Vue.js – Beyond REST to GraphQL

How we build Vue.js SPAs against GraphQL

Bloomberg Law®

Who We Are

Priya Kathiri

Ronald Rogers

Tracey Holinka

Bloomberg Law®

Bloomberg BNA

We’re a subsidiary of Bloomberg LP.

Bloomberg Law Bloomberg Government
Bloomberg Tax Bloomberg Environment

We’re trying to do what Bloomberg LP did in the 80’s to finance industry by using data and analytics for non-financial professions such as legal, tax, and labor.

Bloomberg Law®

Web App Developers

We’re looking for Developers who are:

Good problem solvers
Have a full stack understanding of Web Apps

Some of the things we’re using to build Web Apps:

JavaScript/ Vue.js	GraphQL	Java
Ruby on Rails	Python	

Bloomberg Law

Why GraphQL and not REST?

GrahpQL is a “graph” — you can grab the entire database in a single request. With standard REST API you can only grab parts of the database. Less calls to the database equals better performance.

GraphQL is easy to start serving—you write a thin layer over your already existing data schema.

Bloomberg Law®

Why GraphQL and not REST?

GraphQL is widely adopted throughout the technical community.

There are tools available that make it easy to integrate with GraphQL like Apollo client.

There are developer tools like GraphiQL and Apollo Dev Tools.

Bloomberg Law

GraphQL Overview

GraphQL is a single endpoint (as opposed to REST).

GraphQL is broken up into **Queries** and **Mutations**.

Queries are used to get data from the graph.

Mutations are used to change data in the graph.

Bloomberg Law

GraphQL Server Libraries

There are GraphQL Server libraries for:

JavaScript Python Ruby PHP
Java .NET and others

GraphQL: http://graphql.org/

Bloomberg Law

GraphQL is Easy to Setup

A major benefit of GraphQL is how easy it is to write an API.

You just define your types, define your queries, and you're ready to go.

It is also easy to query GraphQL once you have an endpoint up and running.

Bloomberg Law®

Live Coding

Bloomberg Law®

Apollo GraphQL Client

Apollo makes it easy to build UI components that fetch data with GraphQL and it makes front-end state management easy.

There are Apollo clients for:

Vue.js	React	Angular	Ember
Meteor	Android	iOS	

Apollo: https://www.apollographql.com/

Bloomberg Law

Apollo Makes State Management Easy

Apollo queries the backend and caches it locally. As you make changes to the state, by changing query variables or mutations, it automatically updates the local state.

If you are using Vuex for state management, you would need to create all the actions, mutations, etc. to update the state yourself.

Apollo figures this all out for you.

Bloomberg Law

Apollo Manages Mutations

Without Apollo when you make a change on the backend, you had to manually refetch the data to update your UI.

With Apollo, when you call a mutation, the data that gets returned from the mutation automatically updates your UI.

