But that's not all!

Ok, so I think we're about out of time for this blog post. But there's a bunch more details in the docs:

- Using URL Parameters
- Requests, Routes and Matchers
- Nested Routes and Views
- Programmatic navigation
- Guarding routes with authentication
- Improving SEO with static rendering
- Using Navi with react-router
- Using Navi with react-helmet
- Get a head start with a create-react-app based starter

You can also check out the examples directory of the Navi repository for full code examples, including:

- A basic site with create-react-app
- A simple site with static rendering
- A blog with tags, pagination and RSS
- That same blog but written in TypeScript

Oh, and did I mention that Navi is build with TypeScript, so the typings are first-class?

Help me, Obi-Wan Kenobi. You're my only hope.

Ok, so even if you're not Obi-Wan Kenobi, I'd really appreciate your help.

Actually, a lot of little things are way more helpful than they might seem. Can you try Navi out and file an issue for any missing features? *Awesome.* Can you make tiny improvements to the docs as you learn? *Radical.* Can you put something small online and add it to the README's list of sites using Navi? *Phenomenal.*

With that said, there are a few big ticket items that I'd love some help with:

- Navi needs some dev tools. Internally, all data is represented as simple objects called Chunks—which are then reduced into `Route` objects with a Redux-like reducer. As a result of this design, it should be possible for dev tools to provide a useful window into what's going on, along with time-travel capability—but I want to leave this task for the community. So if *you* want to be the person who made Navi's dev tools, let's discuss the details 🤓

- Matcher functions like `mount()`, `lazy()` and `route()` are just Generator Functions. In fact, it's entirely possible to create custom matchers. For instance, you could create a `withTimeout()` matcher that switches routes based on how long they take to load. And if you *do* create useful custom matchers, send a tweet or DM to @james_k_nelson so I can spread the word 🤩

- If you can provide a translation for even a single page from the docs, I'll be forever grateful ❤️

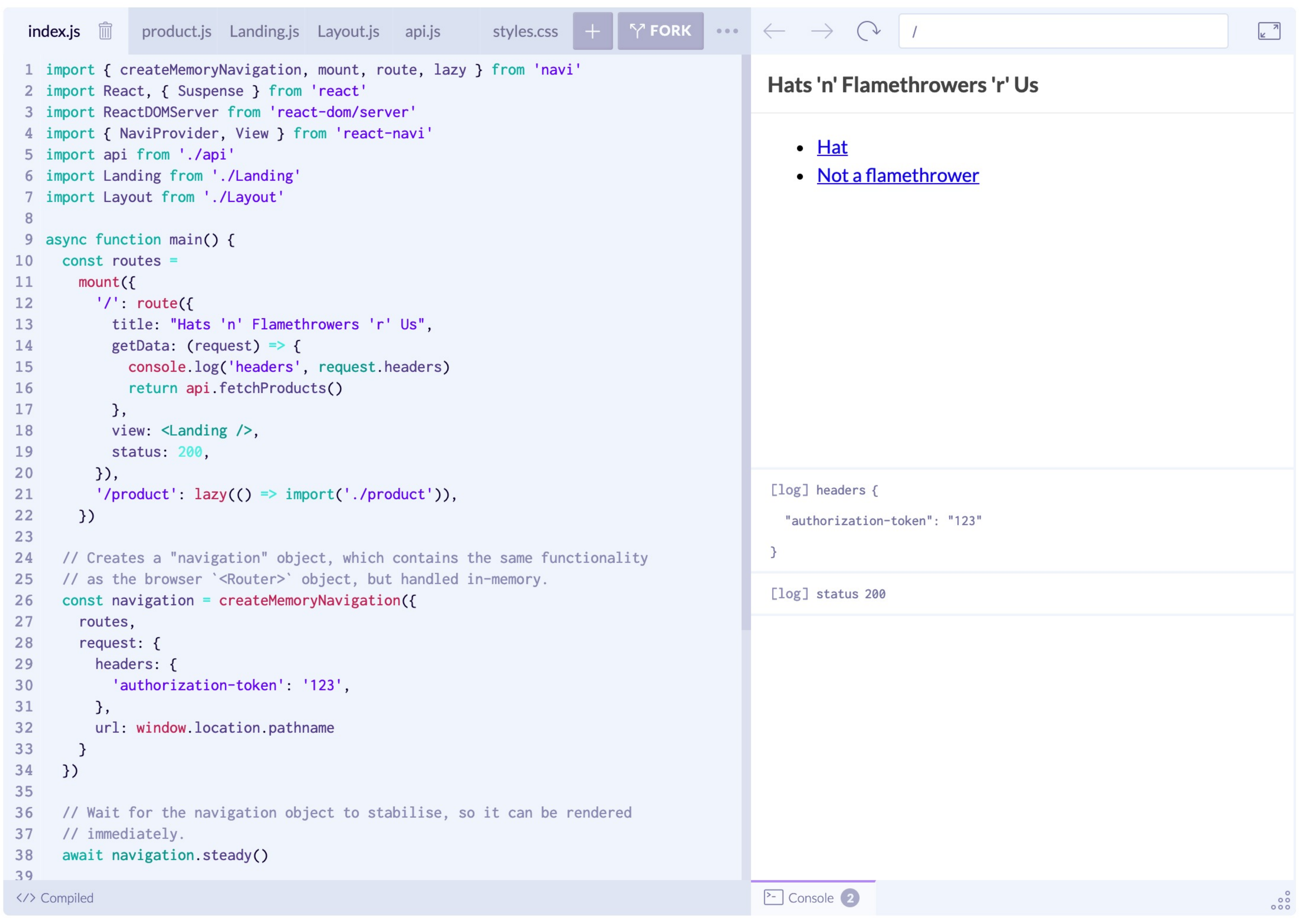
- If you'd like to see this project grow, please give it a 🌟 Star on Github!

One more thing…

Navi now has experimental server rendering support. Matchers like `route()` and `mount()` have access to an entire `Request` object, including `method`, `headers`, and `body`. Your routes aren't limited to matching just a `view` and `data`—they can also match a HTTP `status` and `headers`.

You can now handle routing on the client, the server and in serverless functions with *exactly* the same code.

Of course, you could already do some of this with Next.js—but if all you need is a router, then Navi is a *lot* smaller (and more flexible). If you'd like to know more about the difference, take a read through Navi vs. Next.js. Or if you just want the server rendering demo:



```
import { createMemoryNavigation, mount, route, lazy } from 'navi'
import React, { Suspense } from 'react'
import ReactDOMServer from 'react-dom/server'
import { NaviProvider, View } from 'react-navi'
import api from './api'
import Landing from './Landing'
import Layout from './Layout'

async function main() {
  const routes =
    mount({
      '/': route({
        title: "Hats 'n' Flamethrowers 'r' Us",
        getData: (request) => {
          console.log('headers', request.headers)
          return api.fetchProducts()
        },
        view: <Landing />,
        status: 200,
      }),
      '/product': lazy(() => import('./product')),
    })

  // Creates a "navigation" object, which contains the same functionality
  // as the browser `<Router>` object, but handled in-memory.
  const navigation = createMemoryNavigation({
    routes,
    request: {
      headers: {
        'authorization-token': '123',
      },
      url: window.location.pathname
    }
  })

  // Wait for the navigation object to stabilise, so it can be rendered
  // immediately.
  await navigation.steady()
```

```
[log] headers {
  "authorization-token": "123"
}
[log] status 200
```

As of right now, there's no official package for integrating Navi with Express. There should be one. Please make one. I'll tell the world about it.

But seriously, I *will* tell the world about anything you make with Navi. Here's how I'll do it: the Frontend Armory weekly newsletter. Want to hear about awesome React shit that other readers are making? Join it. You know you want to.

Join Frontend Armory for Free »

But that's it from me today. Thanks *so* much for reading. I've poured my heart into this project because I believe that it'll make *your* life as a React developer so much easier. Now go build something amazing!

P.S. I haven't forgotten about the React in Practice course—I'm working on it right now, and it'll include hooks. I'll have more more details real soon.

Related Links

- Navi's Documentation
- Navi on GitHub
- Follow @james_k_nelson on Twitter