How to Use Suspense and GraphQL with Apollo to Build Great User Experiences✨

Jerel Miller and Alessia Bellisario,
Apollo Client

RVAJS Conference 2023

go.apollo.dev/c/suspense-talk

Jerel Miller / @jerelmiller

Principal Software Engineer

Alessia Bellisario / @alessbell

Staff Software Engineer

What this talk *is* about

- (Re-)introduction to Suspense concepts…
- for use in client-rendered web apps…
- using Apollo Client’s Suspense hooks released in 3.8 🎉

What this talk *is not* about

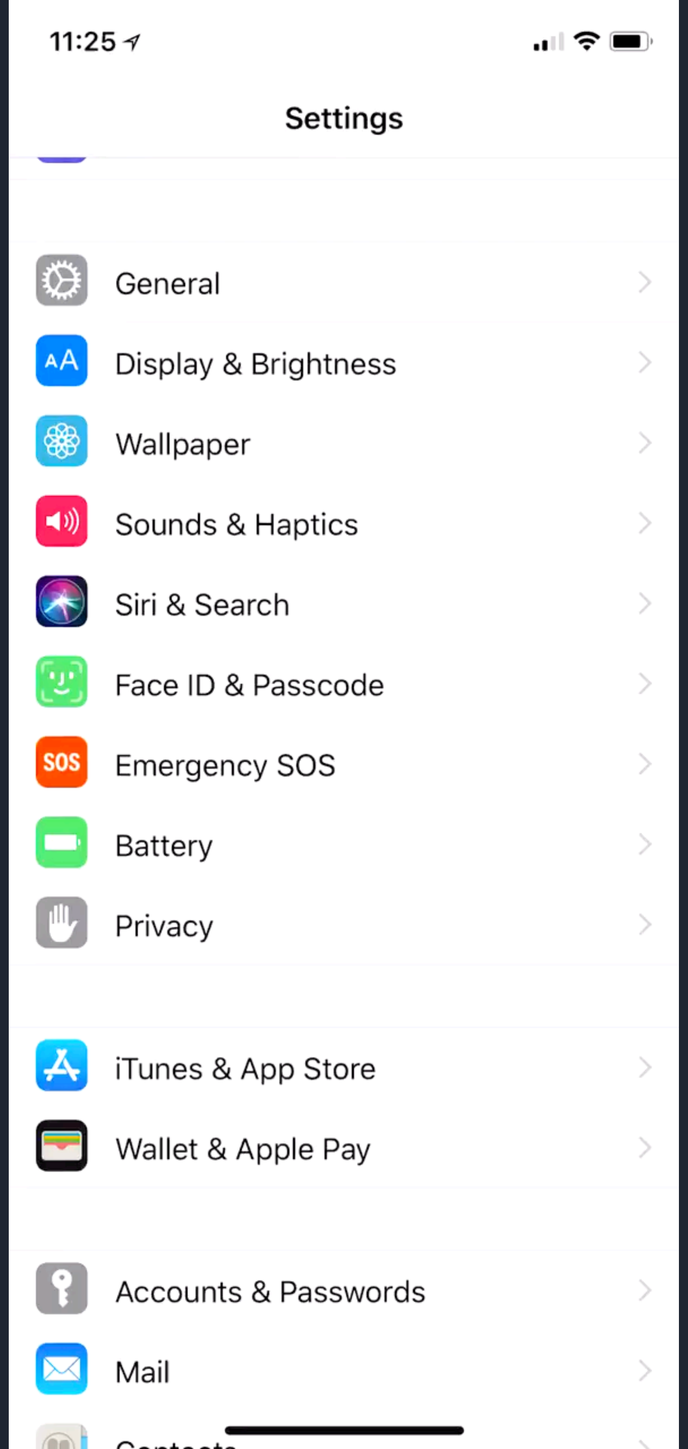
- React Server Components (RSC) and/or Streaming SSR
  - See our experimental package, @apollo/experimental-nextjs-app-support
- Implementing Suspense support in a data fetching library 😅

Andrew Clark
@acdlite · **Follow**

Replying to @acdlite

One reason iOS feels so much nicer than the web: fewer unnecessary loading states. Look what happens when you tap an option in the Settings app: rather than transition immediately and show a spinner, it pauses for a moment until the view is ready.

