Framer Motion Shared Layout - Create an image gallery in Framer Motion - Animating between components in React

Written by Seth Corker on Benevolent Bytes

What we're making

A simple interaction where the user can click on an image on the right side and it will expand over to the left, becoming the primary image. This technique could be used as an image gallery or a product chooser for an ecommerce site. Take a look at the CodeSandbox if you want to tinker straight away.

I've also done a simpler effect before animated layout transitions, take a look at my tap to expand image interaction made with Framer Motion

<AutoplayVideo src={[{ src: "s3-bucket://build/2021-04-24-framer-motion-shared-layout-gallery/image-gallery-animation-demo.mp4", type: "video/mp4", },]} controls poster="s3-bucket://build/2021-04-24-framer-motion-shared-layout-gallery/image-gallery-animation-demo-poster.jpg" />

How to create the effect

We'll be using React and the Framer Motion animation library to create this effect. The key components that make this relatively easy to execute is `AnimateSharedLayout` which allows for different components to animate between each other and `AnimatePresence` which allows components to animate on mount or unmount in React. We're using `motion` components too but we're keeping the default values and not setting the `exit`, `animate` and `initial` props. Let's start with some CSS to get things into position.

A note about the CSS

To make the animations predictable, I've opted for set sizes. This isn't necessary but it allows us to achieve the effect with fewer elements and less handling of edge cases. You'll see the `.container` which houses everything on the page is centered on the page, has a static height of `620px` and has `overflow: hidden`. This last property is useful because there are 3 images on the right but when we click on one to expand it, another the current primary image is added to this list as the one we clicked on becomes the new primary image. If we didn't have this we'd see 4 images on the side panel. By hiding the overflow, we only ever see 3.

The `.primary-container` also has a static height of `620px` and `min-width: 1070px`. This is so that the images on the right (the `.product-gallery`) never move to the left as the primary image is unset and reset on the page. Without this, the images move all the way to the left as the primary image disappears before the next one take its place. The overall effect isn't desirable so it's easier to just ensure the container will always be wide enough to hold the right gallery in position.

The other minor tweak is setting `.product-gallery` with `z-index: 1`. This adjusts the gallery so it's always on top, when the primary image is replaced it will animate to the right but it performs a cross-fade which is quite fast and unnecessary so we mask it by making sure other images are always in front. That's all the important pieces out of the way, let's take a look at how we use `AnimatePresence` for the primary image and the gallery images on the right.

Using Animate Presence

Primary Image The first place we use `<AnimatePresence />` is for the primary image itself.

<AutoplayVideo src={[{ src: "s3-bucket://build/2021-04-24-framer-motion-shared-layout-gallery/animate-presence-1.mp4", type: "video/mp4", },]} poster="s3-bucket://build/2021-04-24-framer-motion-shared-layout-gallery/animate-presence-1-poster.jpg" controls autoPlay={false} preload="none" />

This video shows the different adding the AnimatePresence component makes to the primary image

If we don't wrap the primary image in an `AnimatePresence` component then we'll see the current primary image disappear and then the new image animate in from the right. Wrapping the component tells Framer Motion to animate the current primary image out as the new one enters which is a nicer transition.

```
<AnimatePresence>
  <motion.img
    key={primaryProduct}
    className="primary-product-image"
```

```
    src={`https://picsum.photos/id/${primaryProduct}/1280/620`}
    alt=""
    layoutId={`product-${primaryProduct}`}
  />
</AnimatePresence>
```

A code snippet showing how AnimatePresence works

Gallery Images The next place `<AnimatePresence />` is used is for the gallery on the right.

<AutoplayVideo src={[{ src: "s3-bucket://build/2021-04-24-framer-motion-shared-layout-gallery/animate-presence-2.mp4", type: "video/mp4", },]} controls autoPlay={false} preload="none" poster="s3-bucket://build/2021-04-24-framer-motion-shared-layout-gallery/animate-presence-2-poster.jpg" />

This video shows the different adding the AnimatePresence component makes to the images in the gallery

It's a little more obvious why we need this. As we click on an image in this gallery, the one we click on will be come the new primary image and we'll add the current primary image back to the gallery. Without this component telling Framer Motion to animate in the new image, the current primary image will appear underneath the last image on the right and the remaining images will slide upward to take the empty space. This isn't a bad effect but it wasn't the effect I was trying to create, feel free to play around with the animation to get the look and feel you're going for!

```
<AnimatePresence>
  {productIds.map(id => (
    <ProductImage id={id} key={id} onExpand={setAsPrimary} />
  ))}
</AnimatePresence>
```

A code snippet showing how AnimatePresence works

Notice we set the ***`key`*** *prop, this is important in React in general when mapping through an array but it's especially important for Framer Motion here because it ensures each* ***`<ProductImage />`*** *is unique and it can animate them properly.*

This is all very well but now it's time for the main course.

The Animate Shared Layout component

We use the `AnimateSharedLayout` component to animate from a gallery image to the primary image. To achieve the effect we need to choose the components to animate between and decide which shared layout transition we want to use.

The layoutId prop `<AnimateSharedLayout />` works by identifying the elements that in the states they should animate between, to tell Framer Motion which elements we want to treat as the same we need to share a `layoutId` prop. This ensures that although the elements can live in different components as long as they are descendants of the `AnimateSharedLayout` component, they will be treated as the same element animating between different states. This is essential because we need to make an interaction that gives the effect of a `<ProductImage />` on the right expanding over to the left and becoming the primary image.

The first place we add the `layoutId` prop is on the primary image.

```
<motion.img
  key={primaryProduct}
  className="primary-product-image"
  src={`https://picsum.photos/id/${primaryProduct}/1280/620`}
  alt=""
  layoutId={`product-${primaryProduct}`}
/>
```

The second is within the `ProductImage` component.

```
function ProductImage({ id, onExpand }) {
  return (
    <motion.img
      src={`https://picsum.photos/id/${id}/200/200`}
      alt=""
```

```
      onClick={() => onExpand(id)}
      className="related-product-image"
      layoutId={`product-${id}`}
    />
  )
}
```

Choosing crossfade vs. switch Switch is the type of transition which happens by default if no `type` prop is set. The old element will be hidden instantly when a new one enters, and the new one will perform the full transition. This is useful in some situations as it it gives the appearance of moving from one state to the other. For our animation though, we want want the new image to move over and the old one to move out simultaneously, this is where crossfade comes in. Crossfade results in both elements performing the same transition. It blends the different states of the elements that share the same `layoutId`. This is preferable for the effect we want so there are no gaps in the gallery.

The final product

Here's the final product in CodeSandbox if you want to see the code and preview side-by-side. Play around with it and tweak it to your liking.

Resources

- Framer Motion is the animation library I reach too often because it's powerful and easy to use.
- Take a look at another tutorial if you're interested in animating between pages in Next.js
- If you don't use Next.js, don't worry, you can still do transitions between pages with React Router and Framer Motion