# react.js路由详解

#### 大纲

#### 课程内容

web应用的路由

前端路由的实现方式

vue-router

react-router

知道http服务的基本概念,了解页面请求后服务端返回的过程了解现代前端web基础知识,如es6、react、vue、redux等了解es6模块化的基础知识

# 大纲

- web应用路由的实现概述
  - ♂后端路由(MVC)
  - 前端路由
- 前端路由的实现方式
  - hashHistory
  - history
- vue-router
  - 实现原理
  - 使用方式
- react-router
  - 实现原理
  - 使用方式
- 从0到1实现简单react-router的实现(源码)
  - 模块设计
  - Context实现
  - Provider实现
  - BrowserRouter
  - Route
  - Link





# 课程内容

### web应用的路由

• 路由的概念

JavaScript 口复制代码

- 1 https://news.163.com/world/index.html
- 2 https://xxx.163.com/xx/index.php
- 3 ...

#### 访问过程为

- 1. 浏览器发出请求
- 2. 服务器监听到80端口(或443)有请求过来,并解析url路径
- 3. 根据服务器的路由配置,返回相应信息(可以是 html 字串,也可以是 json 数据,图片等)
- 4. 浏览器根据数据包的`Content-Type`来决定如何解析数据

简单来说路由就是用来跟后端服务器进行交互的一种方式,通过不同的路径,来请求不同的资源,请求不同的页面是路由的其中一种功能。

- 前端路由的诞生
  - ajax的出现
  - 前端框架的演变

## 前端路由的实现方式

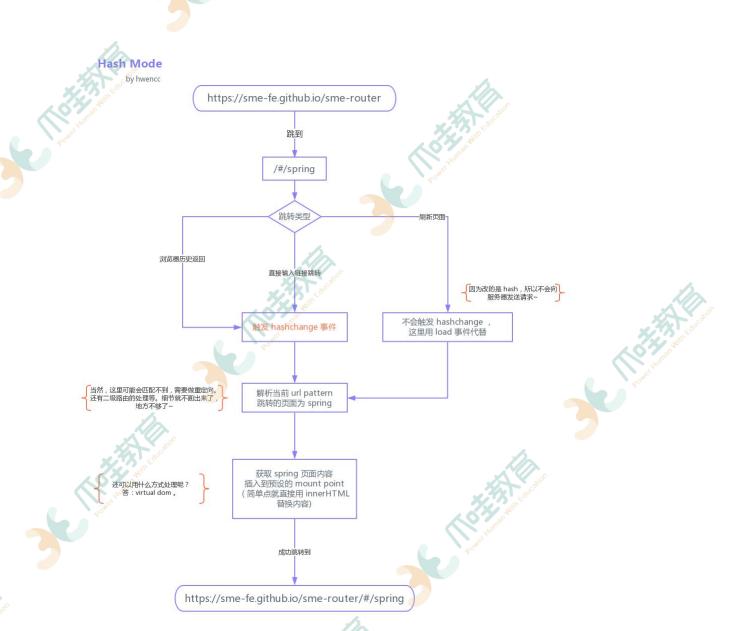
后端通过固定入口返回静态文件模板,前端在模板上构建大型spa应用路由上,通过检测 url 的变化,截获 url 地址,然后解析来匹配路由规则。

• hash路由的实现(2014年以前)

JavaScript 口复制代码

1 https://www.xxx.com/home/#my





• history的实现(2014年后)

14年后,因为HTML5标准发布。多了两个 API,`pushState` 和 `replaceState`,通过这两个 API 可以改变 url 地址且不会发送请求。同时还有 `onpopstate` 事件。通过这些就能用另一种方式来实现前端路由了



**HTML5 Mode** by hwencc https://sme-fe.github.io/sme-router 跳到 /spring 跳转类型 通过浏览器历史返回 刷新页面或直接输入链接跳转 通过 pushState 跳转 会向浏览器发送请求 , 所以需要服务器做重定向。 然后触发 load 事件 pushState 会改变 url 且不会向浏览 触发 popstate 事件 器发送请求 解析当前 url pattern 跳转的页面为 spring 从这里开始的流程,跟 hash mode 就 相差无几了 获取 spring 页面内容 插入到顶设的 mount point (简单点就直接用 innerHTML 替换内容) 成功跳转到 https://sme-fe.github.io/sme-router/spring

## vue-router

- 实现原理
  - hash模式(/dev/src/history/hash.js)

```
JavaScript
                                                                    C
复制代码
     ☀ 添加 url hash 变化的监听器
     */
     setupListeners () {
     const router = this.router
 6
 7
     /**
      * 每当 hash 变化时就解析路径
 8
      * 匹配路由
9
      */
10
     window.addEventListener('hashchange', () => {
11
12
       const current = this.current
13
       /**
14
        * transitionTo:
15
        * 匹配路由
        * 并通过路由配置,把新的页面 render 到 ui-view 的节点
16
17
       this.transitionTo(getHash(), route => {
18
         replaceHash(route.fullPath)
19
20
21
22
```

■ history模式(/dev/src/history/html5.js)

```
export class HTML5History extends History {
     constructor (router, base) {
       super(router, base)
 6
        * 原理还是跟 hash 实现一样
        * 通过监听 popstate 事件
 8
        * 匹配路由, 然后更新页面 DOM
 9
       window.addEventListener('popstate', e => {
10
11
         const current = this.current
12
13
         // Avoiding first popstate` event dispatched in some browsers but
     first
14
         // history route not updated since async guard at the same time.
         const location = getLocation(this.base)
15
         if (this.current === START && location === initLocation)
16
17
           return
18
         }
19
         this.transitionTo(location, route => {
20
21
           if (supportsScroll) {
             handleScroll(router, route, current, true)
22
23
24
         })
25
       })
26
     }
27
28
     go (n) {
29
       window.history.go(n)
30
31
     push (location, onComplete, onAbort) {
32
33
       const { current: fromRoute } = this
       this.transitionTo(location, route => {
34
         // 使用 pushState 更新 url, 不会导致浏览器发送请求, 从而不会刷新页面
35
         pushState(cleanPath(this.base + route.fullPath))
36
37
         onComplete && onComplete(route)
38
       }, onAbort)
39
     }
40
41
     replace (location, onComplete, onAbort) {
42
      const { current: fromRoute } = this
43
       this.transitionTo(location, route => {
44
         // replaceState 跟 pushState 的区别在于,不会记录到历史栈
         replaceState(cleanPath(this.base + route.fullPath))
45
46
         onComplete && onComplete(route)
47
       }, onAbort)
```

#### **使用方式文档**

#### react-router

- react-router-dom 和 react-router
- react-router-dom使用方式文档
- react-router实现原理

