Making Magic in Framer Motion 2

Written by Seth Corker on Benevolent Bytes

Framer Motion is my go-to animation library for React. Matt Perry, the creator of the library and its predecessor Pose, has been tweeting about version 2 (still in beta at the time of writing) so I decided to check it out. It has some great new features I can't wait to use when it's released. Until then, let's take a look at one of my favourites - AnimateSharedLayout.

<AutoplayVideo src={[{ src: "s3-bucket://build/2020-03-28-framer-motion-2/framer-motion-magic-expand-rec.2020-03-28-17_15_50.webm", type: "video/mp4", },]} poster="s3-bucket://build/2020-03-28-framer-motion-2/framer-motion-magic-expand-rec.2020-03-28-17_15_50-poster.jpg" />

An animation made with Framer Motion using the MagicMotion component

What is AnimateSharedLayout?

Framer Motion is introducing a new component, `<AnimateSharedLayout />`. It allows for animating between components easily. Matt Perry's tweet illustrates a great usage of it here. The code is easy to understand and doesn't require a lot of setup to achieve.

The *magic* fits all with a few lines of code!

```
import React, { useState } from "react"
import { motion, MagicMotion } from "framer-motion"

export default function App() {
  const [selected, setSelected] = useState(0)

  return (
    <MagicMotion>
      <ol>
        {screens.map(({ title, color }, i) => (
          <motion.li
            magic
            key={i}
            className={`title ${i === selected && "selected"}`}
            style={{ color: i === selected ? color : "#333" }}
            onClick={() => setSelected(i)}
          >
            {i === selected && (
              <motion.div
                magicId="underline"
                className="underline"
                style={{ backgroundColor: color }}
              />
            )}
            {title}
          </motion.li>
        ))}
      </ol>
    </MagicMotion>
  )
}
```

Code sample from the Magic Motion underline menu demo Wrap the animation in MagicMotion (now AnimateSharedLayout) and assign a magicId (now layoutId) prop to the components you want to animate between. So let's jump into another example and I'll break it down.

Experimenting with AnimateSharedLayout in Framer Motion

I decided to get a better idea of how this works and see what it can do by making my own experiment. The codesandbox is below to play around with. Click on the dates to show an expanded view. Calendar animation codesandbox shown in the title card animation In version 1, this would require a bit more setup to get right and

we would be limited to a single component to perform the animation. This new approach gives developers more flexibility as it allows for different components to be used. The transition can then be linked together with a global identifier.

How does it work?

Two components make up the core of the interaction, `<ExpandedCard/>` and `<CompactCard/>` they both contain `<motion.div/>` components with the same `layoutId` prop set to `expandable-card`. Let's take a look at the components in their entirety.

```
function ExpandedCard({ children, onCollapse }) {
  return (
    <>
      <motion.div
        className="card expanded"
        layoutId="expandable-card"
        onClick={onCollapse}
      >
        {children}
      </motion.div>
      <motion.p
        className="card expanded secondary"
        onClick={onCollapse}
        transition={{ delay: 0.3 }}
        initial={{ opacity: 0, top: "6rem" }}
        animate={{ opacity: 1, top: "3rem" }}
      >
        Today is clear
      </motion.p>
    </>
  )
}

function CompactCard({ children, onExpand, disabled }) {
  return (
    <motion.div
      className="card compact"
      layoutId="expandable-card"
      onClick={disabled ? undefined : onExpand}
    >
      {children}
    </motion.div>
  )
}
```

Code sample from a Framer Motion experiment The CSS for the two components defines the two states we want to animate between. The expanded card also contains some more information but we animate this separately so it doesn't just appear, instead, it slides in from the bottom.

Transitioning between the two components is as easy as wrapping them in a `<AnimateSharedLayout />` component and conditionally rendering the one you want to show. The transition will be handled automgically like so.

```
<AnimateSharedLayout>
  {isExpanded ? (
    <ExpandedCard onCollapse={collapseDate} day={day}>
      <Content day={day} disabled={disabled} />
    </ExpandedCard>
  ) : (
    <CompactCard onExpand={expandDate} disabled={disabled} day={day}>
      <Content day={day} disabled={disabled} />
    </CompactCard>
  )}
```

```
</AnimateSharedLayout>
```

We store in state whether the component is expanded or collapsed and render either the `<ExpandedCard/>` or `<CompactCard/>` respectively. When the component which is currently rendering changes, the layoutId ensures a transition happens between them. The key properties which are changing in the experiment are the corner radius, position, size and background colour.

```
.expanded {
  width: 10rem;
  height: 10rem;
  background-color: navy;
  position: relative;
  left: -100%;
  top: 150%;
}

.compact {
  width: 3rem;
  height: 3rem;
  padding: 0.5rem;
  border-radius: 1.5rem;
}
```

The expanded class is re-positioned and enlarged. The shape changes from a circle to a square with moderately rounded corners and from white to navy blue. There is also another trick we use to ensure the date within the card component animates too.

The `<Content />` component displays the day which changes colour depending on whether it is in a disabled, collapsed or expanded state. To ensure the transition happens, we assign a magicId so even though we render different it in two different places, within a `<ExpandedCard/>` or a `<CompactCard/>`, framer-motion can handle the transition smoothly.

```
function Content({ day, disabled }) {
  return (
    <motion.h1
      className="title"
      magicId="title"
      style={{ opacity: disabled ? 0.2 : 1 }}
    >
      {day}
    </motion.h1>
  )
}
```

The Content React component The disabled state is shown here using an inline style but CSS classes handle the expand and collapse states.

```
.title {
  color: navy;
  font-weight: 800;
  margin: 0;
}

.expanded .title {
  font-size: 5em;
  color: white;
}

.compact .title {
  font-size: 1.5em;
}
```

CSS for the Content component Framer Motion handles the colour transition and size change without any additional changes. `AnimateSharedLayout` is a great addition to this already powerful animation library. It makes complex animations much simpler to implement in React and should allow more fluid transitions on the

web which we are usually more accustomed to on native platforms. I hope this has whetted your appetite. If it has, take a look at how to get access to the beta version and start hacking.

How to use the beta version of framer-motion library

At the time of writing, `2.0.0-beta.31` is the latest version. (The article has been updated to work with beat 42)

Screenshot of NPM framer-motion page showing version history

You can specify the version you'd like to use in your `package.json` like so and start hacking.

```
{
  "name": "using-framer-motion-beta",
  "version": "1.0.0",
  "dependencies": {
    "react": "16.12.0",
    "react-dom": "16.12.0",
    "react-scripts": "3.0.1",
    "framer-motion": "2.0.0-beta.31"
  }
}
```

Should I use it?

Framer Motion version 2 is still in beta, although it seems pretty stable it should be used with caution. The API is unlikely to change but don't go building something meant for production right away. I haven't encountered anything unexpected yet but I've only played around with the beta in a limited capacity. If you're interested in playing around with it right now and don't mind the lack of documentation then go ahead. Changes are likely to be minor between now and release and it's great to see what it's capable of right now. If you're looking for stability, stick with version 1 for now and wait for the official release.

If you'd like to delve into more Framer Motion experiments, take a look at these:

- React Animation: Tap to Expand
- Page Transitions in React Router with Framer Motion