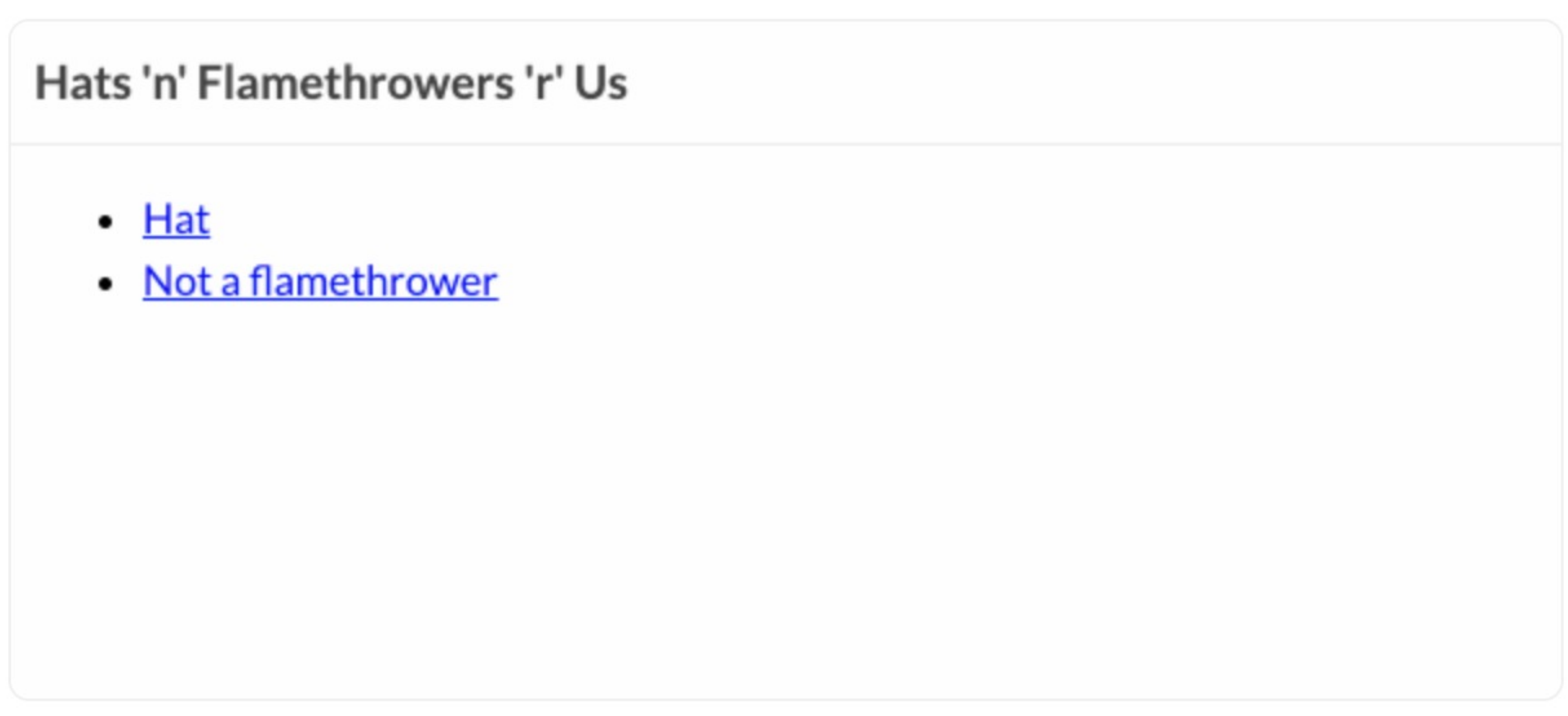
Visualizing loading routes

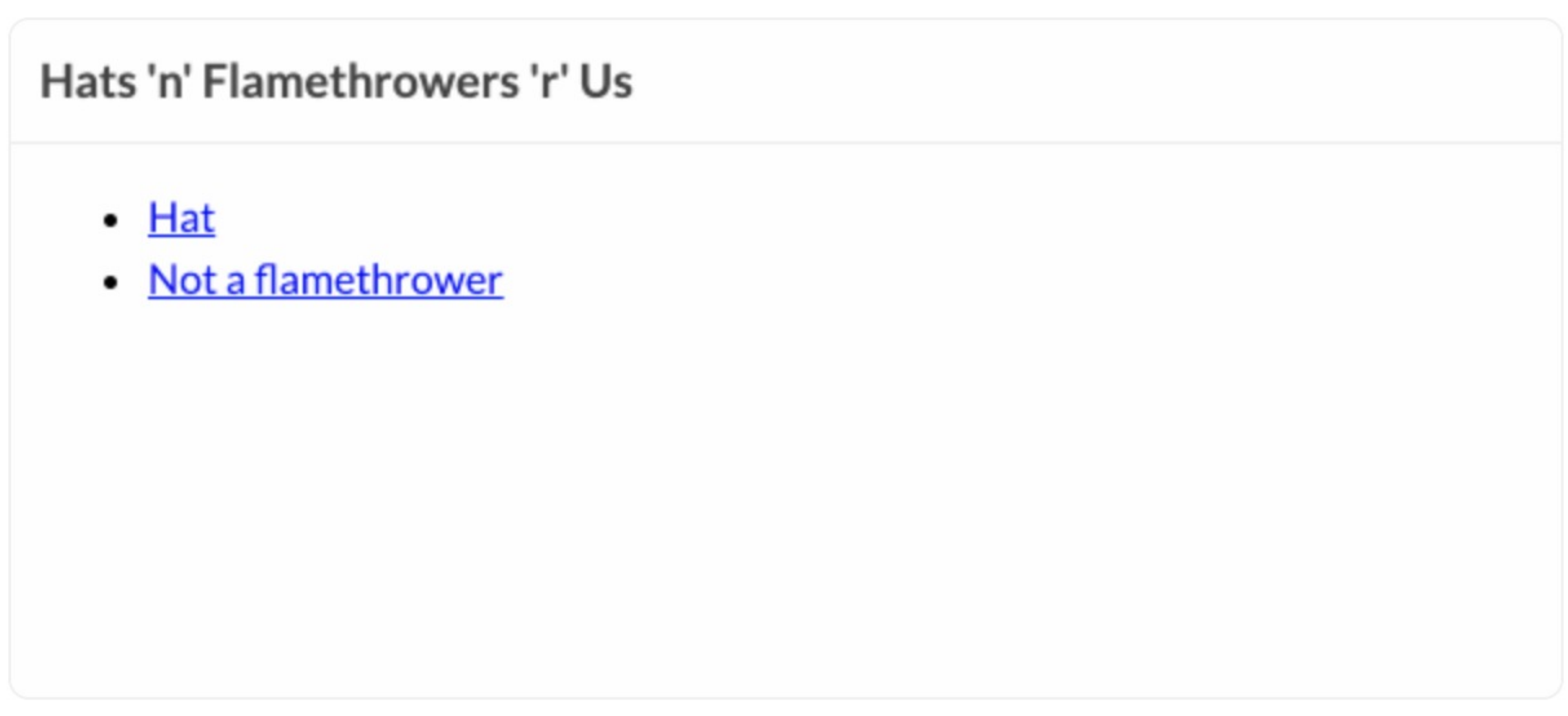
When routes take a long time to load, you'll want to display some sort of loading indicator to the user — and there a number of approaches that you *could* take. One option would be to show a fallback with `<Suspense>`, just as with the initial load. But this looks a bit shit.


