链滴

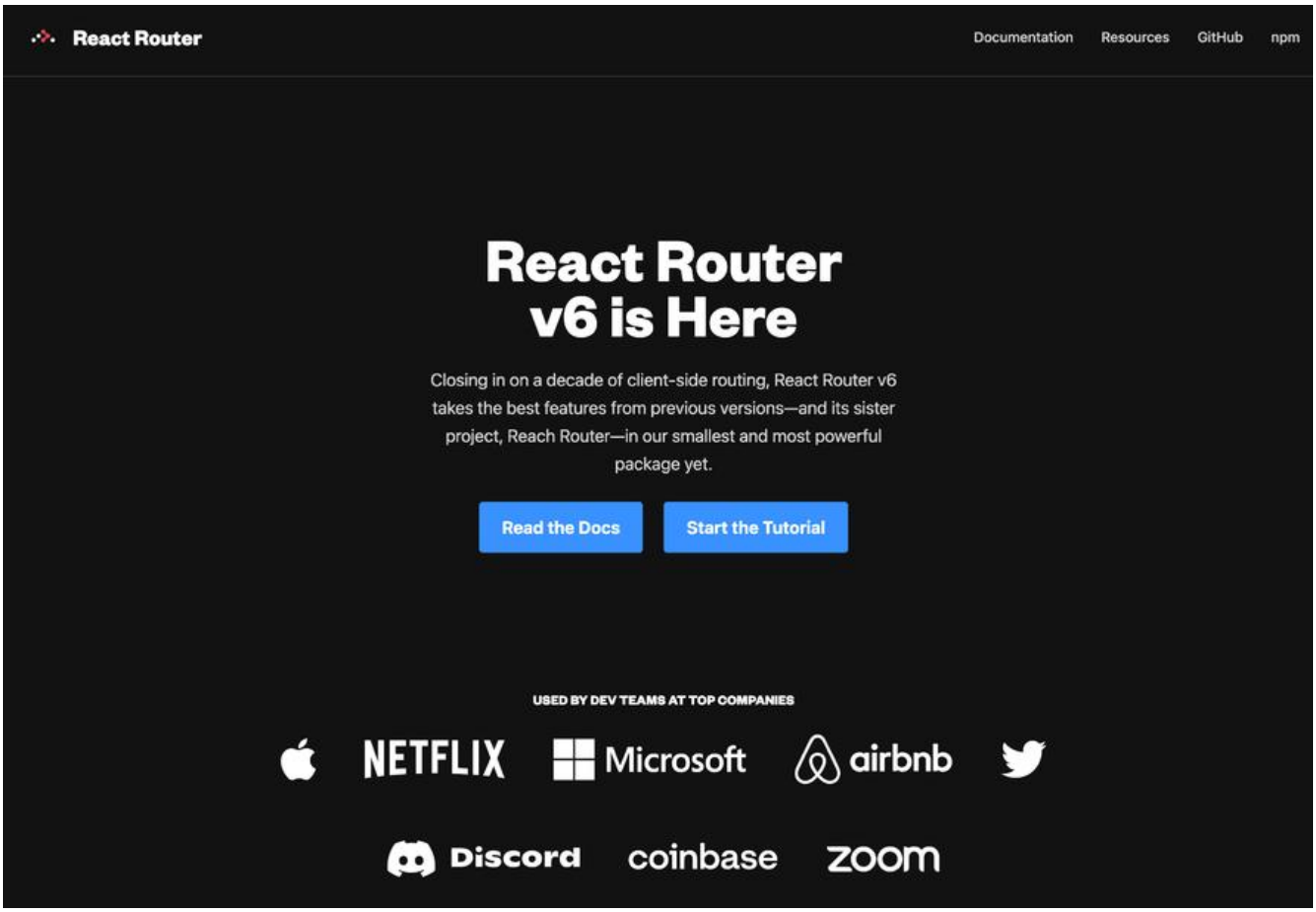
React Router 6 (React 路由) 最详细教程

作者：HiJiangChuan

原文链接：https://ld246.com/article/1648492192613

来源网站：链滴

React Router 经历多个版本的发展，现在已经到了 **React Router 6**。虽然网络上写 React-Router 路由本身的教程很多，但真正讲到 React-Router 6 的并不多。同时因为第 6 版引入了很多新的概念以及大量使用 Hook，因此网上的很多旧教程已经不实用了。这篇文章里我们总结 React Router 6 由器的用法，用例子说明如何实现各种场景和需求，比如程序化跳转等等。



