**安装**

npm install react-router-dom@6

**一个完整的示例**

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

function Home() {

  return (

    <div>

      <Link style={{marginRight: 10}} to='a' >A</Link>

      <Link style={{marginRight: 10}} to='b' >B</Link>

      <Link style={{marginRight: 10}} to='cd' >CD</Link>

    </div>

  )

}

function A() {

  return (

    <div>AAAAAAAAAA</div>

  )

}

function B() {

  return (

    <div>BBBBBBBBBB</div>

  )

}

function CD(){

  return (

    <div>

      <div>CDCDCDCDCDCDCDCDCD</div>

      <Link style={{marginRight: 10}} to='c' >C</Link>

      <Link style={{marginRight: 10}} to='d' >D</Link>

      <Routes>

        <Route path="c" element={<C />}></Route>

        <Route path="d" element={<D />}></Route>

      </Routes>

    </div>

  )

}

function C() {

  return (

    <div>CCCCCCCCCCCCC</div>

  )

}

function D() {

  return (

    <div>DDDDDDDDDDDDDDDD</div>

  )

}

function App() {

  return (

    <BrowserRouter>

      <Routes>

        <Route path="/" element={<Home />} />

        <Route path="/a" element={<A />} />

        <Route path="/b" element={<B />} />

        <Route path="/cd/\*" element={<CD />} />

      </Routes>

    </BrowserRouter>

  );

}

export default App;

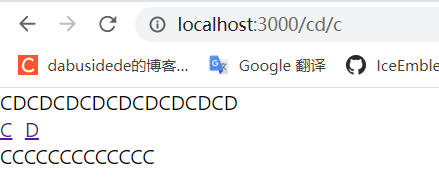
当访问/时显示Home组件

当访问/a时显示A组件

当访问/cd时显示CD组件



当访问/cd/c时显示CD组件其CD组件下的C组件



**BrowserRouter**

Routes必须被BrowserRouter包裹，推荐将BrowserRouter放置在最外层

1. basename

basename可以设置url的前缀

<BrowserRouter basename='/test'>

      <Routes>

        <Route path="/a" element={<A />} />

      </Routes>

</BrowserRouter>

访问/test/a才会显示A

**Routes**

Routes的目的是从多个Route中匹配一个Route进行显示，Routes用于包裹Route

**Route**

Route是一个具有路由的组件，当路由被匹配到是，Route会被渲染出了，示例：

<Route path="/" element={<Home />} />

如下说明Route的props

1. path: 路径

用于匹配路由，其格式有一下几种

/xxx: 完全匹配，只能匹配 /xxx，类似 /xxx/l 匹配不到

/xxx/\*: 匹配以 /xxx 开头的，如 /xxx/lllk/cxvs/aaa/cc/aawe 也可以匹配到

/xxx/:param:

匹配以 /xxx 开头的后面跟一个名称的，如 /xxx/1 可以匹配，单 /xxx/1/aaa不能匹配，在组件中可以使用useParams()获取param的值，示例

<Route path="d/:id" element={<D />}></Route>

// 示例路由 /cd/d/1

// 则打印值为 {\*: 'd/1', id: '1'}

function D() {

  console.log(useParams());

  return (

    <div>DDDDDDDDDDDDDDDD</div>

  )

}

/xxx-\*: 可以匹配到 /xxx-\*\*\*\* 的路由

1. element

要渲染的组件

1. index

等同与path="/"

<Route index element={<Home />} />

**Link**

Link可以在路由之间跳转

示例：

<Link style={{ marginRight: 10 }} to='c' >C</Link>

使用说明：

// 假设当前路由是 /cd

// 跳转到当前路由下的 c，即跳转至 /cd/c

<Link style={{ marginRight: 10 }} to='c' >C</Link>

// 跳转到上级目录，即跳转至 /

<Link style={{ marginRight: 10 }} to='..' >Home</Link>

// 跳转至上级目录下的a，即 /a

<Link style={{ marginRight: 10 }} to='../a' >A</Link>

// 跳转至 /a

<Link style={{ marginRight: 10 }} to='/a' >A</Link>

**重定向**

react-router 提供了一个组件供我们重定向

function Home() {

  return (

    <Navigate to="/a" replace />

  )

}

该示例将路由重定向到 /a

**控制路由跳转**

我们通过useNavigate来控制路由跳转

import { BrowserRouter, useParams, useNavigate } from "react-router-dom";

function Home() {

  let navigate = useNavigate();

  return (

    <div>

      <a onClick={() => {

        navigate('/a');

      }}>A</a>

    </div>

  )

}

3种用法

// push 方式，跳转过去后点击浏览器的返回按钮返回到当前页面

navigate('/a');

// replace 方式，跳转过去后无法通过点击浏览器的返回按钮返回到当前页面

navigate('/a', {replace: true});

// 跳转至 /a 并携带 state，在 /a 的组件种可以使用 useLocation 获取到 state

navigate('/a', { state: { id: 1 } });

**获取当前路由的信息**

使用useLocation可以获取当前路由的信息

function D() {

  console.log(useLocation());

  return (

    <div>DDDDDDDDDDDDDDDD</div>

  )

}

log输出示例

{

    hash: "",

    key: "8tktnq5p",

    pathname: "/cd/d/1",

    search: "?order=1",

    state: { id: 1 }

}