Adding Page Transitions with gatsby-plugin-transition-link

This guide will cover how to use `gatsby-plugin-transition-link` to animate transitions between pages on your Gatsby site.

Overview

The `TransitionLink` component provides a way of describing a page transition via props on a Link component. It works with many animation libraries, like react-pose, GSAP, animejs, and many others.

Note that currently, as the plugin is based on link navigation, transitions when navigating with the browser buttons are not supported.

For other page transition options, see the overview on adding page animations.

Getting started

First, install the plugin:

```
npm install gatsby-plugin-transition-link
```

Make sure to add the plugin to your `gatsby-config.js`:

```
module.exports = {
    plugins: [
      `gatsby-plugin-transition-link`
    ]
];
```

Finally, import the `TransitionLink` component wherever you want to use it:

```
import TransitionLink from "gatsby-plugin-transition-link"
```

Predefined transitions

You can use the `AniLink` component to add page transitions without having to define your own custom transitions. It's a wrapper around `TransitionLink` that provides 4 predefined transitions: `fade`, `swipe`, `cover`, and `paintDrip`. You can preview them at this demo site.

To use AniLink, you will need to install the `gsap` animation library:

```
npm install gsap
```

Then, import the AniLink component:

```
import AniLink from "gatsby-plugin-transition-link/AniLink"
```

Finally, make sure you provide your desired animation's name as a blank prop to `AniLink`:

```
<AniLink paintDrip to="page-4">
  Go to Page 4
</AniLink>
```

Options like transition duration, direction, and more are customizable with props. See the documentation of AniLink for more details.

Custom transitions

You have two main methods of creating page transitions:

1. Use the `trigger` function defined in your `exit`/`entry` prop. More details in the 'Using the `trigger` function' subsection.
2. Use the props passed by `TransitionLink` to define your transitions. More details in the 'Using passed props' subsection.

Additionally, you can specify a number of props and options on the `TransitionLink` component, like `length`, `delay`, and more. For more options and details, see the documentation of TransitionLink. For further examples of usage, visit the plugin's GitHub repository.

Using the trigger function

You can specify a `trigger` function that will handle the animation. This is useful for *imperative* animation libraries like animejs or GSAP that specify animations with function calls.

```
<TransitionLink
  exit={{
    length: length,
    // highlight-next-line
    trigger: ({ exit, node }) =>
      this.someCustomDefinedAnimation({ exit, node, direction: "out" }),
  }}
  entry={{
    length: 0,
    // highlight-next-line
    trigger: ({ exit, node }) =>
      this.someCustomDefinedAnimation({ exit, node, direction: "in" }),
```

```
  }}
  {...props}
>
  {props.children}
</TransitionLink>
```

Using passed props

The exiting and entering pages/templates involved in the transition will receive props indicating the current transition status, as well as the `exit` or `entry` props defined on the `TransitionLink`.

```
const PageOrTemplate = ({ children, transitionStatus, entry, exit }) => {
  console.log(transitionStatus, entry, exit)
  return <div className={transitionStatus}>{children}</div>
}
```

You can combine these props with a *declarative* state-based animation libraries like react-pose or react-spring to specify transitions for exiting and entering a page.

If you want to access these props in one of your components instead of a page/template, you should wrap your component in the `TransitionState` component. This component takes a function that will have access to the same props as above, which you can then use in your component.

Here's an example using `TransitionState` and `react-pose` to trigger enter/exit transitions for a `Box` component.

```
import { TransitionState } from "gatsby-plugin-transition-link"

const Box = posed.div({
  hidden: { opacity: 0 },
  visible: { opacity: 1 },
})

<TransitionState>
      {({ transitionStatus, exit, entry, mount }) => {
        console.log("current page's transition status is", transitionStatus)
        console.log("exit object is", exit)
        console.log("entry object is", entry)

        return (
            <Box
              className="box"
              pose={
                mount // this is true while the page is mounting or has mounted
                  ? 'visible'
```

```
                    : 'hidden'
                }
              />
          )
        }}
</TransitionState>
```

Now, the `Box` component will be aware of whether the page it's a child of is mounting or unmounting, and it will fade in/out accordingly.

Excluding elements from page transitions

You may want to have elements on a page that persist throughout the page transition (*ex. a site-wide header*). This can be accomplished by wrapping elements in a persistent layout component by using the following plugin option in your `gatsby-config.js`.

```
module.exports = {
    plugins: [
       {
          resolve: "gatsby-plugin-transition-link",
          options: {
              layout: require.resolve(`./src/components/Layout.js`)
            }
       }
    ]
];
```

As always, check out the installation docs for more information.

Further reading

- Official documentation
- Source code for plugin
- Demo site
- Blog post: 'Per-Link Gatsby page transitions with TransitionLink'
- Using transition-link with react-spring