原文链接：React Router 6 (React 路由) 最详细教程

在卡拉云中，我们也大量地使用了 React-Router 6，所以在讲解过程中我们会用一些在实际使用的子来说明问题，但本文的主要例子会放在 github 仓库中，方便你参考。如果你觉得有用，不妨分享加星，或在博客中链回本文，让更多人看到。

本系列中其它优秀教程请参考

- React 表格教程
- React 拖拽教程
- React 富文本组件

当然如果你希望快速搭建后台系统，也推荐尝试卡拉云，可以免掉前后端开发、维护的烦恼

什么是 React-Router

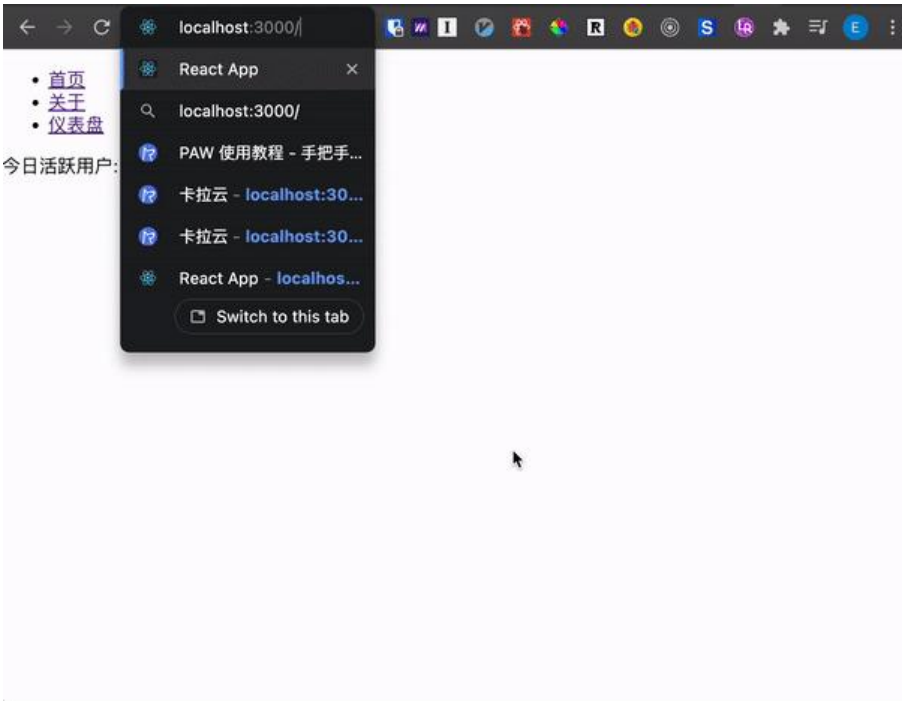
要理解什么是 React-Router 就要先理解什么是 SPA (Single Page Application)，也就是俗称的单应用。

每个单页应用其实是一系列的 JS 文件，当用户请求网站时，网站返回一整个(或一系列)的 js 文件和 HTML，而当用户在某个页面内点击时，你需要告诉浏览器怎么加载另一个页面地址。单页应用中通常有一个 index.html 文件的，所以浏览器自带的 <a> 链接 tag 并不能用来做单页应用的跳转，因此需要一个在 React 中的路由实现。

然而 React 框架本身是不带路由功能的，因此如果你需要实现路由功能让用户可以在多个单页应用中转的话，就需要使用 React-Router。

React-Router 从 2014 年开始开发，到现在已经经历了 6 次大版本迭代，而从它的使用者来看，Netflix, Twitter, Discord 等等大厂纷纷背书，因此 React-Router 已经基本成了在 React 中做路由的默认项。如果你现在还在用老的版本，想要升级，那么可以参考升级教程，否则的话可以一步步参考本文。

在读完本文后，你应该可搭起来如下这样的简单应用，用一个导航栏控制用户可以访问的页面，同时护某些页面，必须在用户登录后才可以进入。



虽然这个应用看起来简单，但是它却包含了 React-Router 中常见的功能和 API，包括

- BrowserRouter
- Link
- Routes
- Route
- Outlet

等等

如何安装 React-Router

安装 React-Router 非常简单，如果你使用的是 yarn 或者 npm，则用通常的安装方式即可

我们先用 create-react-app 脚手架建起一个 app 来

```
npx create-react-app react-router-6-tutorial
```

