Why Apollo: Advantages and Disadvantages

JULY 04, 2018 BY ROBIN WIERUCH - EDIT THIS POST

Follow on Twitter 17k Follow on Facebook

Interested in reading this tutorial as one of many chapters in my GraphQL book? Checkout the entire The Road to GraphQL book that teaches you to become a fullstack developer with JavaScript.

This tutorial is part 2 of 2 in this series.

Part 1: Why GraphQL: Advantages, Disadvantages & Alternatives

Finding the right solution for a given problem is not always simple, and web applications build with GraphQL are a good example of how changing times make for constantly evolving challenges. Moreover, evolving challenges create a scenario where the solutions must also evolve, so even the number of choices becomes a task. This article will decipher the pros and cons of one such solution: Apollo for GraphQL, with alternative solutions in case you decide against it.

GraphQL is only the query language that has a reference implementation in JavaScript, and Apollo builds its ecosystem on top to make GraphQL available for a wider audience. This

includes the client-side as well as the server-side, because they provide a large ecosystem of libraries for both. The libraries provide an intermediate layer too: Apollo Engine, which is a GraphQL gateway. Essentially there's a reason Apollo is one of the most popular choices for using GraphQL in JavaScript applications.

TABLE OF CONTENTS

APOLLO ADVANTAGES

The following topics show you some of the advantages of using Apollo, to provide a well-rounded pro and con list. Feel free to contact me if you think anything is missing from either list.

Apollo's Ecosystem

While GraphQL is in its early stages, the Apollo ecosystem offers solutions for many of its challenges. Beyond that we can see how much the ecosystem is growing, because the company announces an update for Apollo or another library that can be used with Apollo's tech stack at every other technology conference. Apollo isn't just covering GraphQL, though; they also have effort invested in REST interfaces for backward compatibility to RESTful architectures. This even takes GraphQL beyond the network layer and remote data, offering a state management solution for local data, too.

The Company and Community behind Apollo

The company behind Apollo is pouring lots of resources into its success. They are also active in open source, offering in-depth articles about their products, supported by an established presence at the conferences. In general, the GraphQL ecosystem seems to be in good shape for the future. The community behind GraphQL is growing, as more developers adopt it and use Apollo for client and server-side JavaScript applications.

Who is using Apollo?

Tech-savvy companies are taking advantage of Apollo already. Many were familiar with the popular Meteor framework before, but new and extremely popular companies like Airbnb and Twitch are using it. These are just a few of their stories:

- Airbnb [1] [2]
- Twitch
- The New York Times
- KLM
- Medium

Apollo's Documentation

While Apollo continues to evolve, the team and community behind it keeps the documentation up to date, and they have plenty of insight about how to build applications. In fact, they cover so many areas it can be overwhelming for beginners.

Apollo Libraries

Apollo offers plenty of libraries for implementing an effective GraphQL tech stack for JavaScript applications, and their libraries are open-sourced to be more manageable. For instance, Apollo Link provides an API for chaining different features into a GraphQL control flow. This makes it possible for automatic network retries or RESTful API endpoints instead of a GraphQL endpoints (the endpoints can be used together, too).

Apollo is also offering exchangeable libraries which can be seen in the Apollo Client Cache. The Apollo Client itself is not biased toward its cache, where the data is stored, as any cache advertised by Apollo or its community works. There are already caches available that can be used to setup a Apollo Client instance.

Apollo's Features

Apollo comes with built-in features to pull all the complexity out of applications and handle the intersection between client and server applications. For instance, Apollo Client caches requests, which are not made twice when the result is already in the cache. The function provides a performance boost for applications, saving valuable network traffic. Also, Apollo Client normalizes data, so nested data from a GraphQL query is stored in a normalized data structure in the Apollo Client Cache. Data can be read from the Apollo Client Cache by an identifier, without looking up an "article" entity in an "author" entity. Beyond caching and normalization, Apollo Client comes with many more features like error management, support for pagination and optimistic UI, prefetching of data, and connection of the data layer (Apollo Client) to the view layer (e.g. React).

Interoperability with other Frameworks

One of Apollo's libraries makes it possible to connect Apollo Client to React. Just like libraries like Redux and MobX, the React-Apollo library has higher-order and render prop components to connect both worlds. However, there are other libraries out there that bridge not only Apollo Client to React, but also Apollo to Angular or Apollo to Vue. That's what makes Apollo Client view layer agnostic, which is great for the growing JavaScript ecosystem.

Apollo is also library agnostic on the server-side, and it offers several solutions to connect with Node.js libraries. Apollo Server for Express.js is one of the most popular choices among developers and companies, and there are other solutions for Koa and Hapi on Node.js for Apollo Server as well.