2:31 PM · Jan 20, 2018



The status quo:

The status quo:

`useQuery`

Albums.js

```
import { useQuery, gql } from "@apollo/client";

const GET_ALBUMS_QUERY = gql`
  query GetAlbums {
    albums {
      id
      year
      title
    }
  }
`;

export default function Albums() {
  const { data, loading } = useQuery(GET_ALBUMS_QUERY);

  if (loading) return <Loading />;

  return (
    <ul>
      {data.albums.map((album) => (
        <li key={album.id}>
          {album.title} ({album.year})
        </li>
      ))}
    </ul>
  );
}

function Loading() {…
```

Browser Tests Terminal

https://zqx5rr.csb.app/

Open The Beatles artist page

Console 0 Problems 0 React DevTools 0

ArtistPage.js × Albums.js

```
import { Suspense } from "react";
import Albums from "./Albums.js";

export default function ArtistPage({ artist }) {
  return (
    <>
      <h1>{artist.name}</h1>
      <Suspense fallback={<Loading />}>
        <Albums artistId={artist.id} />
      </Suspense>
    </>
  );
}

function Loading() {
  return <h2>🌀 Loading...</h2>;
}
```

Browser Tests Terminal

https://x6x8hw.csb.app/

Open The Beatles artist page

Console 0 Problems 1 React DevTools 0

Personal Free

sparkling-butterfly-s9zlw3

0 Embed Fork Create

Albums.js

```javascript
import { fetchData } from './data.js';

// Note: this component is written using an experimental API
// that's not yet available in stable versions of React.

// For a realistic example you can follow today, try a framework
// that's integrated with Suspense, like Relay or Next.js.

export default function Albums({ artistId }) {
  const albums = use(fetchData(`/${artistId}/albums`));
  return (
    <ul>
      {albums.map(album => (
        <li key={album.id}>
          {album.title} ({album.year})
        </li>
      ))}
    </ul>
  );
}

// This is a workaround for a bug to get the demo running.
// TODO: replace with real implementation when the bug is fixed.
function use(promise) {
  if (promise.status === 'fulfilled') {
    return promise.value;
  } else if (promise.status === 'rejected') {
    throw promise.reason;
  } else if (promise.status === 'pending') {
    throw promise;
  } else {
    promise.status = 'pending';
    promise.then(
      result => {
        promise.status = 'fulfilled';
        promise.value = result;
      },
      reason => {
        promise.status = 'rejected';
        promise.reason = reason;
      },
    );
```

Browser Tests Terminal

https://s9zlw3.csb.app/

The Beatles

- Let It Be (1970)
- Abbey Road (1969)
- Yellow Submarine (1969)
- The Beatles (1968)
- Magical Mystery Tour (1967)
- Sgt. Pepper's Lonely Hearts Club Band (1967)
- Revolver (1966)
- Rubber Soul (1965)
- Help! (1965)
- Beatles For Sale (1964)
- A Hard Day's Night (1964)
- With The Beatles (1963)
- Please Please Me (1963)

Console 0 Problems 0 React DevTools 0

aed801575

Ln 1, Col 39 Spaces: 2 UTF-8 LF JavaScript

```
// Note: this component is written using an experimental API
// that's not yet available in stable versions of React.

// For a realistic example you can follow today, try a framework
// that's integrated with Suspense, like Relay or Next.js.

export default function Albums({ artistId }) {
  const albums = use(fetchData(`/${artistId}/albums`));
  return (
    <ul>
      {albums.map(album => (
        <li key={album.id}>
          {album.title} ({album.year})
        </li>
      ))}
    </ul>
  );
}
```

```
// This is a workaround for a bug to get the demo running.
// TODO: replace with real implementation when the bug is fixed.
function use(promise) {
  if (promise.status === 'fulfilled') {
    return promise.value;
  } else if (promise.status === 'rejected') {
    throw promise.reason;
  } else if (promise.status === 'pending') {
    throw promise;
  } else {
    promise.status = 'pending';
    promise.then(
      result => {
        promise.status = 'fulfilled';
        promise.value = result;
      },
      reason => {
        promise.status = 'rejected';
        promise.reason = reason;
      },
    );
    throw promise;
  }
}
```

https://github.com/acdlite/rfcs/blob/first-class-promises/text/0000-first-class-support-for-promises.md

```typescript
export const __use =
  realHook ||
  function __use<TValue>(promise: Promise<TValue>) {
    const statefulPromise = wrapPromiseWithState(promise);

    switch (statefulPromise.status) {
      case "pending":
        throw statefulPromise;
      case "rejected":
        throw statefulPromise.reason;
      case "fulfilled":
        return statefulPromise.value;
    }
  };
```

https://github.com/apollographql/apollo-client/blob/main/src/react/hooks/internal/__use.ts