然后用 npm 安装

如果使用 npm 的话则是

```
npm install react-router-dom@6
```

yarn 安装

```
yarn add react-router-dom@6
```

这样 react-router 就安装好了。注意如果在 web 上的话，你需要的是 react-router-dom 而不是 react-router 这个包。它们的区别是，后者包含了 react-native 中需要的一些组件，如果你只需要做网页用的话，用前者就可以了

React Router API

React Router 的 API 在它的官方文档上已经介绍得比较清楚了，我们这里简单地总结一下几个可能到的 API。具体的用法在下文中我们详细来讲，这里只是作为参考，如果碰上问题可以查一查

BrowserRouter

在 React Router 中，最外层的 API 通常就是用 BrowserRouter。BrowserRouter 的内部实现是用了 history 这个库和 React Context 来实现的，所以当你的用户前进后退时，history 这个库会记住用户历史记录，这样需要跳转时可以直接操作。

BrowserRouter 使用时，通常用来包住其它需要路由的组件，所以通常会需要在你的应用的最外层用，比如如下

```
import ReactDOM from 'react-dom'
import * as React from 'react'
import { BrowserRouter } from 'react-router-dom'
import App from './App`
```

```
ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
, document.getElementById('app))
```

Route

Route 用来定义一个访问路径与 React 组件之间的关系。比如说，如果你希望用户访问 https://your_ite.com/about 的时候加载 <About /> 这个 React 页面，那么你就需要用 Route:

```
<Route path="/about" element={<About />} />
```

Routes

Routes 是用来包住路由访问路径(Route)的。它决定用户在浏览器中输入的路径到对应加载什么 React 组件，因此绝大多数情况下，Routes 的唯一作用是用来包住一系列的 Route，比如如下

```
import { Routes, Route } from "react-router-dom";

function App() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
    </Routes>
  );
}
```

在这里，Routes 告诉了 React Router 每当用户访问根地址时，加载 Home 这个页面，而当用户访问 about 时，就加载 <About /> 页面。

React Router 实操案例

在上文中我们介绍了 React Router 的 API，余下全文中我们用一个实例来说明如何使用 React Route。

首先我们建起几个页面

```
<Home />

<About />

<Dashboard />
```

Home 用于展示一个简单的导航列表，About用于展示关于页，而 Dashboard 则需要用户登录以后可以访问。

首先我们新建一个 router.js 文件，并在其中加载好 React-Router 组件

```
import './App.css';
import { BrowserRouter, Route, Routes } from "react-router-dom"
```

```
function App() {

  return <BrowserRouter>
    <Routes>
      <Route path="/" element={<Home />} />
    </Routes>
  </BrowserRouter>
}

const Home = () => {
  return <div>hello world</div>
}

export default App;
```

这里我们直接在 App.js 中加上一个叫 Home 的组件，里面只是单纯地展示 hello wolrd 而已。接下，我们再把另外两个路径写好，加入 About 和 Dashboard 两个组件

```
import './App.css';
import { BrowserRouter, Route, Routes } from "react-router-dom"

function App() {

  return <BrowserRouter>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
      <Route path="/dashboard" element={<Dashboard />} />
    </Routes>
  </BrowserRouter>
}

const Home = () => {
  return <div>hello world</div>
}

const About = () => {
  return <div>这里是卡拉云的主页</div>
}

const Dashboard = () => {
  return <div>今日活跃用户: 42</div>
}

export default App;
```

此时，当我们在浏览器中切换到 / 或 /about 或 /dashboard 时，就会显示对应的组件了。注意，在面每个 Route 中，用 element 项将组件传下去，同时在 path 项中指定路径。在 Route 外，用 Routes 包裹起整路由列表。

写到这里，我们其实已经完成了一个基本的路由功能，对于绝大多数可以公开访问的网站(或者内部系)，这差不多就已经完结的。但有时，你可能希望知道用户所在的路径，来做一些对应显示和特殊逻辑理，或者是你需要让用户鉴权后才能访问某个路径，那么你需要继续读一下后文几个章节

如何获取当前页路径

如何在 React-Router 中获取当前用户在访问的页面的路径？其实很简单，在 React-Rotuer 6 中，供了一个 hook 钩子，专门用来获得当前路径。在上文的例子中，我们只需要在对应的页面里，比如 bout 中，加上这个 hook 就可以了