Modern Data Handling with Apollo

Remember back when we had to trigger data fetching in a component's lifecycle methods imperatively? Apollo Client solves this, because its data queries are declarative. React often employs a higher-order component or render prop to trigger a query automatically when a component renders. The GraphQL mutations are triggered imperatively, but that's only because a higher-order component or render prop grants access to the function which executes the mutation (e.g. on a button click). Essentially, Apollo embraces declarative programming over imperative programming.

Modern State Management with GraphQL and Apollo

With the rise of GraphQL in JavaScript applications, state management entered another state of confusion. Even though lots of pain points are eliminated using a GraphQL library like Apollo Client, since it takes care of state management for remote data, some developers are confused about where to put state management libraries like Redux or MobX now. However, it can be made simple using these libraries for local data only and leaving the remote data to Apollo. There is no longer a need to fetch data with asynchronous actions in Redux, so it becomes a predictable state container for all the remaining application state (e.g. local data/view data/UI data). In fact, the remaining application state may be simple enough to be managed by React's local state instead of Redux.

Meanwhile, Apollo has already released their own solution to manage local state--which is supposed to be managed by React's local state, Redux or MobX--by embracing GraphQL for everything. The Apollo Link State library lets us manage local data with GraphQL operations, except on the client-side in Apollo Client. It's Apollo saying: "You don't need any other state management library, we take care of your data." These are exciting times for developing JavaScript applications.

Convenient Development Experience

Using Apollo for JavaScript applications is becoming easier every day. The community is pushing out tools for implementation. There are development tools available as browser extensions, third-party tools to perform GraphQL operations such as GraphiQL, and libraries to simplify developing Apollo applications. For instance, the Apollo Boost library provides an almost zero-configuration Apollo Client setup to get started with GraphQL for client-side applications. Apollo takes away all the boilerplate implementation that comes with the GraphQL reference implementation in JavaScript.

APOLLO DISADVANTAGES

The following topics show you some of the disadvantages of using Apollo, to provide a well-rounded pro and con list. Feel free to contact me if you think anything is missing from either list.

Bleeding Edge

GraphQL is in its early stages. Apollo users and all early GraphQL adopters are working with brand new technology. The Apollo team is developing a rich ecosystem around GraphQL, providing the basics as well as advanced features like caching and monitoring. This comes with pitfalls, however, mainly because everything isn't set in stone. There are sporadic changes that can pose challenges when you are updating GraphQL-related libraries. In contrast, some libraries in GraphQL might be more conservative than the Apollo team, but the features usually aren't as powerful.

The ability for developers to continue learning is also hindered by fast-pace development. Tutorials for GraphQL and Apollo are sometimes outdated, and finding an answer may require external resources. The same is true for most new technology, though.

Under Construction

The Apollo team and community implements many new features in a rapid pace, but going so fast comes with a price. Searching for solutions often leads to GitHub, because there is little other information on the subject. While you may indeed find a GitHub issue for your problem, there is often no solution for it.

Rapid development also comes with the price of neglecting obsolete earlier versions. In my experience, people seemed confused when Apollo abandoned Redux as their internal state management solution. Apollo isn't opinionated about how Redux should be used side by side with it, but since it has been abandoned as internal state management solution, many people didn't know how to proceed when Apollo 2.0 was released. I think the team behind Apollo might be struggling to keep up with the fast-paced GraphQL ecosystem, and it's not always easy to heed all voices in open source development.

It is Bold and Fashionable

Apollo is bold, because it is moving beyond being a network layer ecosystem between client and server for GraphQL in JavaScript, but positioning itself as the data management solution of tomorrow. It connects client and backend applications with GraphQL, apollo-link-rest for RESTful APIs, and apollo-link-state for local state management. Some experts are skeptical about the "GraphQL everything" mentality, but time will tell if it corners that market.

Apollo is fashionable, because it keeps up with the latest trends. In React, the latest trend was render prop components. Because of this, and arguably the benefits of render prop components over higher-order components, the React Apollo library introduced render prop components next to higher-order components. It was a smart move to offer multiple solutions since both higher-order and render prop components come with their own sets of pros and cons. However, Apollo does advocate for render props over higher-order components, and it's not clear if this was hype-driven development or marketing or if they truly believe that this is the way of the future. Render props are relatively new in React, so it will take time for developers to realize they come with their own pitfalls (see: higher-order components). I have seen React applications become too verbose by using multiple render prop components in one React component, even though one render prop didn't depend on another render prop, rather than having those co-located to the React component by using higher-order components. After all, Apollo offers both solutions, render props and higher-order components, so the developer decides on a case by case basis for their applications. It's a good sign for users that the Apollo team is keeping up with the recent trends from other libraries, and not confining themselves to a bubble.