Note

Only Suspense-enabled data sources will activate the Suspense component. They include:

- Data fetching with Suspense-enabled frameworks like Relay and Next.js
- Lazy-loading component code with `lazy`

Suspense **does not** detect when data is fetched inside an Effect or event handler.

The exact way you would load data in the `Albums` component above depends on your framework. If you use a Suspense-enabled framework, you'll find the details in its data fetching documentation.

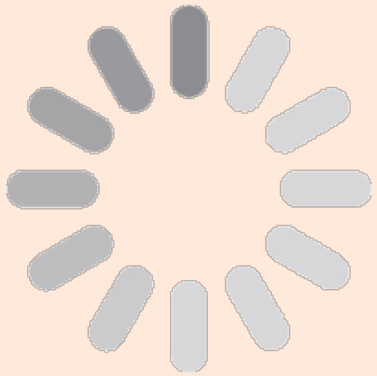
Suspense-enabled data fetching without the use of an opinionated framework is not yet supported. The requirements for implementing a Suspense-enabled data source are unstable and undocumented. An official API for integrating data sources with Suspense will be released in a future version of React.

Andrew Clark
@acdlite · **Follow**

OK everyone, you can use Suspense for data fetching now

1:08 PM · May 4, 2023

319 Reply Copy link



loading states

Introducing...

Introducing...

useSuspenseQuery

Suspenseful useQuery...
that fetches data...
integrated with React 18 Suspense...
including transitions

APOLLO

Home

Search

Your Library

Liked Songs
Playlist · Spotify

Your Episodes
Playlist · Spotify

RVAJS Conference 2023
Playlist · Jerel Miller

React Advanced 2023
Playlist · Jerel Miller

React Summit US 2023
Playlist · Jerel Miller

GraphQL Summit 2023
Playlist · Jerel Miller

daylist
Playlist · Spotify

Release Radar
Playlist · Spotify

RedwoodJS Conf 2023

PLAYLIST

RVAJS Conference 2023

Jerel Miller · 52 songs

Jerel Miller

#	TITLE	ALBUM	DATE ADDED	
1	Jaws - Main Theme M.S. Art	Jaws (Main Theme)	3 minutes ago	2:02
2	Suspenseful Background Andreas Roehrig	Casting Show: Suspense and Drama, Vol. 1	3 minutes ago	1:33
3	Suspense Gold-Tiger	Suspense	3 minutes ago	1:28
4	Loading James Blake	Loading	3 minutes ago	4:44
5	Blank Space Taylor Swift	1989	3 minutes ago	3:51

A Moment Suspended In Time
Underoath

0:09 3:59

Listening on Jerel's Apollo MacBook Pro

APOLLO

Home

Search

Your Library

Liked Songs
Playlist · Spotify

Your Episodes

SIDEBAR

Playlist · Jerel Miller

React Advanced 2023
Playlist · Jerel Miller

React Summit US 2023
Playlist · Jerel Miller

GraphQL Summit 2023
Playlist · Jerel Miller

daylist
Playlist · Spotify

Release Radar
Playlist · Spotify

Redwood JS Conf 2023

USER MENU

PLAYLIST

RVAJS Conference 2023

Jerel Miller · 52 songs

PLAYLIST

#	TITLE		DATE ADDED	
1	Jaws - Main Theme M.S. Art	Jaws (Main Theme)	3 minutes ago	2:02
2	Suspenseful Background Andreas Roehrig	Casting Show: Suspense and Drama, Vol. 1	3 minutes ago	1:33
3	Suspense Gold-Tiger	Suspense	3 minutes ago	1:28
4	Loading James Blake	Loading	3 minutes ago	4:44
5	Blank Space Taylor Swift	1989	3 minutes ago	3:51

A Moment Suspended In Time
Underoath

0:09

PLAYBAR

3:59

Listening on Jerel's Apollo MacBook Pro

APOLLO

Home

Search

Your Library

0:00

0:00

Waiting for localhost...