首先我们导入 useLocation 这个 hook，然后仿照如下代码就可以获得当前位置

```
import { useLocation } from 'react-router-dom'

const About = () => {
  // 使用 hook
  const location = useLocation();
  const { from, pathname } = location

  return <div>这里是卡拉云的网站，你当前在 {pathname}，你是从 {from} 跳转过来的</div>
}
```

如何设置默认页路径(如 404 页)

在上文的路由列表 Routes 中，我们可以加入一个 catch all 的默认页面，比如用来作 404 页面。

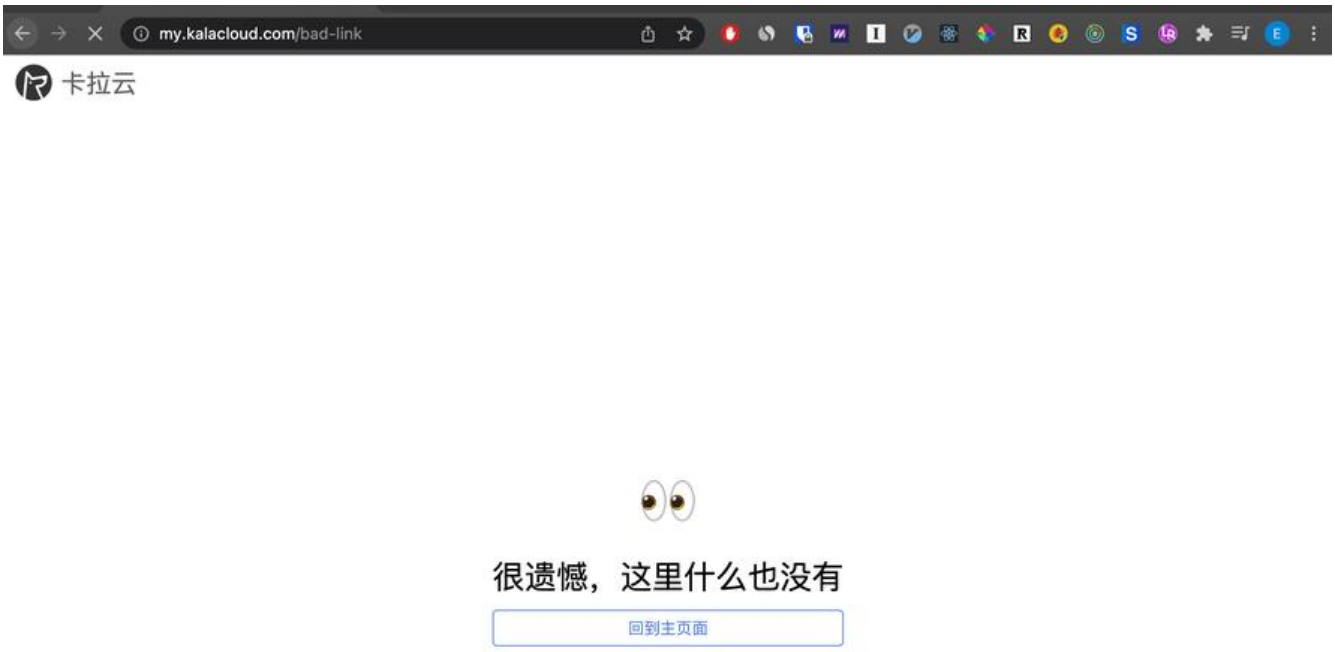
我们只要在最后加入 path 为 * 的一个路径，意为匹配所有路径，即可

```
function App() {

  return <BrowserRouter>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
      <Route path="/dashboard" element={<Dashboard />} />
      <Route path="*" element={<NotFound />} />
    </Routes>
  </BrowserRouter>
}

// 用来作为 404 页面的组件
const NotFound = () => {
  return <div>你来到了没有知识的荒原</div>
}
```

当然你可以把 404 页面做得更好看一点，比如卡拉云中如果访问不存在的链接的话，404 页面如下



如何用 React Router 鉴权并保护路径

总结

本文中我们介绍了如何使用 React-Router，用一个实例说明 React Router 6 中的 API，以及常见的用场景等。如果你对我们的技术博客感兴趣，欢迎继续阅读

- Vue Video.js 教程
- Vue 强制刷新
- Echarts 折线图如何配置
- Vue 弹窗