hashnode.com /post/why-graphql-is-the-future-of-apis-cjs1r2hhe000rn9s23v9bydoq

Why GraphQL is the Future of APIs - Hashnode

Leonardo Maldonadopublished a storyFeb 12, 2019

8-10 minutes

Since the beginning of the web, developing APIs has been a difficult task for developers. The way we develop our APIs must evolve with time so that we can always build good, intuitive and well-designed APIs.

In the last few years, GraphQL has been growing in popularity among developers, and a lot of companies have started adopting this technology to build their APIs. GraphQL is a query language developed by Facebook in 2012 and released publicly in 2015. It has been gaining a lot of traction and been adopted by a lot of big companies such as Spotify, Facebook, GitHub, NYTimes, Netflix, Walmart, and so on.

In this series of tutorials, we're going to examine GraphQL, understand what it is, and see what features make this query language so intuitive and easy to use.

So, let's get started by examining the problems with REST, and how GraphQL solves them. We will also find out why companies have been building their APIs with GraphQL, and why it is the future of APIs.

Oh, REST

A long time ago, when we changed our API design from SOAP to REST, we thought that move would give developers more flexibility in their work. We can't deny that it worked pretty well for some time and was a good move at the time. As the applications and the web have been getting more and more sophisticated, the APIs naturally evolved as well, following these changes.

However, REST does have a lot of problems. Let's see what they are:

A lot of endpoints

Each resource in REST is represented by an endpoint. So, in a real-world application we would end up having a lot of endpoints for a lot of resources. If you want to make a GET request, you would need an endpoint specific to that request, with specific parameters. If you want to make a POST request, you would need another endpoint just for that request.

But, what's the problem with that? Let's imagine we are building a huge social media application like Facebook. We will end up with a lot of endpoints which means more developer time is going to be spent

developing and maintaining these APIs.

Over-fetching and under-fetching of information

A real problem that annoys a lot of developers is over-fetching and under-fetching information through REST APIs. This is because REST APIs always return a fixed structure. We can't get exactly the data that we want unless we create a specific endpoint for that.

So, if we need only a tiny piece of data, we have to work with the whole object. For example, if we only need to get the `firstName`, `lastName`, and `age` of a user in a REST API, there's no way we can get exactly this data without fetching the whole object.

There's also a problem with under-fetching of information. If we want to get data from two different resources, we need to make different calls to two different endpoints. In a huge application, this doesn't scale so well since there will be cases where we only need to get a specific piece of data, not the entire object. Now, imagine we're building an application that's going to have 100 endpoints. Imagine the amount of work and code that we need to write. This will become more difficult with time. The code also gets hard to maintain and developers feel lost in the process.

Versioning

One of the painful points in REST, in my opinion, is versioning. With REST APIs, it is very common to see a lot of APIs with v1 or v2, but in GraphQL there's no need for it since you can evolve your APIs by adding new types or removing old ones.

In GraphQL, all you need to do to evolve your API is to write new code. You can write new types, queries, and mutations without the need to ship another version of your API.

So, you won't see GraphQL APIs with endpoints like the following:

```
https:
https:
```

Why GraphQL is the future

Back in 2012, Facebook faced a problem while they were developing their mobile applications which led them to create GraphQL. Those problems are very common, especially when we're talking about RESTful API design. As discussed these problems are:

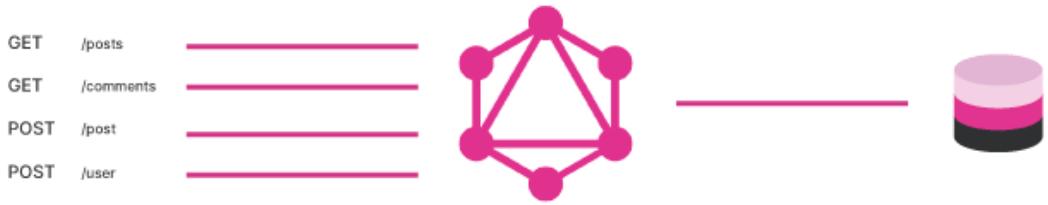
- Poor performance
- A lot of endpoints
- Over-fetching or under-fetching of data
- Shipping another version every time we need to include or remove something
- Difficulty understanding APIs

With a lot of concepts in mind, developers from Facebook developed a better way to design APIs, and later named it GraphQL. Basically, it's the replacement for REST, with a lot of improvements.

With GraphQL, we get a lot of new features that give you superpowers when you are building your APIs. Let's examine them one by one:

Single endpoint

There's no need to build a lot of endpoints! With GraphQL, we only get one endpoint, and with that we can get as much data as we want in a single request. Basically, GraphQL wraps all of your queries, mutations, and subscriptions in one endpoint and makes it available to you. It improves your development cycle because you don't need to make two requests to get data from two different resources. Also, imagine that we're building a huge application, we won't get a lot of endpoints and a ton of code as with REST. We will get a single endpoint, and with that endpoint we can make as many requests as we want.



Also, as I said above, an "endpoint-only" approach makes your API self-documented since there's no need for you to build documentation because your developers already know how to use it. They can understand the API just by looking at the code, or by looking at the playground. We're going to learn more about it later on (next tutorial in this series). Seems magical, but it is just GraphQL!

With GraphQL you fetch only the data you need

No more over-fetching or under-fetching of information. You fetch only the data that you need. Remember the poor performance issues that we discussed initially? We don't have that anymore since GraphQL improves the performance of your API, especially in case of slow network connection.

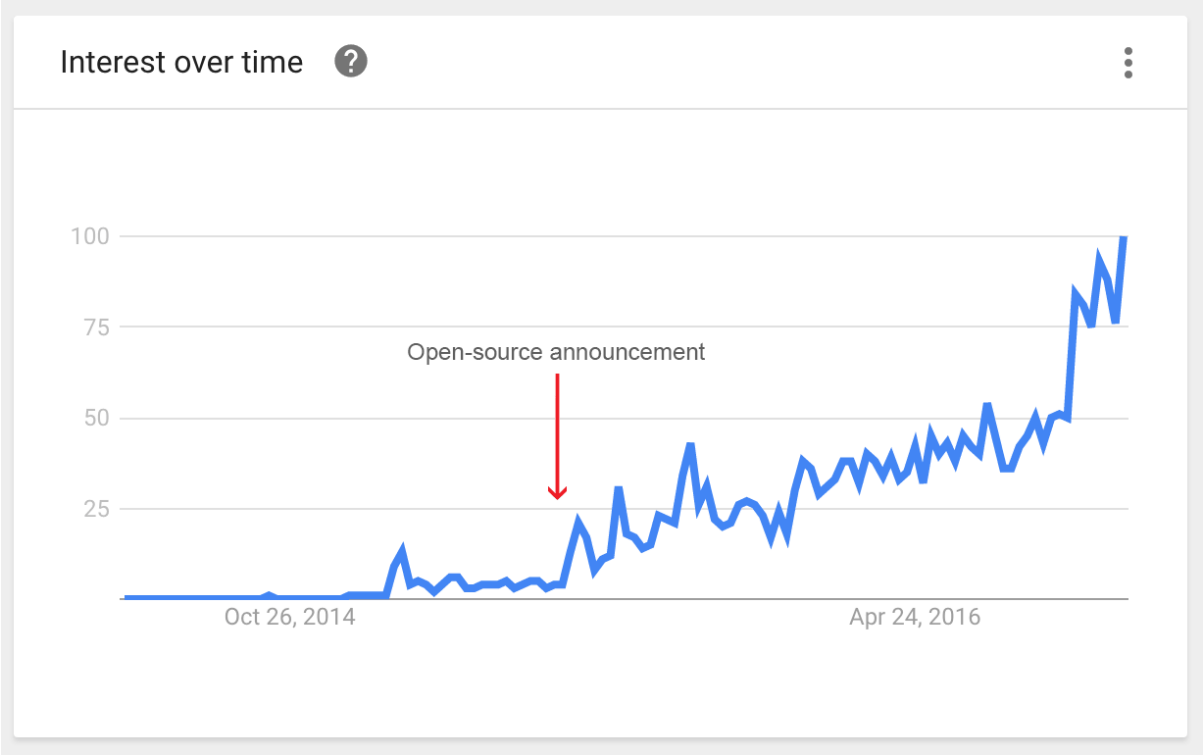
GraphQL makes it easy to start building APIs and be consistent

A lot of people think that GraphQL is pretty complicated because it involves a schema and a single endpoint. Once you start developing APIs with it, you discover that it's easier than you thought. An "endpoint-only" API helps a lot when you're developing your website/app. It makes your API more self-documented and there's no need for you to write a lot of documentation about it.

If you don't use JavaScript as your main language, that's not a problem since GraphQL is an agnostic query language which means you can use it with any language. At the time of writing this tutorial, GraphQL has support for more than 12 languages.

GraphQL Is The Future

The fact that GraphQL is an open source query language means that the community can contribute to it and make improvements to it. When Facebook released it to the community, it gained a lot of traction and approval from developers. Now, GraphQL has been growing rapidly as more and more developers are starting to build APIs with it. However, some people have been asking if this is really going to replace REST or become the new way to build APIs for real-world applications.



At first, I thought that GraphQL was only hype and just another way to create APIs. However, when I started to study it, I found out that GraphQL has the essential features that are required to create modern APIs for modern applications because it really plays well with modern stacks.

So if I could say something to you now is: yes, **GraphQL is really the future of APIs**. That's why big companies are betting on it.

In November 2018, GraphQL created a GraphQL Foundation, in partnership with the Linux Foundation, which means that this query language is encouraging its developers to build more documentation, tools, and support for the language. This foundation will ensure a stable, neutral, and sustainable future for GraphQL. So, this is another reason to consider GraphQL to be the future of APIs.

Of course, it won't replace REST immediately because many apps still use it and it's impossible to rewrite them overnight. As more and more companies adopt GraphQL, both UX and DX will improve.

Conclusion

GraphQL is really the future of APIs and we need to learn more about it. That's why I decided to create a series of 4 tutorials that will show how we can work with the best of GraphQL, starting from Queries and Mutations, then Subscriptions, and Authentication.

In the next tutorial of this series, I'm going to dive deep into GraphQL, show how GraphQL works with types, and create our first Queries and Mutations.

So, stay tuned and see you in the next tutorial!

Learn Something New Everyday,
Connect With The Best Developers!