ReactRouter – 快速开始

什么是前端路由

一个路径 path 对应一个组件 component 当我们在浏览器中访问一个 path 的时候，path 对应的组件会在页面中进行渲染

```
const routes = [
  {
    path: '/about',
    component: About,
  },
  {
    path: '/article',
    component: Article,
  },
]
```

创建路由开发环境

使用路由我们还是采用CRA创建项目的方式进行基础环境配置

1. 创建项目并安装所有依赖

```
npx create-react-app react-router-pro
npm i
```

2. 安装最新的 ReactRouter包

```
npm i  react-router-dom
```

3. 启动项目

```
npm run start
```

快速开始

需求：创建一个可以切换登录页和文章页的路由系统

/login

登录

/article

文章

src/main.jsx

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import {
  createBrowserRouter,
  RouterProvider,
} from "react-router-dom";
import "./index.css";

const router = createBrowserRouter([
  {
    path: "/",
    element: <div>Hello world!</div>,
  },
]);

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

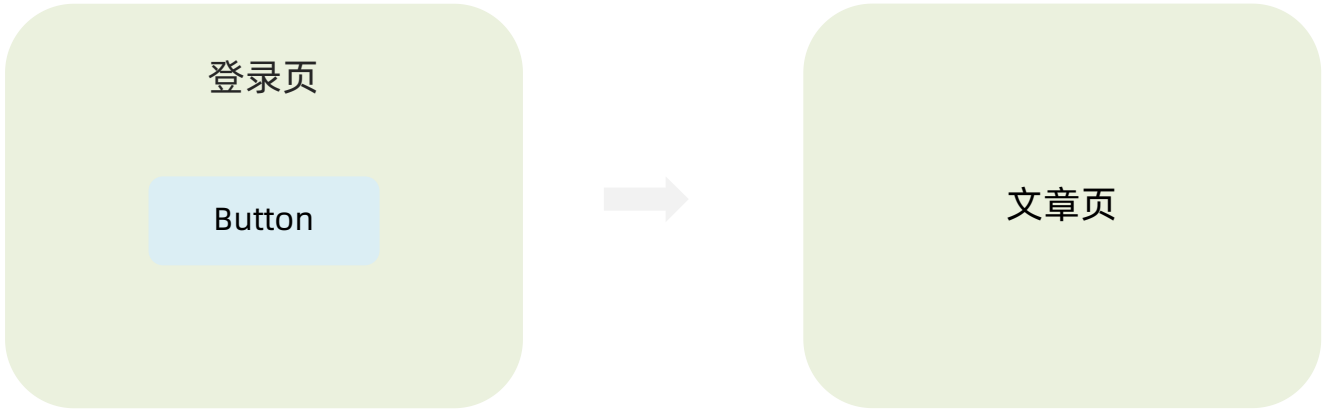
ReactRouter – 抽象路由模块

实际开发中的router配置

ReactRouter – 路由导航

什么是路由导航

路由系统中的多个路由之间需要进行路由跳转，并且在跳转的同时有可能需要传递参数进行通信



声明式导航

声明式导航是指通过在模版中通过 `<Link/>` 组件描述出要跳转到哪里去，比如后台管理系统的左侧菜单通常使用这种方式进行

```
<Link to="/article">文章</Link>
```

语法说明：通过给组件的to属性指定要跳转到路由path，组件会被渲染为浏览器支持的a链接，如果需要传参直接通过字符串拼接的方式拼接参数即可

编程式导航

编程式导航是指通过 `useNavigate` 钩子得到导航方法，然后通过调用方法以命令式的形式进行路由跳转，比如想在登录请求完毕之后跳转就可以选择这种方式，更加灵活

```jsx
import { useNavigate } from "react-router-dom"

const Login = () => {
  const navigate = useNavigate()
  return (
    <div>
      我是登录页
      <button onClick={() => navigate('/article')}>跳转至文章</button>
    </div>
  )
}
```

语法说明：通过调用navigate方法传入地址path实现跳转

ReactRouter – 导航传参

路由导航传参

searchParams 传参

```
navigate('/article?id=1001&name=jack')
```

```
const [params] = useSearchParams()
let id = params.get('id')
```

params 传参

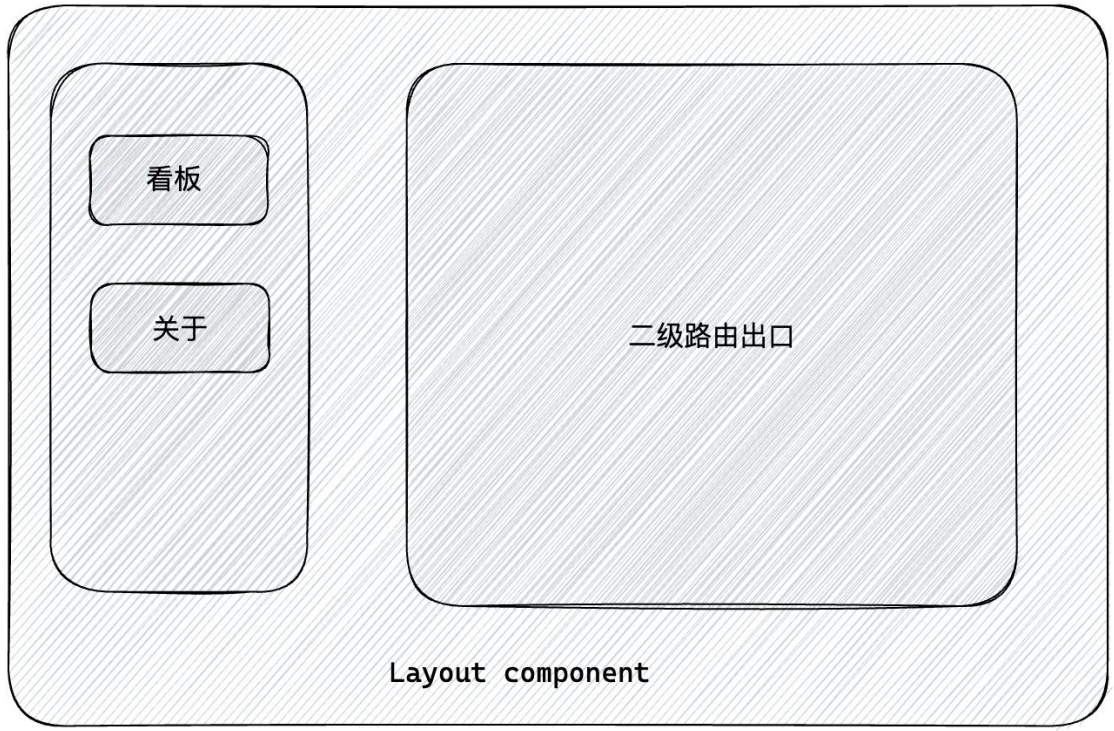
```
navigate('/article/1001')
```

```
const params = useParams()
let id = params.id
```

ReactRouter – 嵌套路由配置

什么是嵌套路由

在一级路由中又内嵌了其他路由，这种关系就叫做嵌套路由，嵌套至一级路由内的路由又称作二级路由，例如：



嵌套路由配置

实现步骤：

1. 使用 children 属性配置路由嵌套关系
2. 使用 `<Outlet/>` 组件配置二级路由渲染位置

```
{
    path: '/',
    element: <Layout />,
    children: [
      {
        path: 'board',
        element: <Board />,
      },
      {
        path: 'about',
        element: <About />,
      },
    ],
  },
```

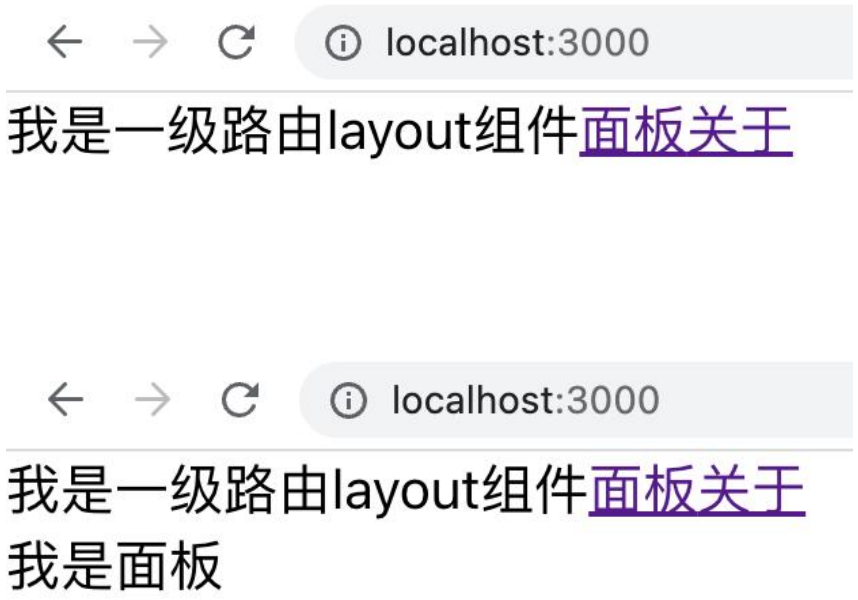
```
const Layout = () => {
  return (
    <div>
      <div>我是Layout</div>
      <Link to="/board">面板</Link>
      <Link to="/about">关于</Link>

      {/* 二级路由出口 */}
      <Outlet />
    </div>
  )
}
```

ReactRouter – 默认二级路由

场景和配置方式

当访问的是一级路由时，默认的二级路由组件可以得到渲染，只需要在二级路由的位置去掉path，设置index属性为true




```
children: [
  {
    index: true,
    element: <Board />,
  },
  {
    path: 'about',
    element: <About />,
  },
],
```

ReactRouter – 404路由配置

404路由

场景：当浏览器输入url的路径在整个路由配置中都找不到对应的 path，为了用户体验，可以使用 404 兜底组件进行渲染

实现步骤：

1. 准备一个NotFound组件
2. 在路由表数组的末尾，以*号作为路由path配置路由

```
const NotFound = () => {
  // 自定义模版
  return <div>this is NotFound</div>
}

export default NotFound
```

```
{
  path: '*',
  element: <NotFound />
}
```

ReactRouter – 俩种路由模式

俩种路由模式

各个主流框架的路由常用的路由模式有俩种，history模式和hash模式, ReactRouter分别由 createBrowerRouter 和 createHashRouter 函数负责创建

路由模式	url表现	底层原理	是否需要后端支持
history	url/login	history对象 + pushState事件	需要
hash	url/#/login	监听 hashChange事件	不需要

传智教育旗下高端IT教育品牌