hooks && Router

本节课重点内容

函数式组件

- 函数式组件，本质就是一个常规函数，接收一个参数 props 并返回一个 reactElement
- 函数式组件中没有this和生命周期函数
- 使用函数式组件时，应该尽量减少在函数中声明子函数，否则，组件每次更新时都会重新创建这个函数

React hooks(钩子)

React hooks 是React 16.8中的新增功能。它们使您无需编写类即可使用状态和其他React功能

常用 hook

- useState
  `const [state, setState] = useState(initialState);`
  let [状态,修改该状态的方法] = useState(初始值);
  1. 在同一个组件中可以使用 useState 定义多个状态
  2. 注意 useState 返回的 setState 方法，不会进行对象合并
  3. 注意 useState 返回的 setState 方法同样是异步方法
- useEffect
  类组件
  componentDidMount、componentDidUpdate 和 componentWillUnmount
  需要清除的副作用
- useRef
  用户关联原生DOM节点，或者用来记录组件更新前的一些数据

React Hooks 优势

- 简化组件逻辑
- 复用状态逻辑
- 无需使用类组件编写

Hook 使用规则

- 只在 React 函数中调用 Hook
  - React 函数组件中
  - React 自定义 Hook 中
- 只在最顶层使用 Hook

路由

路由：根据不同的url规则，给用户展示不同的视图(页面)
当应用变得复杂的时候，就需要分块的进行处理和展示，传统模式下，我们是把整个应用分成了多个页面，然后通过 URL 进行连接。但是这种方式也有一些问题，每次切换页面都需要重新发送所有请求和渲染整个页面，不止性能上会有影响，同时也会导致整个 JavaScript 重新执行，丢失状态。

SPA

Single Page Application : 单页面应用，整个应用只加载一个页面（入口页面），后续在与用户的交互过程中，通过 DOM 操作在这个单页上动态生成结构和内容

优点：

- 有更好的用户体验（减少请求和渲染和页面跳转产生的等待与空白），页面切换快
- 重前端，数据和页面内容由异步请求（AJAX）+ DOM 操作来完成，前端处理更多的业务逻辑

缺点：

- 首次进入处理慢
- 不利于 SEO

SPA 的页面切换机制

虽然 SPA 的内容都是在一个页面通过 JavaScript 动态处理的，但是还是需要根据需求在不同的情况下分内容展示，如果仅仅只是依靠 JavaScript 内部机制去判断，逻辑会变得过于复杂，通过把 JavaScript 与 URL 进行结合的方式：JavaScript 根据 URL 的变化，来处理不同的逻辑，交互过程中只需要改变 URL 即可。这样把不同 URL 与 JavaScript 对应的逻辑进行关联的方式就是路由，其本质上与后端路由的思想是一样的。

```
/*
   http//:www.baidu.com:80/search#hans?name=lll&age=8
   : 后 端口
   # 后 hash
   ？ 后 search
*/
```

前端路由

前端路由只是改变了 URL 或 URL 中的某一部分，但一定不会直接发送请求，可以认为仅仅只是改变了浏览器地址栏上的 URL 而已，JavaScript 通过各种手段处理这种 URL 的变化，然后通过 DOM 操作动态的改变当前页面的结构

- URL 的变化不会直接发送 HTTP 请求
- 业务逻辑由前端 JavaScript 来完成

目前前端路由主要的模式：

- 基于 URL Hash 的路由
- 基于 HTML5 History API 的路由
  https://developer.mozilla.org/zh-CN/docs/Web/API/History_API

React Router

理解了路由基本机制以后，也不需要重复造轮子，我们可以直接使用 React Router 库
https://reacttraining.com/react-router/

React Router 提供了多种不同环境下的路由库

- web
- native

基于 Web 的 React Router

基于 web 的 React Router 为：react-router-dom

安装

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

组件

BrowserRouter 组件 -- history

基于 HTML5 History API 的路由组件

HashRouter 组件 -- hash

基于 URL Hash 的路由组件

Route 组件

通过该组件来设置应用单个路由信息，Route 组件所在的区域就是就是当 URL 与当前 Route 设置的 path 属性匹配的时候，后面 component 将要显示的区域

exact

exact 属性表示路由使用 精确匹配模式，非 exact 模式下 '/' 匹配所有以 '/' 开头的路由

Link 组件

Link 组件用来处理 a 链接 类似的功能（它会在页面中生成一个 a 标签），但设置这里需要注意的，react-router-dom 拦截了实际 a 标签的默认动作，然后根据所有使用的路由模式（Hash 或者 HTML5）来进行处理，改变了 URL，但不会发生请求，同时根据 Route 中的设置把对应的组件显示在指定的位置

to 属性

to 属性类似 a 标签中的 href

传递 props

```
<Route exact path='/' component={Home}
```

如果 Route 使用的是 component 来指定组件，那么不能使用 props

Route : render

```
<Route exact path='/' render={() => <Home items={this.state.items} />} />
```

通过 render 属性来指定渲染函数，render 属性值是一个函数，当路由匹配的时候指定该函数进行渲染

NavLink 组件

NavLink 与 Link 类似，但是它提供了两个特殊属性用来处理页面导航

activeStyle

当当前 URL 与 NavLink 中的 to 匹配的时候，激活 activeStyle 中的样式

activeClassName

与 activeStyle 类似，但是激活的是 className

isActive

默认情况下，匹配的是 URL 与 to 的设置，通过 isActive 可以自定义激活逻辑，isActive 是一个函数，返回布尔值

Switch 组件

该组件只会渲染首个被匹配的组件

Redirect 组件

to

设置跳转的 URL

路由传参

withRouter 组件

如果一个组件不是路由绑定组件，那么该组件的 props 中是没有路由相关对象的，虽然我们可以通过传参的方式传入，但是如果结构复杂，这样做会特别的繁琐。幸好，我们可以通过 withRouter 方法来注入路由对象

hooks

- useHistory
- useLocation
- useParams
- useRouteMatch

下节课内容

- React-router-dom
  - 路由参数
  - 动态路由
  - NavLink
  - 重定向
  - Switch
  - widthRouter
  - Route hooks

练习

- 基于给定视图和react-route-dom 实现一个单页应用