Let's write some <code />

If we had more time...

Avoiding request waterfalls with useBackgroundQuery

Reading queries with useReadQuery

Looking to the future 🔮

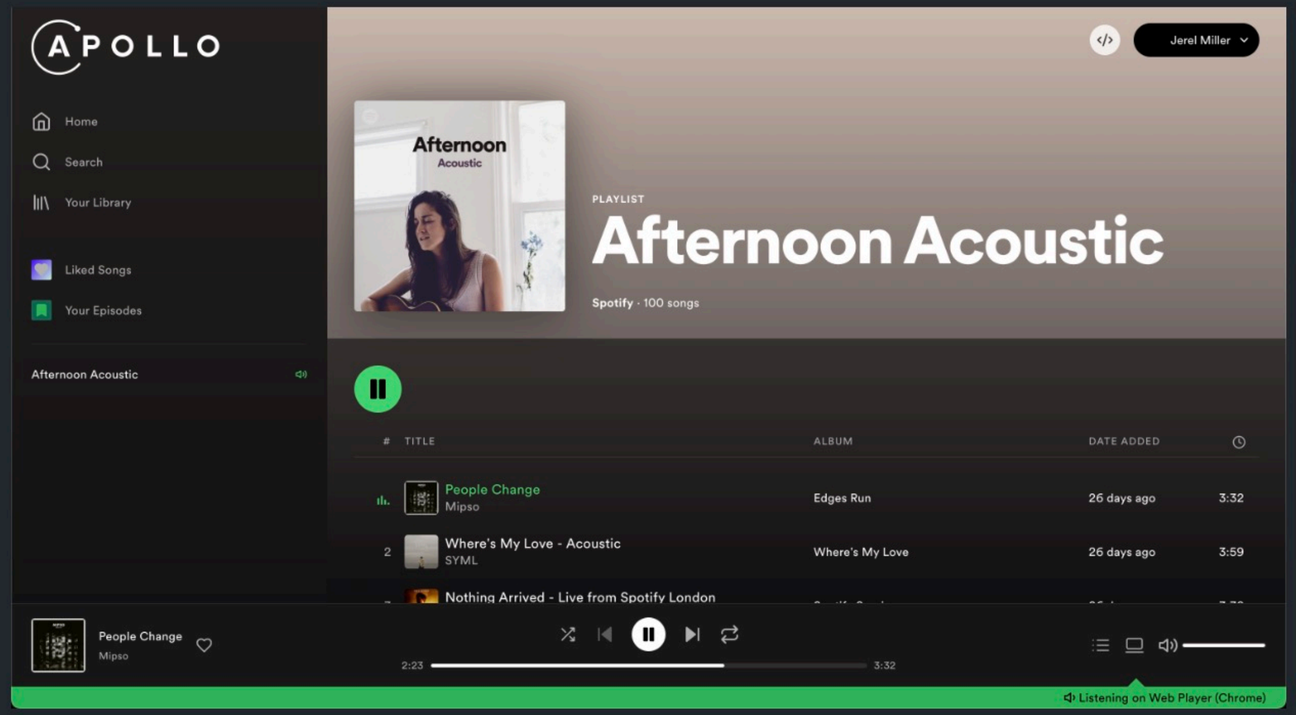
- 3.9.0, currently in development phase
  - Suspenseful useFragment
  - *useLoadableQuery* for interactivity and preloading (#11300)

README.md

TypeScript 99.1% JavaScript 0.5%
CSS 0.2% HTML 0.1%

github.com/apollographql/spotify-showcase

React + Apollo Spotify Showcase



Architecture

The overall API architecture is made up of two GraphQL servers, one exposing subscription/mutation functionality and the other exposing query functionality. Both GraphQL servers use the Spotify REST API as their datasource, but we are hosting the subscription server on dedicated infrastructure (Railway) and the other on serverless functions (Netlify).

Thank you!

go.apollo.dev/c/suspense-talk