Next.js x Framer Motion - Page Transitions in Next.js with Framer Motion - Using AnimateSharedLayout in Next.js

Written by Seth Corker on Benevolent Bytes

What we're building

I'll help you start adding more delight to your Next.js app (or other React apps) with transitions between pages. We're going to use the `AnimateSharedLayout` component from the Framer Motion API to make pages blend between each other seamlessly.

<AutoplayVideo src={[{ src: "s3-bucket://build/2020-11-15-shared-layout-page-transitions/final.mp4", type: "video/mp4", },]} poster="s3-bucket://build/2020-11-15-shared-layout-page-transitions/final-poster.jpg" />

The transition between Next.js pages with Framer Motion

Why?

The web is a powerful platform that has a lot to offer. By adding seamless transitions between webpages, we begin to shift towards a more native feeling UI which delights user's and brings your website or web app into a class of its own.

Why Next.js?

Next.js is a popular framework for building web apps and for good reason. It builds upon React to offer a great developer experience. One of the most obvious features we're leveraging is the built in routing. We're going to build our transitions upon what's already built in to Next. If you'd like to apply this technique to an existing project, it should be easy to do so.

Why Framer Motion?

Framer motion is an easy-to-use library which can achieve our desired transition effect quickly and is powerful. It gives us the ability to mix a variety of different animation types and can be a good foundation for making your pages even more delightful for users. When combined with Next.js, it offers a rich user experience and developer experience that I love.

What we need to start animating

The two key ingredients are Next.js and Framer Motion. If you're not using Next.js and instead are using another React router library, you'll still get and idea of how this could fit in. It's not exclusive to Next.js or Framer Motion, these are just two of my favourites that work well together. Feel free to mix and match.

About the project

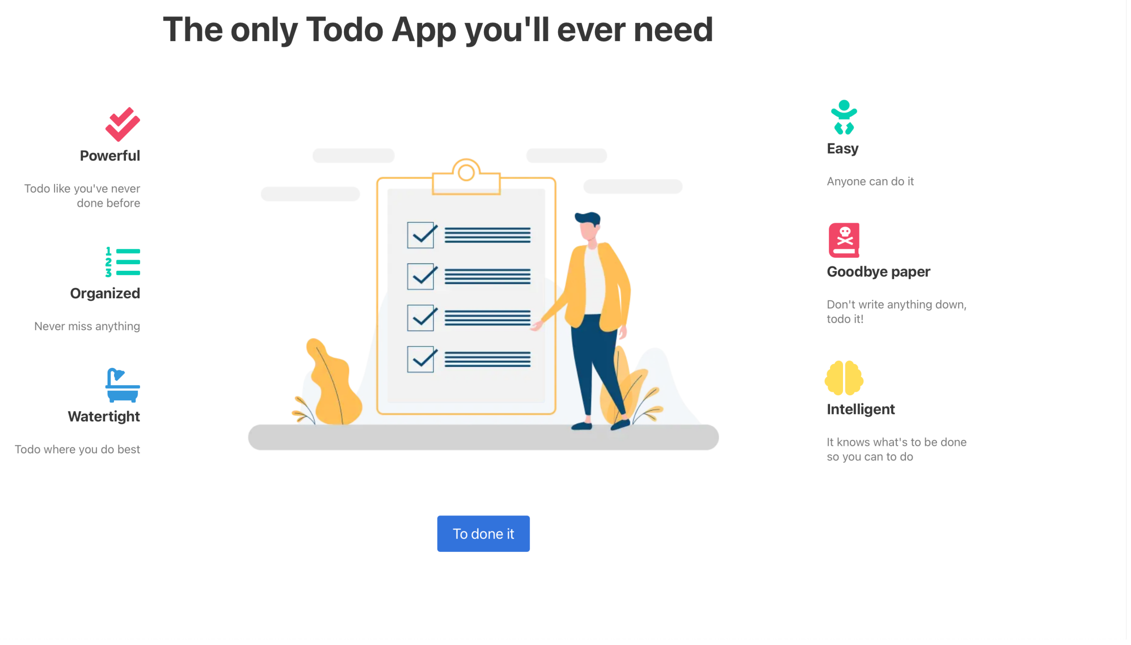
The project contains two pages which we will animate between. The elements that are not shared between pages will animate in while the elements that are shared will animate between pages instead.

Page transition tutorial

Identify the elements that stay the same

For our example we're going to animate between two pages. The first page is a simple landing page with an image, a title, a paragraph and a button. Clicking on the button will navigate to our other page. A features page with the same title and image but it's surrounded but other icons and feature descriptions.

We can't transition between all of these elements because they aren't all on both pages. The title and image are the same so these are the most obvious candidates. The button looks like it could work but the text content differs, it's possible with a bit of work to create something that looks passable but I'll skip the layout transition for this element.


