← Back to Blog

MAY 4, 2017

ARTICLE | DEVELOPMENT

What's the most developer-friendly React animation library?

Author Alex Holachek

React simplifies the conceptual model of a UI by making it a function of state, but animation complicates things by adding change over time into the equation. When I first started working with React, I tended to stick to CSS transitions and keyframe animations, finding other animation techniques a bit hard to wrap my mind around. CSS can actually get you quite far in React, but it has its limits — try implementing a staggered animation with many different DOM elements or complex easing, and you'll soon find yourself getting frustrated. Recently, when thinking about how to implement a simple sequential enter and exit animation in React, I decided it was time to explore my options. I set a challenge for myself – reproduce the following animation using the major React animation libraries:

This animation has a number of features that make it ever-so-slightly complex to implement, including:

1. Nested enter and leave transitions. Initially, when it is first rendered in React, the grid animates in, followed by the staggered animation of its children cards.

2. A slightly fancy (read: slightly annoying) elastic easing for some of the animations.
3. New cards can be added individually to the cards array and should be animated in.
4. Cards can be removed from the cards array and should be animated out as they leave.
5. When the grid is removed from the DOM, it should wait for its children to animate out before animating itself and leaving the DOM.
6. *Stretch goal*: When shuffled, the cards should smoothly transition to their new positions.

In order to keep the focus on developer ease of use, I decided to limit myself to a couple hours to familiarize myself with each library's API and get as close as possible to building the complete animation. Once the time was up, I'd just post whatever I managed to accomplish and compare it with the animations I built in the other React libraries I tried.

If you'd just like to check out the code, you can go directly to the repo here.

The Results (Roughly in order of preference)

1. React-transition-group & animejs

React-transition-group bills itself as "an easy way to perform animations when a React component enters or leaves the DOM" and used to be a React addon before it was split out as its own package. Animejs, meanwhile, is a straightforward animation library that directly manipulates DOM elements.

Pros:

Using react-transition-group and a JavaScript animation library (animejs, GSAP, popmotion, etc.) ended up being my favorite technique, because it offered the flexibility to make custom, sequenced transitions. This method was the only one I could get working perfectly within the time limit, and offered the most control because I was not forced to work within the constraints of a React wrapper library but could instead directly access DOM elements when necessary.

Animejs is lightweight and powerful, and I find the imperative API more intuitive than the typical React approach for complex sequenced animations.

Cons:

As the most low-level approach, this method might seem intimidating to animation beginners. Those fully invested in the declarative approach of React might find it unpleasant to use this more imperative technique.

You have to be sure to handle lower-level concerns like cancelling in-progress animations if necessary.

- my animation attempt
- the code

2. React-pose

This library is sort of like super-powered CSS transitions in JS with a React-friendly API. There's a lot of attention to detail packed into this small library, such as default appropriate easings depending on the value being animated, and automatic FLIP transitions when components update.

Pros:

In terms of pure ease of use and beginner-friendliness, this one was the winner for me. I could totally see someone new to animations creating something cool with this library.

The automatic FLIP animations are awesome (try shuffling the cards in the demo to see it in action), and the default easings made the animations look beautiful with very little effort.

I liked how the library automatically applies transitions to DOM elements for you instead of just tweening values and making you handle the style updates yourself.

Cons:

I was not able to get the example working perfectly due to a bug that causes repeat animations to have "ghost" elements animated in if the previous animation didn't have time to fully complete (I've filed an issue so hopefully it will be fixed soon).

This library is on the newer side so some bugs or API instability might be part of the deal.

- my animation attempt
- the code

3. React-spring

This newcomer melds the powers of react-motion and react-animated into one user-friendly library (and the docs have tons of cool examples).

Pros:

React-spring's keyframes API, which I used in my example to sequence animations, is intuitive and makes great use of async/await (but it's marked as experimental in the docs).

Cons:

Like Popmotion Pose, this library is quite new. I was able to find a fairly serious bug — there seemed to be a serious memory leak created when adding new cards, though this could have been something I did incorrectly. Ultimately, I couldn't quite get the whole thing working the way I wanted to.

- my animation attempt
- the code

4. React-move

This is a lightweight library that helps D3 and React work together.

Pros:

This library is essentially a React-ready wrapper around D3, so a lot of the functionality is easy to pick up if you've used D3 before.

Cons:

It ended up not being quite flexible enough for the needs of this task. It's also a bit of a smaller project than some of the others mentioned here.

- my animation attempt
- the code

5. velocity-react

A straightforward option that got me far but then ended up tripping me up when it came time to get the nested leave animations working.

Pros:

The library API will be familiar to anyone who has used VelocityJS before, and the components use a similar API to ReactTransitionGroup. It offers a ton of functionality out of the box: easing options, stagger options, pre-registered animations, easy ways to specify delays, durations, etc. The provided code examples helped me get up and running more quickly.

Cons:

I found the documentation slightly confusing, since it mostly delegates to the main VelocityJS documentation, and didn't answer some questions I had about the React-specific implementation. In addition, the abstraction on top of the react-transition-group did leave me frustrated when I was not able to get the leave transitions to work as I wanted them to.

At one point, this library might have been one of the better options for animating in React, but now with popmotion-pose, react-spring, and react-transition-group v2, there are more powerful, updated alternatives with better documentation.

- my animation attempt
- the code

6. React-motion

An hour and a half wasn't enough time for me to understand the intricacies of this physics-based animation library and to create the example animation.

Pros:

The spring-based animation looks nice, and the animations are cancelable by default.

Cons:

This was by far the most confusing library to get up and running, with seemingly the most code that needed to be written by the developer. I know my example doesn't do it justice. I was not able to get the StaggeredMotion and TransitionMotion components to play nice together– it seems other people have maybe also had similar difficulties. Specifying spring parameters instead of animation timing durations is a shift in thinking and I'm not convinced it's totally necessary for many animations.

- my sad animation attempt 🙁

- the code

AUTHOR

Alex Holachek

More posts from this author

Sign up to receive updates from our blog

First Name*

Last Name*

Email*

CAPTCHA

SUBMIT

WHAT WE DO

Expertise

From concept to design, we'll partner with your team to deliver amazing product and website experiences.

LEARN MORE ⟶

RECENT PROJECTS

Work

See the results of our most recent digital product and website engagements.

SEE OUR WORK ⟶

SHARE ON FACEBOOK SHARE ON TWITTER SHARE ON LINKEDIN SUBSCRIBE VIA RSS

About

Contact

Fresh Tilled Soil

(862) 221-1805