Hats 'n' Flamethrowers 'r' Us

- Hat
- Not a flamethrower

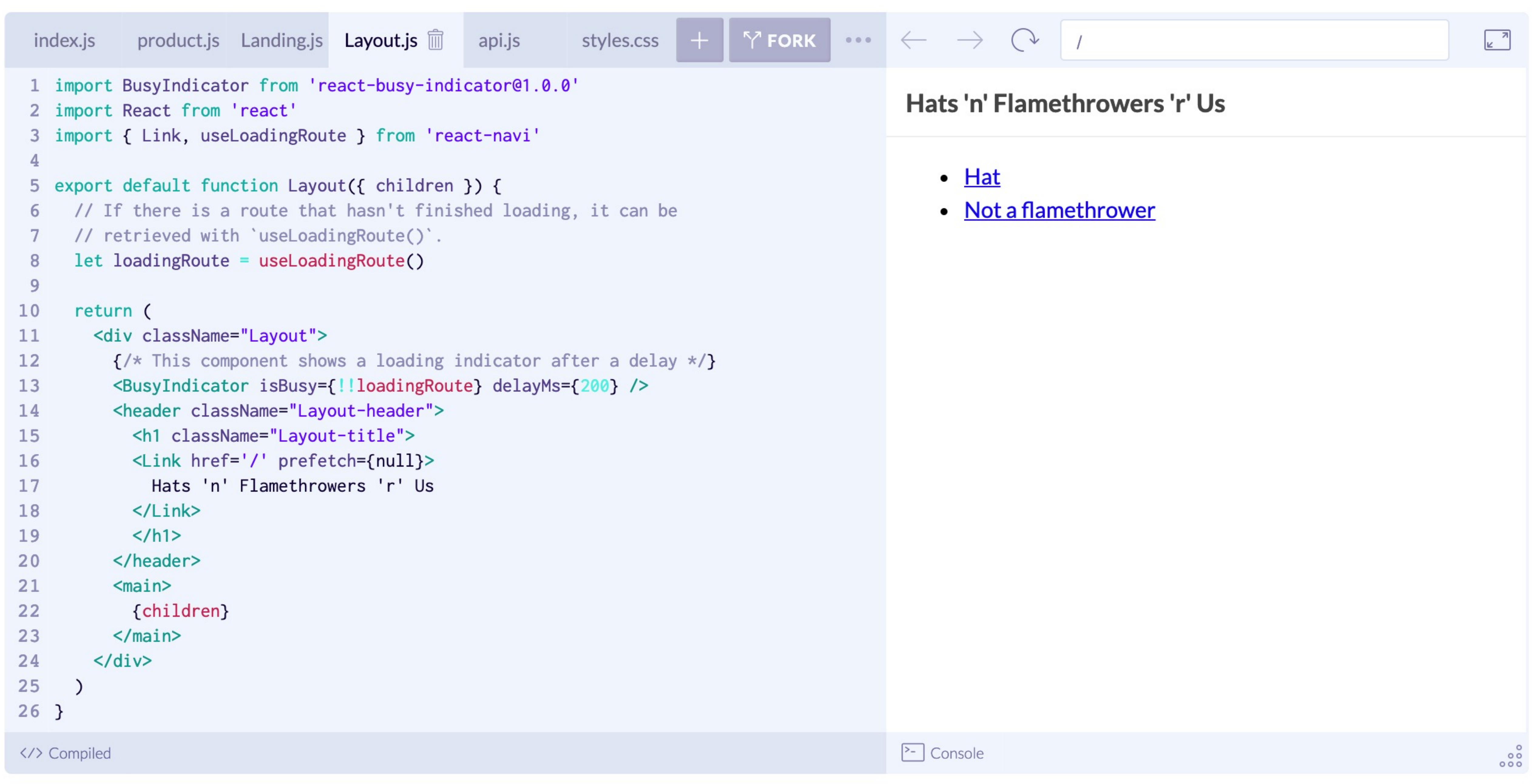
What you'd *really* like to do is to display a loading bar over the current page while the next route loads… well, unless the transition only takes 100ms. Then you probably just want to keep displaying the current page until the next one is ready, because showing a loading bar for only 100ms also looks a bit shit.

