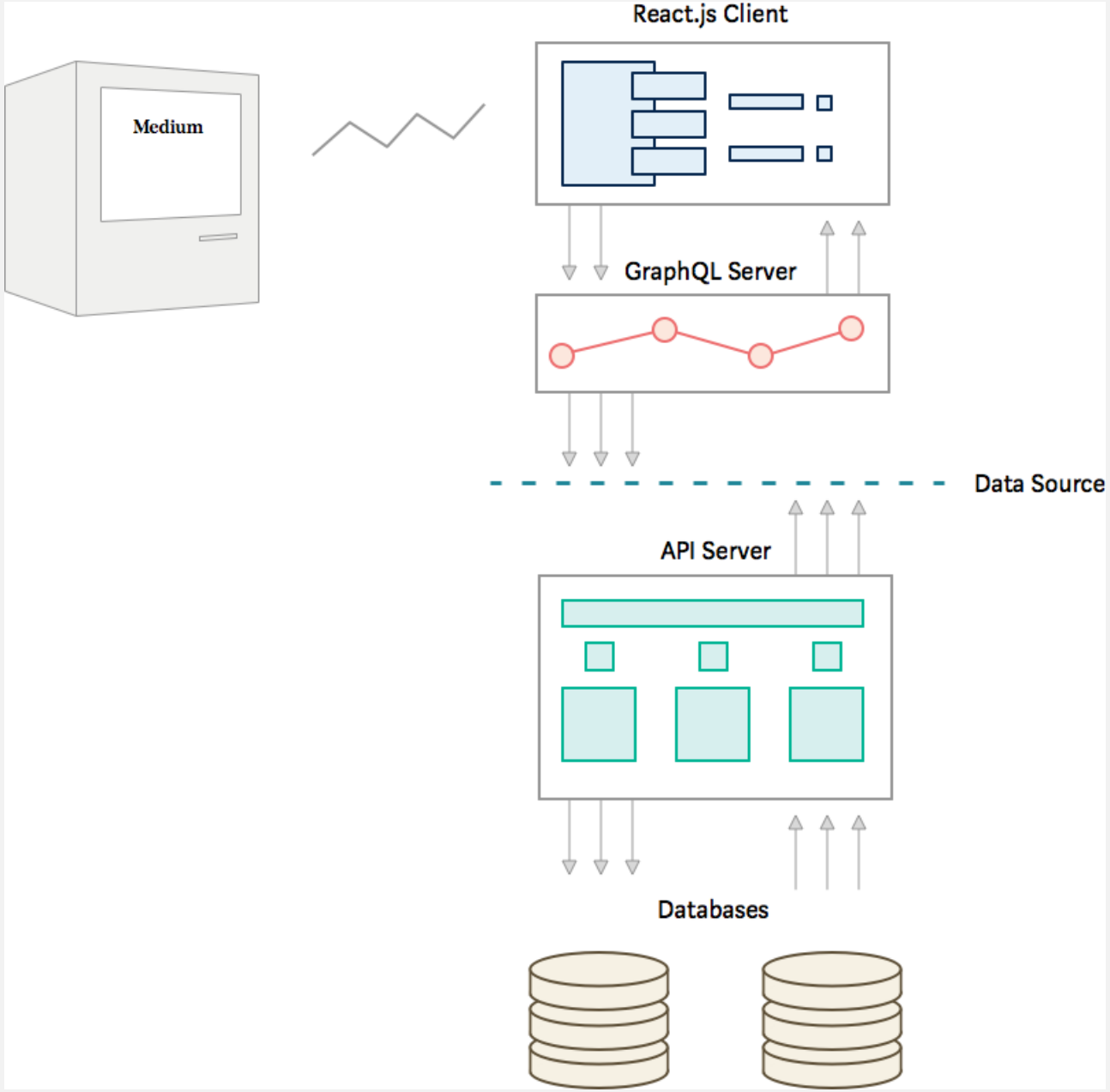
APOLLO

GraphQL Server Approaches

Connect to DB

Wrap Existing APIs

Mix these two


React.js Client
Medium
GraphQL Server
Data Source
API Server
Databases

GraphQL Downsides

GraphQL Downsides

Flexibility Costs

Expensive Query Risk

Caching

UI Coupling

Lazy Loading

Freedom Isn’t Free

Traditional APIs default to rigid

GraphQL defaults to flexible

Unfamiliar with the data? This ***may*** be easier.

Downside 2: Expensive Queries

```
query {
   users {
      name
      education {
         degree
         year
      }
      age
      address {
         country
         city
         street
      }
   }
}
```

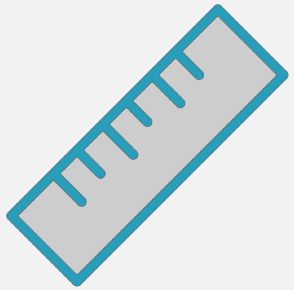
The N+1 Problem

May lead to many SQL queries
May be better off lazy loading

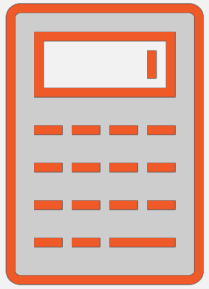
Handling Expensive Queries

Require field args

Timeout



Max depth



Complexity rating

Throttling

Downside 3: Caching

Since GraphQL POSTs against a single endpoint, you can't use traditional network caching tools.

Two Assumptions For GraphQL Caching

Same path = Same object

Same ID = Same object

Caching only works for the exact same query.

Different queries that return some of the same data = no cache hit.

This can lead to UI out-of-sync issues if GraphQL doesn’t realize these queries return the same author.


dataIdFromObject
GraphQL Result Data
Unique ID

Downside 5: Lazy Loading

GraphQL's power encourages needless eager loading.

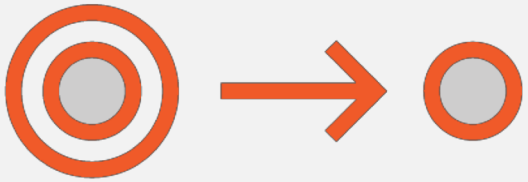
It's *slightly* easier to lazy load with HATEOAS.

Let’s Wrap Up

The GraphQL Rhythm

Declare schema

Describe data types



Declare resolver

Describe actions

Write queries

Use GraphiQL

You Don’t Need GraphQL If…

1. Allowing careful evolution
   (instead of global versioning)
2. Serializing data
   (instead of returning directory from the data store)
3. Implementing sparse fieldsets
   (to slim down responses)
4. Supporting include queries
   (to avoid round trips)
5. Gzip’ing your contents
   (to save bandwidth)
6. Documenting your data structures
   (e.g. use JSON Schema)

You don’t need GraphQL if you’re doing the hard stuff that few people do, and even fewer do well.

GraphQL vs REST:
Consider Using Both.

Wrap REST APIs for optional use

Use GraphQL as a single entry point

Solved!

Transformations

Under/over fetching

API mods to support new field

API design convos

Caching (Apollo / Relay)

Client-side proxy (Apollo / Relay)

Docs

Discovery (GraphiQL)

Test queries and mutations (GraphiQL)

Loading and error state