Missing Competition

Most of these concerns are about the newness of GraphQL, concerns that could be applied to virtually any other open source solution in the same field. One major concern, though, is the missing competition in the GraphQL in JavaScript domain. A couple of alternatives to Apollo are listed in the next section, but they are limited compared to the Apollo ecosystem. While it is possible to write your own library for GraphQL (e.g. a simple GraphQL in React client), not many developers have attempted it yet. Some problems solved by Apollo are not trivial, but I think competition would be a healthy push for GraphQL in JavaScript ecosystem. There is huge potential in GraphQL now, and open source developers would be wise to take advantage.

APOLLO ALTERNATIVES FOR JAVASCRIPT, REACT AND NODE.JS

Some disadvantages stem from using GraphQL as an alternative to a RESTful-driven architecture. There are some alternatives for Apollo Client and Apollo Server that can consume GraphQL APIs in JavaScript. The following list should grant insights about solutions in the JavaScript ecosystem, used for React on the client-side and Node.js on the server-side.

Apollo Client Alternatives for React

When it comes to Apollo Client for React, Angular, Vue, or similar applications, there are several alternatives to check out. Like Apollo, these come with their own advantages and disadvantages.

- plain HTTP request: Even though sophisticated GraphQL libraries can be used to perform your GraphQL operations, GraphQL itself isn't opinionated about the network layer. So it is your GraphQL operations, GraphQL itself isn't opinionated about the network layer. So it is

possible for you to use GraphQL with plain HTTP methods using only one endpoint with an opinionated payload structure for GraphQL queries and mutations.

- Relay: Relay is Facebook's library for consuming GraphQL on the client-side in React applications. It was among the first GraphQL client libraries before Apollo emerged.
- urql: urql is a GraphQL client library from Formidable Labs for consuming GraphQL in React applications. It was open-sourced as minimalistic alternative to the growing Apollo behemoth.
- graphql.js: graphql.js shouldn't be mistaken for the GraphQL reference implementation. It's a simple GraphQL client for applications without powerful libraries such as Vue, React, or Angular.
- AWS Amplify - GraphQL Client: The AWS Amplify family offers libraries for cloud-enabled applications. One of the modules is a GraphQL client used for general GraphQL servers or AWS AppSync APIs.

Apollo Server Alternatives for Node.js

When it comes to Apollo Server for Node.js with Express, Koa, Hapi or something else, there are several alternatives you can check out. Obviously these come with their own advantages and disadvantages whereas these things are not covered here.

- express-graphql: The library provides a lower-level API to connect GraphQL layers to Express middleware. It takes the pure GraphQL.js reference implementation for defining GraphQL schemas, where Apollo Server simplifies it for developers.
- graphql-yoga: A fully-featured GraphQL Server with focus on easy setup, performance & great developer experience. It builds on top of other GraphQL libraries to take away even more boilerplate code from you.

. . .

There are many reasons to use Apollo and its striving ecosystem for JavaScript applications, when you want to use a GraphQL interface over a RESTful interface. Their libraries are framework agnostic, so they can be used with a wide variety of frameworks on the client-side like React, Angular, Vue, and server-side applications like Express, Koa, Hapi.

Continue Reading: React with Apollo and GraphQL Tutorial

Continue Reading: A complete Apollo Server with GraphQL and Express Tutorial

Show Comments

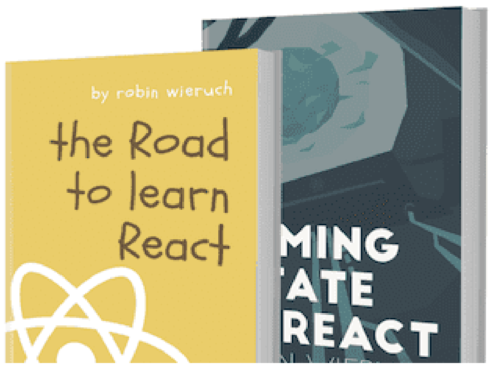
KEEP READING ABOUT FIREBASE

GRAPHQL RESOLVER MIDDLEWARE

GraphQL resolvers are used to resolve GraphQL queries to actual data. In this GraphQL tutorial, you will learn how to set up a GraphQL middleware for these resolvers for dealing with authorization and...

HOW TO BUILD A GRAPHQL CLIENT LIBRARY FOR REACT

You may have used a GraphQL client library that was view-layer agnostic and thus able to work with React or other solutions like Angular or Vue. Other GraphQL client libraries like Relay and Urql aren...



TAKE PART

NEVER MISS AN ARTICLE ABOUT WEB DEVELOPMENT AND JAVASCRIPT.

- ✓ Join 50.000+ Developers
- ✓ Learn Web Development
- ✓ Learn JavaScript
- ✓ Access Tutorials, eBooks and Courses
- ✓ Personal Development as a Software Engineer

SUBSCRIBE

View our Privacy Policy

View our Privacy Policy.