Animate your React App with Pose

Written by Seth Corker on Benevolent Bytes

How to bring your React app to life with animation using Pose

Animation on the web can be challenging because there are many ways to achieve the same goal. You could use CSS to achieve transitions or more complex animations or you could try one of the many JS libraries available for animation.

Have you ever asked yourself one of the following:

> Why is animation so difficult in React?
>
> How do I animate a component in on mount?
>
> Do I use CSS or JS to animate my components?

I use CSS animations whenever possible but they can quickly become unwieldy beyond basic `:hover` and `:focus` states. It's possible to achieve a lot by toggling classes to your React components but the most flexible and easy to use method I've found is to use the React animation library Pose, from the fine people over at Popmotion. The techniques I use here can be used in React, React Native and Vue. It's easy to use, very powerful and produces great looking animations.

Why should I add animations to my web app?

If you aren't fully convinced why you should use animations in your React app, allow me to champion some reasons.

1. **You use React/Vue already** If you already use a modern component based JS framework like React or Vue, you already know the benefits. React allows for interfaces that are dynamic and responsive for users. Animation is made easier than ever with React and doesn't have to be a nightmare.
2. **Delight your users** A user doesn't click a button and wait for an entire page to be returned from the server, user's expectations are for rich experiences which animation enhances. It's icing on the cake that elevates your app.
3. **Contextualize actions** A static webpage tells the user that nothing is happening. Animations can help contextualize what the user is doing and how it impacts change in your app. A loading indicator that animates, tells the user your app is alive and is (hopefully) doing something in the background. Clicking a delete button and seeing the item fade out or slide away, gives the user trust that the action has taken place. It hasn't just popped out of existence.

Know that you know why, let's take a look at how I use Pose to animate a simple UI.

A Chat App Example

<AutoplayVideo src={[{ src: "s3-bucket://build/2019-06-23-bring-your-react-app-to-life/chat-anim.2019-06-23-5_18_48-pm.webm", type: "video/webm", },]} poster="s3-bucket://build/2019-06-23-bring-your-react-app-to-life/chat-anim.2019-06-23-5_18_48-pm-poster.jpg" />

The animations in action

See the code in action or take a look at the repo on GitHub:

How the message bubble animation works

```
const ChatWindow = posed.ol({
  visible: {
    staggerChildren: 120
  },
  hidden: {
    staggerChildren: 100
  }
});

const MessageItem = posed.li({
```

```
      visible: {
        x: 0,
        opacity: 1
      },
      hidden: {
        x: ({ fromDirection }) => (fromDirection === "left" ? -350 : 350),
        opacity: 0
      }
    });

    function Chat({ messages, visible }) {
        return(
            <ChatWindow
              className="chat"
              pose={visible ? "visible" : "hidden"}
            >
              {messages.map(m => (
                <MessageItem
                  className={`container ${m.isOwner ? "sent" : "received"}`}
                  fromDirection={m.isOwner ? "right" : "left"}
                  key={m.id}
                />)
              }
        </ChatWindow>);
    }
```

Simplified view of Chat, focused on the animation aspect The two main components that make this animation work are the *ChatWindow* and *MessageItem*. We use `posed.ol` to create a component that will stagger the animations on child components by 120ms when it becomes visible and 100ms when the component hides. We can control whether the *Chat* is visible with a simple prop. We then use `posed.li` for the child components. They start in a hidden state where they are off the screen by 350px (either to the left or right depending on whether we are the sender or recipient of the message), so when they are told to become visible, they animate towards an opacity of 1 and an x coordinate of 0. That's it.

Pose handles the animation for us, we simply define the possible states we want to animate between and pose does the rest. This is a very flexible system that's easy to understand and produces really nice results with minimal effort.

How the person badges animate in and out

```
import React from "react"
import posed, { PoseGroup } from "react-pose"
import Person from "./Person"

const PersonItem = posed.li({
  enter: {
    opacity: 1,
    scale: 1,
    delay: ({ i }) => i * 100,
  },
  exit: {
    opacity: 0,
    scale: 0,
    delay: ({ i }) => i * 20,
  },
})

function PersonSelector({ people }) {
  return (
    <ul className="person-selector">
      <PoseGroup animateOnMount>
        {people.map((p, i) => (
```

```
          <PersonItem className="item" key={p.id} i={i}>
            <Person {...p} />
          </PersonItem>
        ))}
      </PoseGroup>
    </ul>
  )
}

export default PersonSelector
```

The PersonSelector component that sits at the top of the chat window We make use of the built in **enter** and **exit**states for our *PersonItem.* We want the animation to happen on mount so we wrap the children in a *PoseGroup* with the **animateOnMount**. When the component mounts or the **people** prop changes this will trigger the animation. We also make use of the **delay** property in a similar way to the **staggerChildren** property in our *ChatWindow* component. This gives a staggered animation as each Person fades and scales in.

How do I add animations to my React app?

To get started today, take a look at the getting started guide for pose, it takes you through the fundamentals so you can start breathing life into your React apps. (as well as React Native or Vue). Pose is a great library that delivers good looking animations without having to spend hours tweaking CSS animations or dealing with some other complex animation libraries. Set up your states or poses and let Pose take care of the rest.