Suspense should improve with time, and there are already plans to improve support for cases like this. The caveat is that it could be a while before it works on the server.


Hats 'n' Flamethrowers 'r' Us

- Hat
- Not a flamethrower

There's just one problem. Doing this with currently available tools is ridiculously hard, right? Well actually… You can add it to the above demo in just 3 lines of code, using the `useLoadingRoute()` hook and the `react-busy-indicator` package.



```
import BusyIndicator from 'react-busy-indicator@1.0.0'
import React from 'react'
import { Link, useLoadingRoute } from 'react-navi'

export default function Layout({ children }) {
  // If there is a route that hasn't finished loading, it can be
  // retrieved with `useLoadingRoute()`.
  let loadingRoute = useLoadingRoute()

  return (
    <div className="Layout">
      {/* This component shows a loading indicator after a delay */}
      <BusyIndicator isBusy={!!loadingRoute} delayMs={200} />
      <header className="Layout-header">
        <h1 className="Layout-title">
        <Link href='/' prefetch={null}>
          Hats 'n' Flamethrowers 'r' Us
        </Link>
        </h1>
      </header>
      <main>
        {children}
      </main>
    </div>
  )
}
```

Hats 'n' Flamethrowers 'r' Us

- Hat
- Not a flamethrower

Go ahead and try clicking between these pages a few times. Did you notice how smooth the transition back to the index page is? No? It was so smooth that you didn't notice that there's actually a 100ms delay? Great! *That's exactly the experience that your users want.*

Here's how it works: `useCurrentRoute()` returns the most recent *completely loaded* route. And `useLoadingRoute()` returns any requested-but-not-yet-completely-loaded route. Or if the user *hasn't* just clicked a link, it returns `undefined`.

Want to display a loading bar while pages load? Then just call `useLoadingRoute()`, check if there's a value, and render a loading bar if there is! CSS transitions let you do the rest.

More neat tricks

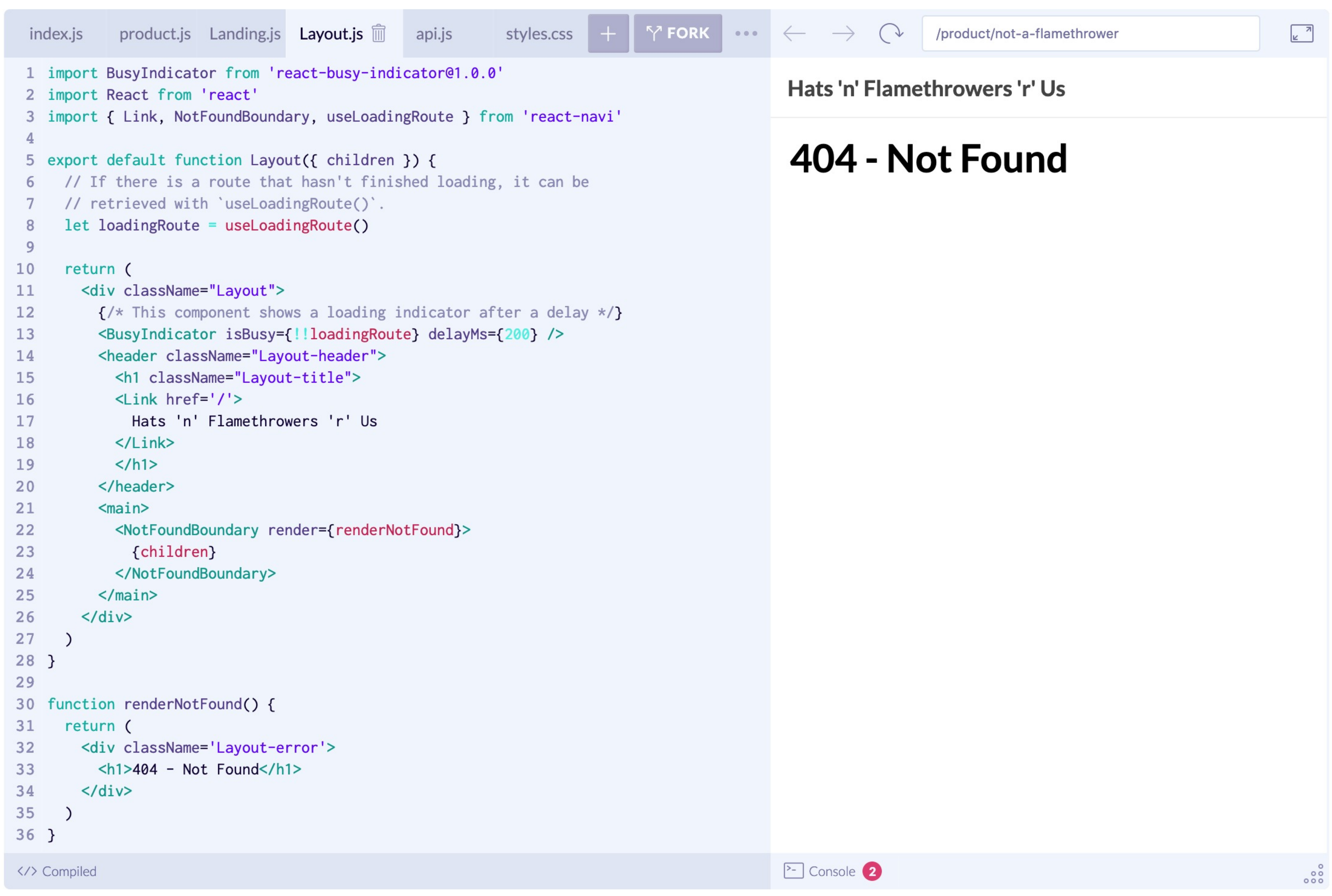
I'm not going to drop the entire set of guides, API reference, and docs on integrating with other tools on you right now. You're reading a blog post, so you might not have time for all that juicy information. But let me ask you a question:

What happens if the route doesn't *load?*

One of the things about asynchronous data and views is that sometimes they *don't bloody work*. Luckily, React has a great tool for dealing with things that don't bloody work: Error Boundaries.

Let's rewind for a moment to the `<Suspense>` tag that wraps your `<View />`. When `<View />` encounters a not-yet-loaded route, it *throws a promise*, which effectively asks React to *please show the fallback for a moment*. You can imagine that `<Suspense>` catches that promise, and then re-renders its children once the promise resolves.

Similarly, if `<View />` finds that `getView()` or `getData()` have thrown an error, then it re-throws that error. In fact, if the router encounters a 404-page-gone-for-a-long-stroll error, then `<View />` will throw that, too. These errors can be caught by Error Boundary components. For the most part, you'll need to make your own error boundaries, but Navi includes a `<NotFoundBoundary>` to show you how its done:



```
import BusyIndicator from 'react-busy-indicator@1.0.0'
import React from 'react'
import { Link, NotFoundBoundary, useLoadingRoute } from 'react-navi'

export default function Layout({ children }) {
  // If there is a route that hasn't finished loading, it can be
  // retrieved with `useLoadingRoute()`.
  let loadingRoute = useLoadingRoute()

  return (
    <div className="Layout">
      {/* This component shows a loading indicator after a delay */}
      <BusyIndicator isBusy={!!loadingRoute} delayMs={200} />
      <header className="Layout-header">
        <h1 className="Layout-title">
        <Link href='/'>
          Hats 'n' Flamethrowers 'r' Us
        </Link>
        </h1>
      </header>
      <main>
        <NotFoundBoundary render={renderNotFound}>
          {children}
        </NotFoundBoundary>
      </main>
    </div>
  )
}

function renderNotFound() {
  return (
    <div className='Layout-error'>
      <h1>404 - Not Found</h1>
    </div>
  )
}
```

Hats 'n' Flamethrowers 'r' Us

404 - Not Found