Bloomberg Law

Live Coding

Bloomberg Law®

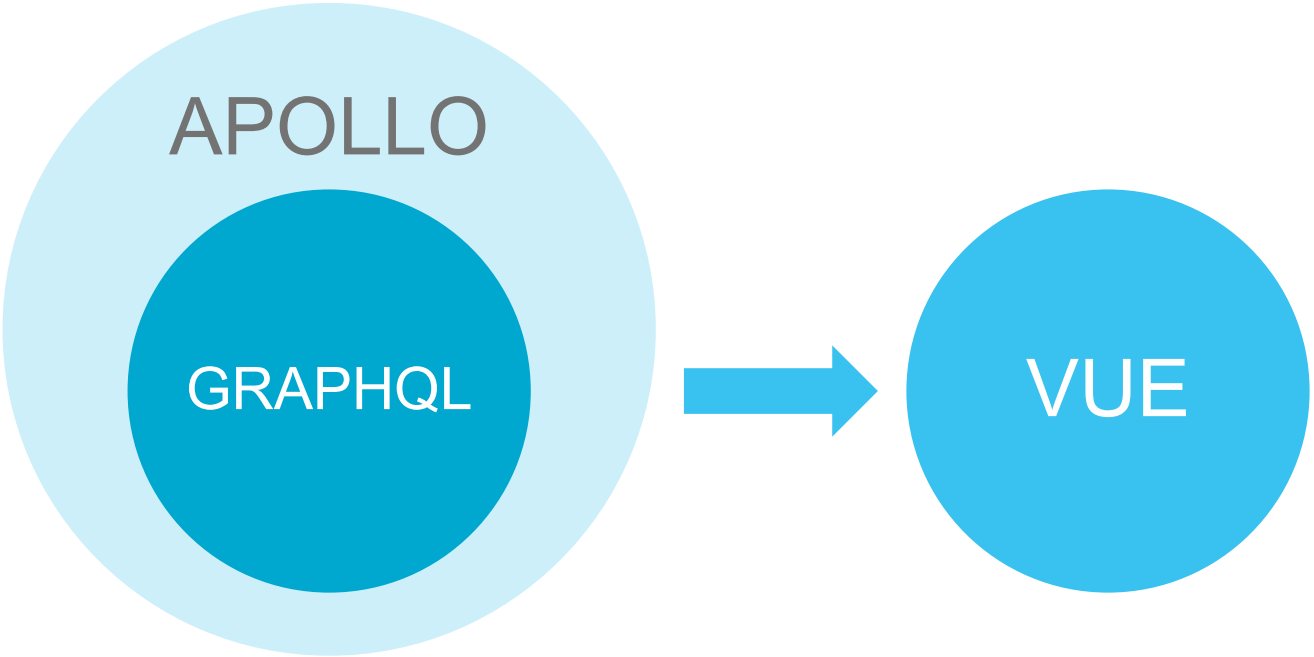
Vue.js & Apollo

Vue.js automates keeping the UI in sync with the local state. As the local state is updated, Vue.js automatically updates the UI.

The job of Vue.js is to automate updates to the UI.
The job of Apollo is to automate updates to the state.

Bloomberg Law

Bloomberg Law®



Bloomberg Law®

Apollo Watches Variables

When one of your variables changes Apollo automatically re-queries GraphQL and updates the state. Then Vue.js updates the UI.

If it has fetched the data recently already, it just grabs it from the local cache.

Bloomberg Law

Live Coding

Bloomberg Law®

External Data Access With GraphQL

A data resolver can get its data from anywhere, either a local database or other systems in your organization.

This enables you to quickly serve resources from anywhere in your organization with a GraphQL server acting as single data source.

Bloomberg Law

Real Time Updates with GraphQL

Through subscriptions whenever the backend changes the frontend UI automatically gets updated without any end-user interaction.

Apollo can subscribe to a query.

Whenever the query returns new data, it is automatically adds to the local state, and Vue automatically updates the UI.

Bloomberg Law

Real Time Updates with GraphQL

With GraphQL subscriptions you can easily make your sites appear ”living”.

This is how Facebook, the creators of GraphQL, update users’ timelines.

Bloomberg Law

State Management is not Fun

As we developed our app we quickly came to the realization how complex state management is.

It is the most difficult aspect of any JavaScript application.

GraphQL and Apollo—state management without pain.

Bloomberg Law

GraphQL + Apollo + Vue.js = Awesome

Bloomberg Law®

Questions

Bloomberg Law®

Addendum

To correct some of the errors we made in the presentation and to expound upon some of the questions we received at the end of the presentation we've decided to include this addendum.

GraphQL subscriptions

We're not entirely sure about this as we haven't yet implemented subscriptions. GraphQL is just a spec, it likely depends pretty heavily on the language and server implementation you are using. If your server implementation supports subscriptions the transport will most likely be web-sockets.
Also they will likely be subscriptions to domain events and not just normal queries.

https://github.com/facebook/graphql/blob/master/rfcs/Subscriptions.md

Bloomberg Law

Addendum

Gotchas we ran into when implementing Apollo

1. You will want to use batch querying if you desire to use in-component queries (like in the demo).
https://www.apollographql.com/docs/link/links/batch-http.html

Your GraphQL server implementation must also support query batching, otherwise each query will result in its own network request.

2. If you use union or interface types you need to have a process that gets that part of the schema from your API and hands it off to Apollo's fragment matcher.
https://www.apollographql.com/docs/react/recipes/fragment-matching.html

Bloomberg Law®

Addendum

3. You need to come up with your own way to return errors to the user, the GraphQL spec will not do it for you.

In our mutations we return types with the signature:

```
type ExampleMutationResult {
 [some result],
 errors: [Error]
}
```

The `setAuthorName` mutation in the demo would probably actually return something like:

```
type AuthorMutationResult {
  author: Author,
  errors: [Error]
}
```

Bloomberg Law

Addendum

4. Variables not in queries that otherwise might change a query's result will not cause that query to get automatically refetched.

A common example is the user token passed in via a header - queries do not react to presence of header values, so if that changes, the query will remain the same.

If a user logs out, your UI will not refetch all user dependent queries and present a logged out view. You will have refetch those queries yourself, or just tell Apollo to refetch everything: `` `this.$apollo.getClient().resetStore()` ``

You might be able to use the viewer pattern to help with this, but Facebook has said this is an anti-pattern: https://github.com/lucasbento/graphql-pokemon/issues/1

Bloomberg Law

Addendum

How to handle authorization

Just put the user in the GraphQL context and reference that in your resolvers.

You might be tempted to use the viewer pattern as we were, but Facebook has said this is an anti-pattern: https://github.com/lucasbento/graphql-pokemon/issues/1).

Corrections

Oops. Turns out Facebook does not actually use GraphQL subscriptions to update the timeline in Facebook, they use their own event-based system.

Bloomberg Law