The only Todo App you'll ever need

Powerful

Todo like you've never done before

Organized

Never miss anything

Watertight

Todo where you do best

Easy

Anyone can do it

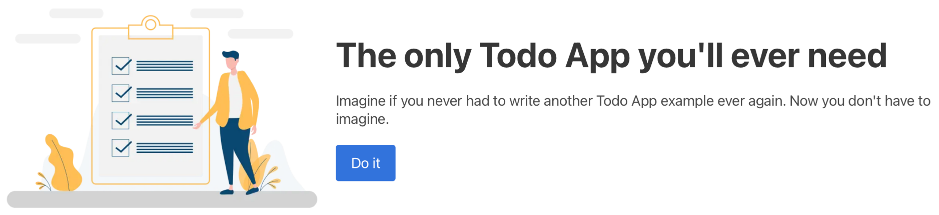
Goodbye paper

Don't write anything down, todo it!

Intelligent

It knows what's to be done so you can to do

Figure 1: Features page


The only Todo App you'll ever need

Imagine if you never had to write another Todo App example ever again. Now you don't have to imagine.

Figure 2: Home page

Adding the AnimateSharedLayout to _app

The first step is to set and forget our `AnimateSharedLayout` component. The component enables us to animate layout changes between different components and tell Framer which elements are shared and therefore should transition to the new state. To set this up, inside `_app` we'll import it:

`import { AnimateSharedLayout } from "framer-motion";` then wrap our `Component`.

Our `_app` looks like this. I'm using the Bulma CSS library and SASS along with FontAwesome icons. The `Component` prop in the context of `MyApp` is the whatever page we are currently on. In Next.js, this acts as a layout which is used across our pages. It's especially useful for our purposes because we want to be able to animate seamlessly between all the pages in our app.

```
import "../styles/theme.sass"
import "@fortawesome/fontawesome-free/css/all.css"
import { AnimateSharedLayout } from "framer-motion"

function MyApp({ Component, pageProps }) {
  return (
    <AnimateSharedLayout>
      <Component {...pageProps} />
    </AnimateSharedLayout>
  )
}

export default MyApp
```

We've added `AnimateSharedLayout` but nothing has changed, let's turn our title and image elements into motion components.

Using layoutId to setup the transitions

On our first page, `index.js`, let's import the motion component from framer-motion, `import { motion } from "framer-motion";`. This component allows us to turn normal html elements into animatable components. Usually, we'd add the `animate` prop, so we can define the animation but the only thing we need for the transition is the `layoutId` prop. It's a string which will be unique across our app.

Our title will go from this:

```
<h1 className="title has-text-weight-bold is-1 is-size-2-mobile is-spaced">
  The only Todo App you'll ever need
</h1>
```

To this:

```
<motion.h1
  className="title has-text-weight-bold is-1 is-size-2-mobile is-spaced"
  layoutId="title"
>
  The only Todo App you'll ever need
</motion.h1>
```

We'll do the same with the figure element that wraps our image.

```
<motion.figure className="image" layoutId="image">
  <Image src="/assets/task.png" layout="responsive" width={780} height={501} />
</motion.figure>
```

Let's replace the matching elements on our `features.js` page too. When the page transitions from `/` to `/features`, framer will treat the elements on one page with the elements on the second page as the same if they share an identical `layoutId`.

Try it out, and we'll see the title and image moves smoothly across the screen.

Completing the look with motion

The approach is very cool because it allows us to go forward and back between pages and the transition works as expected. This is a good starting point, and we've achieved the core effect but I think it needs a bit more.

Let's finish the look off by utilising Framer Motion to add an intro animation to elements that aren't shared across pages.

For elements that are unique to a page, we'll fade them in.

Here's an example of the subtitle on the index page:

```
<motion.p
  className="subtitle"
  initial={{ opacity: 0 }}
  animate={{ opacity: 1 }}
  transition={{ delay: 0.2 }}
>
```

This is a straightforward fade in animation that starts with an opacity of zero and animates to an opacity of 1 after a delay of 200ms.

But what happens when the user first visits a page? Can we animate all elements in? With Framer Motion, yes we can. We can add the `animate` prop to have an intro animation that will play when the user opens the page and this won't affect the transition to the next page.

Here's our animation, a fade in with a slight scale up effect

```
<motion.h1
  initial={{ scale: 0.8, opacity: 0 }}
  animate={{ scale: 1, opacity: 1 }}
  className="title has-text-weight-bold is-1 is-size-2-mobile is-spaced"
  layoutId="title"
>
```

Let's see the final product:

<AutoplayVideo src={[{ src: "s3-bucket://build/2020-11-15-shared-layout-page-transitions/final.mp4", type: "video/mp4", },]} poster="s3-bucket://build/2020-11-15-shared-layout-page-transitions/final-poster.jpg" />

Our animation, notice the routes changing

Where to now?

We've created some cool effects that really breath life into the app with pretty minimal effort for the developer. Framer Motion allows us to create these complex layout animations easily and without compromise. It's easy to tweak how pages animate in or out as well as the transition between pages. I hope you've learned something new and will apply some of these techniques to your projects.

The web is a platform that can do so much more than static webpages. As a platform for rich user experience, we have the opportunity to craft more imaginative user interfaces that really react to the user's actions and exceed expectations. Animation through page transitions is one of the tools we have available to enrich experiences, and they're easy with Next.js and Framer Motion.

Resources

- See the demo
- Read more about how this works and more uses cases, Framer Motion API: AnimateSharedLayout documentation
- Find the completed project repo on GitHub
- Follow along and make your own adjustments with my starter project repo on GitHub.