爱创课堂前端培训

# 名称

第6天课堂笔记（本课程共10天）

班级：北京前端训练营13期

讲师：张容铭

日期：2018年7月13日

张容铭老师

微博：@张容铭\_YYQH

QQ : 286031482

E-mail : yuye\_qinghe@qq.com

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

路由

我们使用路由的目的是为了实现单页面应用程序（SPA），

一个完整的url分成几个部分

协议，域名，端口号，路径，文件，搜索词，哈希

只有hash的改变，不会导致浏览器端想服务器端发送新的请求，所以前端的路由是基于hash实现的

在React中，版本更新变化是比较大的，因此每一个版本都有一套对应的路由

React为了实现特点3（多端适配），因此将路由做了拆分，来适应各个端

https://github.com/ReactTraining/react-router

在react中使用路由分三步

第一步 在应用程序组件中，定义路由渲染容器，通过Switch组件定义

在里面通过Route组件定义每一条规则

name表示路由名称

component表示渲染的组件

path 表示路由规则

通过/将路由分成两个部分

默认是不变的（静态路由）（一对一的）

我们可以通过在模块名称前面添加:定义动态路由（一对多的）

可以通过?定义query路由等等

如果想匹配所有的，我们用\*表示（由于\*匹配的比较广，所以我们通常定义在最后面）

第二步 选择路由渲染策略，渲染应用程序组件

React16版本提供了两种侧路

一种是基于path的改变实现的路由，叫BrowserRouter，需要后端配合

一种是基于hash的改变实现的路由，叫HashRouter，前端独立完成

第三步 渲染第二步得到的结果（路由规则）

路由模块

BrowserRouter 基于path的改变，实现的路由策略

HashRouter 基于hash的改变，实现的路由策略

Link 路由导航组件（实现页面切换的，默认渲染成a标签）

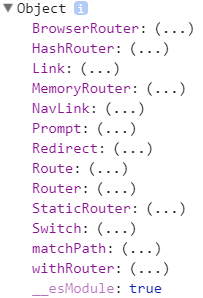
MemoryRouter 内存路由策略

NavLink 导航链接

Redirect 重定向组件

Route 定义每一条规则的

Switch 路由容器



路由信息

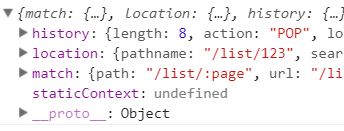
路由渲染的组件会添加一些信息，我们可以通过组件的属性数据获取他们

路由子组件是不具备这些信息的，想拥有这些信息，我们可以传递属性数据

history react路由内部维护的history对象

location router路由模拟location对象，存储自身的路由信息（hash模块的切割）

match 路由匹配的数据



路由中使用redux

我们通过Provider组件渲染页面DealApp，为了传递store，

路由通过HashRouter|BrowserRouter渲染组件

1 所以我们HashRouter|BrowserRouter放在Provider中即可

2 在HashRouter|BrowserRouter组件中，通过Route渲染DealApp组件

想绑定路由调度者，我们可以使用redxu提供的combineReducers方法实现绑定

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |