# vue-router 路由（掌握）

我们要实现**单页应用**程序，所以我们要学习路由。

## spa

single-page application，单页应用。一个应用程序只有一个页面。

单页 Web 应用 (single-page application 简称为 SPA) 是一种特殊的 Web 应用。它将所有的活动局限于一个Web页面中，仅在该Web页面初始化时加载相应的HTML、JavaScript 和 CSS。一旦页面加载完成了，SPA不会因为用户的操作而进行页面的重新加载或跳转。而是利用 JavaScript 动态的变换HTML的内容（通过div切换显示和隐藏，或者是数据的渲染），从而实现UI与用户的交互。由于避免了页面的重新加载，SPA 可以提供较为流畅的用户体验。得益于ajax，我们可以实现无跳转刷新，又多亏了浏览器的histroy机制，我们用hash的变化从而可以实现推动界面变化。

（php中，单入口程序）

优点：提供流畅的用户体验，避免了不必要的跳转和重新渲染，服务器压力小。

缺点：不利于seo; 初次加载页面更耗时; 历史管理需要编程实现

传统的网页：地址各不一样

超市

<https://pro.m.jd.com/mall/active/2hqsQcyM5bEUVSStkN3BwrBHqVLd/index.html>

牛奶

https://pro.m.jd.com/mall/active/3nxcGU8mCsjXN5MjTGQ1AX2LbdfF/index.html

Spa:小米主页

<https://m.mi.com/#/index>

<https://m.mi.com/#/product/view?product_id=2172300011>

https://m.mi.com/#/product/view?product\_id=10000061

## spa的实现

页面只有一个，但是，你要根据用户的地址栏中的信息去展示不同的内容给用户，怎么办？

两种方法：

* 动态组件
* 路由

### 动态组件

动态组件的功能有限（例如，不能传参），我们可以去使用另一种方法--路由。

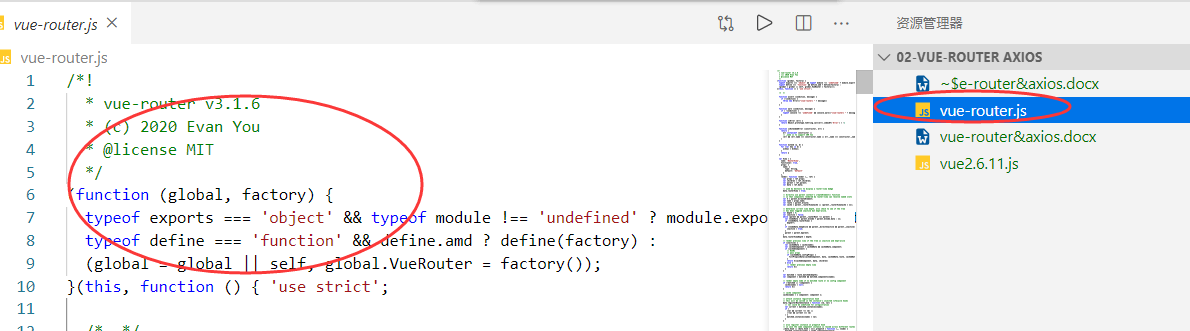
## vue-router基本使用

Vue.js中并没有提供路由的功能 ，靠vue-router来实现。Vue-router是一个插件。

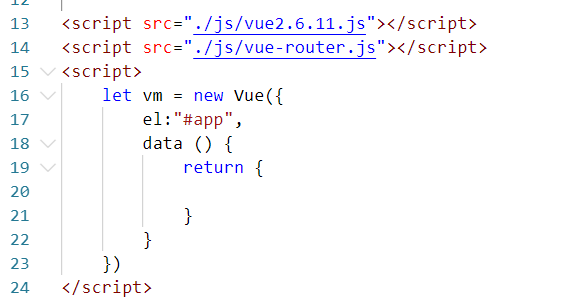
官网：<https://router.vuejs.org/zh/>

Vue Router 是 [Vue.js](http://cn.vuejs.org/" \t "https://router.vuejs.org/zh/_blank) 官方的路由管理器。它和 Vue.js 的核心深度集成，让构建单页面应用变得易如反掌。

它是一个单独的js文件。需要我们下载，或使用CDN，我就下载了：



先下载，然后在页面中引用。必须在vue之后引入。如下：

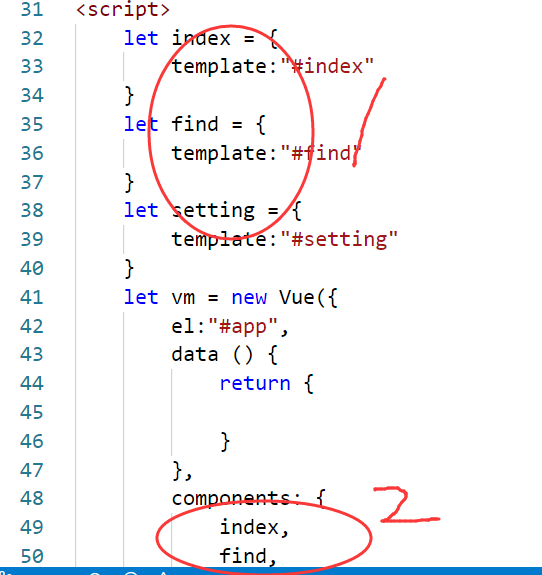


### 准备好组件对象

**组件使用的三步曲：**

* 创建组件对象
* 注册组件
* 使用组件

代码如下：

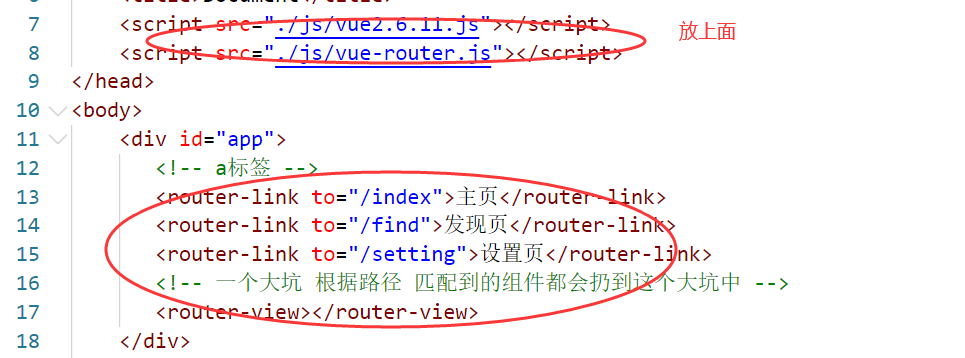




### 用router-link和router-view准备好视图

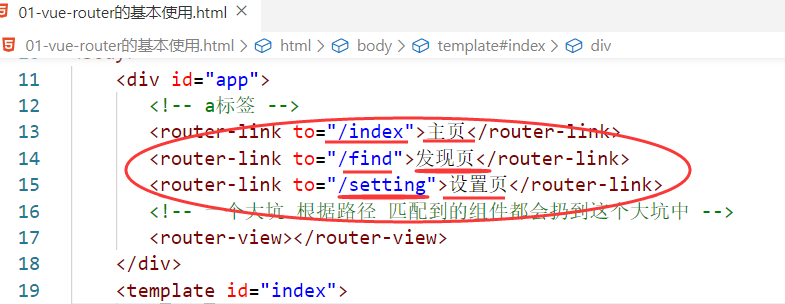
router-link和router-view是vue-router提供好的两个组件，我们可以直接使用。

router-link当成a标签来使用。 router-view就是一个大坑，如果路由匹配上了，对应的组件就扔到了这个大坑中。

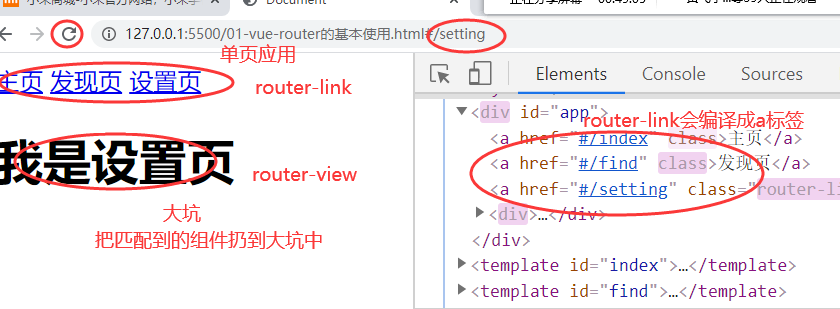


只写上面的代码，还是会报错的。你要使用vur-router，你需要按照人家的规则来写。所谓的框架就是你在合适的位置写在合适的代码，等框架来调用，所以我们继续写：



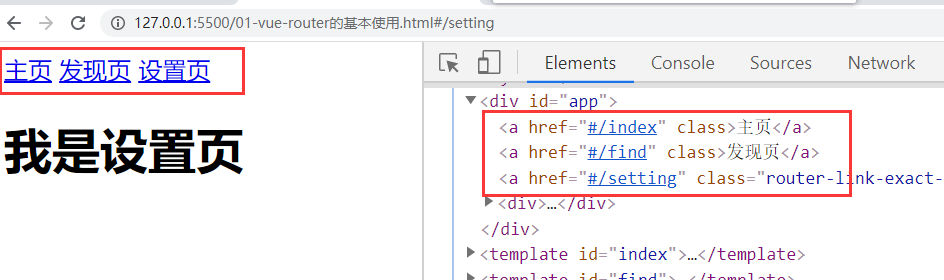


匹配到的组件会被 放到大坑中，如下：



#### router-link

router-link最后，会被编译成a标签，如下 ：



to属性用来指定路由 --- 相当于a标签的**href**属性。它必不可少。

# 暂时不管

**路由分两种**

* 一种是hash路由 有#，比较丑
* 一种是histrory路由

#### router-view

是一个占位符（大坑），用来装当前的组件。

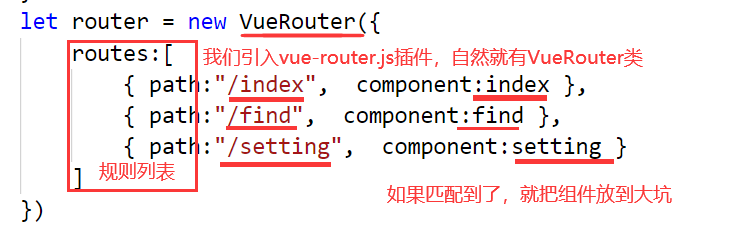
### 实例化 路由对象

let router = new VueRouter({

routes: [ {path:”/”,component:组件1},{path:”/add”,component:组件2}]

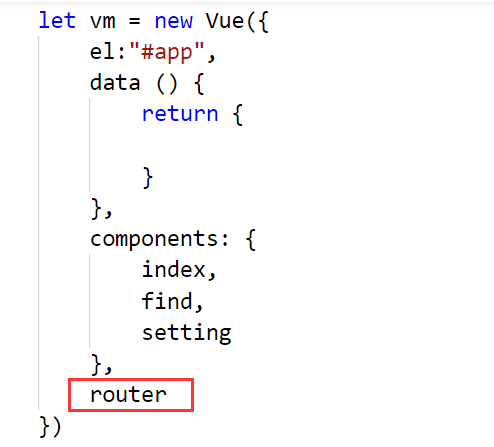
})

配置路由。指定路由和组件之间的关系



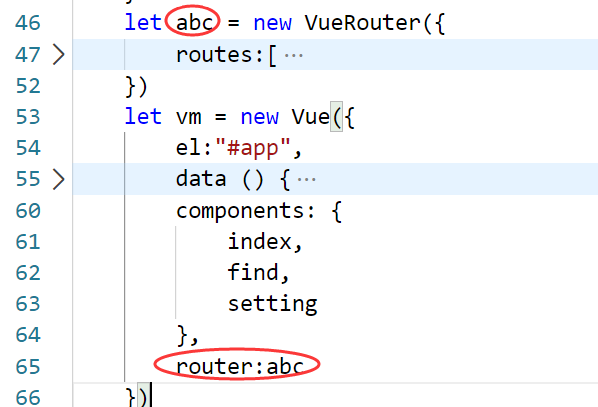
下图中红色部分是路由

### 4.在vue实例中配置router



你要注意，上面的写法是简写方式。相当于：router:router .

如果你前面创建的路由对象不叫router，如下：



则不能简写，必须写全。

### 效果如下：



**总结：**

* 创建一个VueRouter对象：new VueRouter()。
* 在VueRouter中，需要传递一个`routes`参数。这个参数是一个数组类型，数组中存储的是对象，对象中最少要有两个属性，一个是path，代表url，第二个是component，代表数据更新的组件。示例代码如下：

let router = new VueRouter({

routes: [

{path: "/",component: index},

{path: "/find",component: find},

{path: "/friend",component: friend}

]

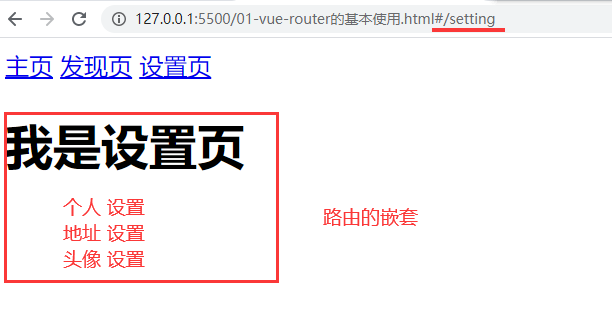
})

* 将router传给Vue。
* 把网页中之前的`a`标签，替换成router-link。
* 使用router-view指定网页中哪个地方要被更新。

## 嵌套路由

### 定义

实际开发中的应用界面，通常由多层嵌套的组件组合而成，如：设置项中还有设置的子项。



### 实现

1. 设置视图 。 在子组件中，**再次使用**router-link和router-view。
2. 修改路由配置。设置相关路由的children项

let router = new VueRouter({

routes: [

{

path:”/setting”,

component:setting,

**children:[ {path:”/setting/set1”,component:子组件} ]**

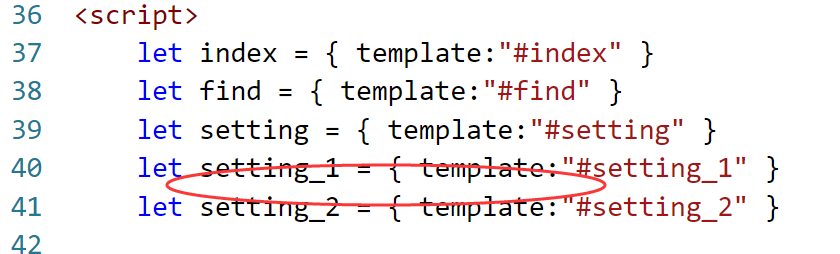
},

})

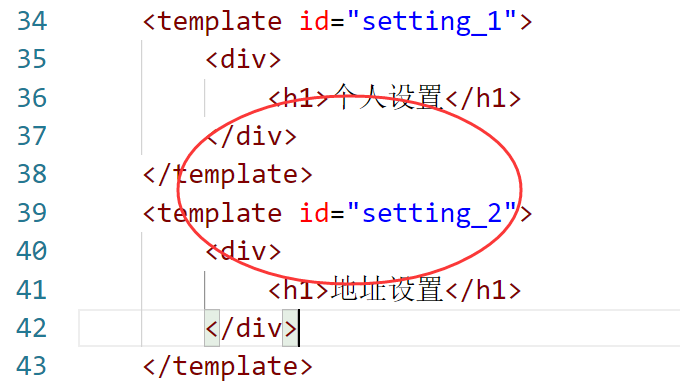
### 示例

在前面的案例的基础上，给设置页，下面再加两个子页。

#### 第一步：设置好子组件。

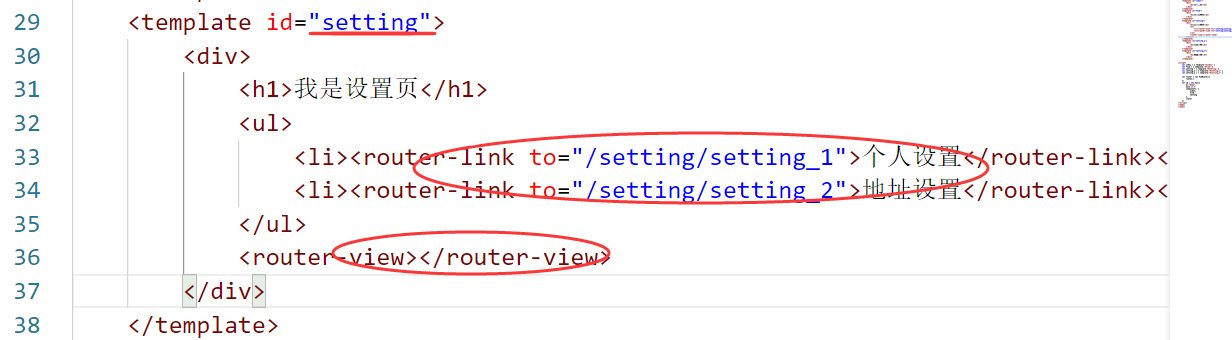


对应的两个template如下：



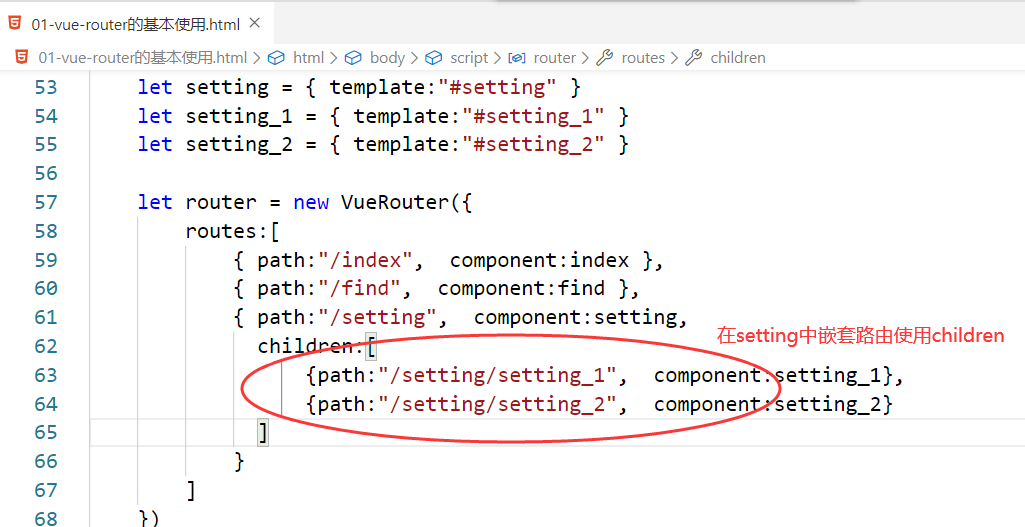
#### 第二步：修改setting组件的视图

因为 这个两个子组件将会嵌套在setting视图中，所以我们要去修改setting组件的视图：加上router-link 和router-view.

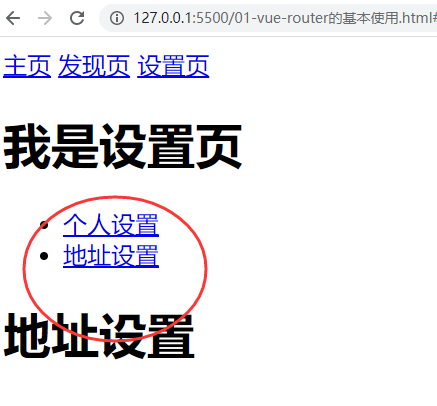


**并没有结束，不要忘记路由的规则，不要忘记路由的规则，不要忘记路由的规则。**

#### 第三步：修改路由配置



#### 效果如下:



如果还有三级的嵌套，做法跟上面是一致的。

**总结：**

* 在大的路由下面，有时候想要使用一些子路由来切换数据。那么这时候可以使用路由嵌套。
* 首先在定义路由的时候，不需要在`routes`中单独添加一个映射。而应该放在父路由的children中：

let router = new VueRouter({

routes: [

{

path: "/",

component: index

},

{

path: "/user/:userid",

component: user,

children: [

{path: "",component: setting},

{path: "setting",component: setting},

{path: "message",component: message}

]

}

]

})

* 在父路由的组件中，要记得添加路由出口<router-view>。示例代码：

<div>

<h1>我的主页</h1>

<ul class="nav nav-tabs">

<li role="presentation" class="active">

<router-link to="/user/123/setting">设置</router-link>

</li>

<li role="presentation">

<router-link to="/user/123/message">消息</router-link>

</li>

</ul>

<router-view></router-view>

</div>

## 动态路由-传递参数

例如：商品详情页：

[https://m.mi.com/#/product/view?product\_id=](https://m.mi.com/#/product/view?product_id=10000022)**[10000022](https://m.mi.com/#/product/view?product_id=10000022)**

https://m.mi.com/#/product/view?product\_id=**6458**

上面两个地址实际上共用一个页面，但由于传递的参数不同，则显示内容也不同。

这种带参数的路由，就是动态路由。

### 实现

可以在 vue-router 的路由路径中使用 动态路径参数。

动态路径参数 **以冒号开头**

{ path: '/detail/:id', component: 组件}

当匹配到一个路由时，参数值会被设置到 this.$route.params.id，可以在每个组件内使用。

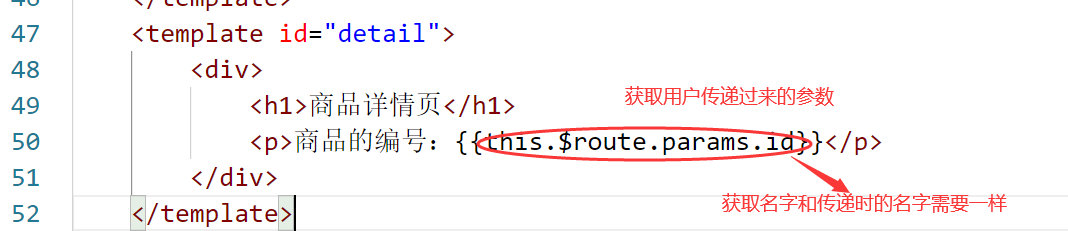
### 示例 -- 只有一个参数

在上面的基础，再加一个显示商品详情的组件detail，它的功能是根据不同的商品的编号，去加载商品的信息，再显示出来。

#### 创建一个组件对象



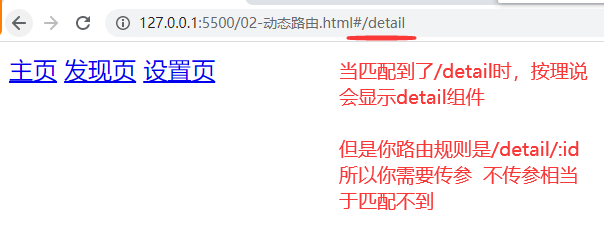
它对应的template如下：

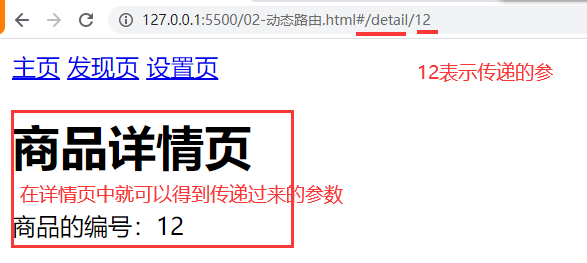


#### 修改路由配置



#### 测试路由

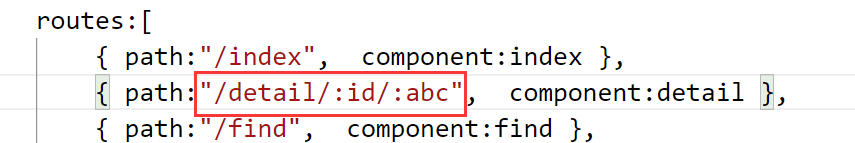


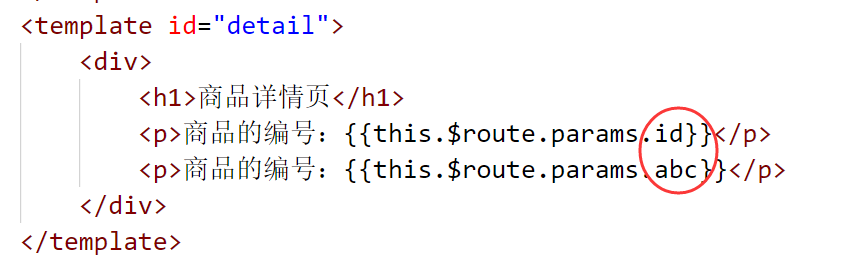




### 示例 -- 多个参数

只需修改相关的路由规则即可：

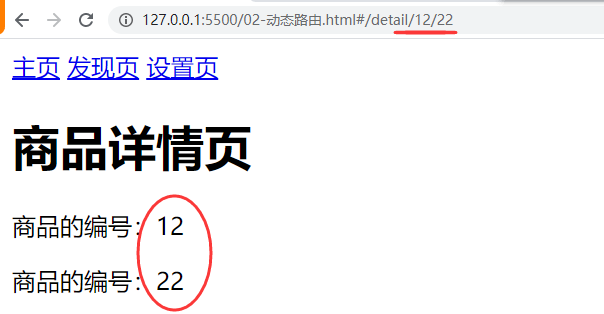




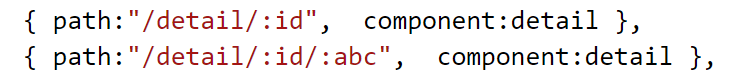
如上：表示detail必须要跟两个参数。如下：

/detail/01/abc 这可以匹配成功 。abc的值就是01，xyz的值就是abc

~~/detail/01 这不能匹配~~



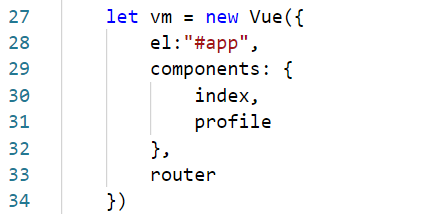
可以把多个规则对应一个组件：



**总结：**

* 在url中，通过定义一个参数，那么以后url中就可以动态的传递这个参数。语法是：/profile/:参数名
* 在组件中，可以通过this.$route.params.参数名拿到，或者是组件的模板中，可以通过$route.params.参数名拿到。
* this.$route和this.$router的区别：
* this.$route：代表的是当前这个路由里的一些信息集合。比如params，query，fullPath等。
* this.$router：代表的是全局的VueRouter对象。

快速再回顾一下：



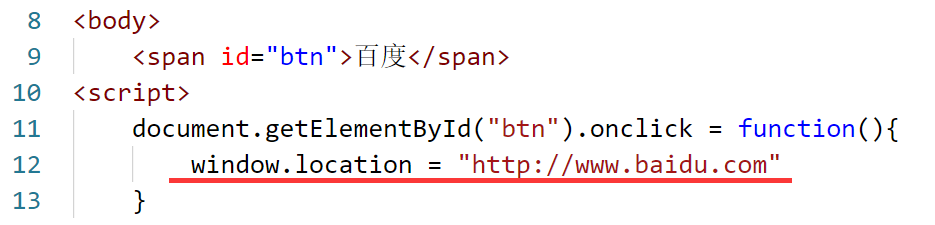
## 编程式路由-实现页面跳转

所谓的编程式路由，就是**写代码**实现的路由。

**对比理解：页面跳转有两种方法**

* a标签可以设置href之后，实现页面跳转。
* 另外，还有一种方法去实现页面跳转：**写代码**window.location = “”

如下：



就可以实现点击跳转了。

编程式路由：就像通过用代码设置location一样，去进行页面的跳转。

### 格式：

在**vue-router**中提供了一个叫router的对象，还有一个叫route的对象，两个是不同的对象。

* route route.params.id 没有r的往往是去拿属性
* router router.push() 有r的往往是后面跟上方法

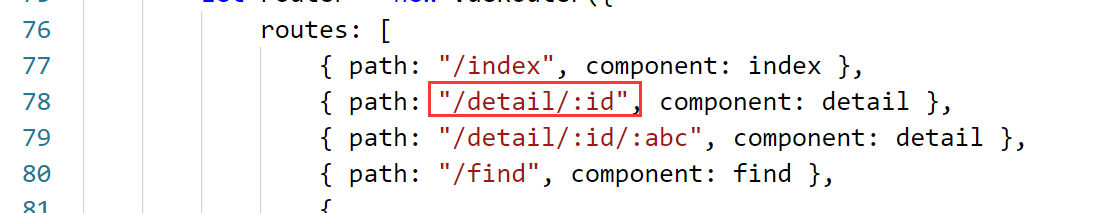
router.push({ path: 'home' })

### 示例

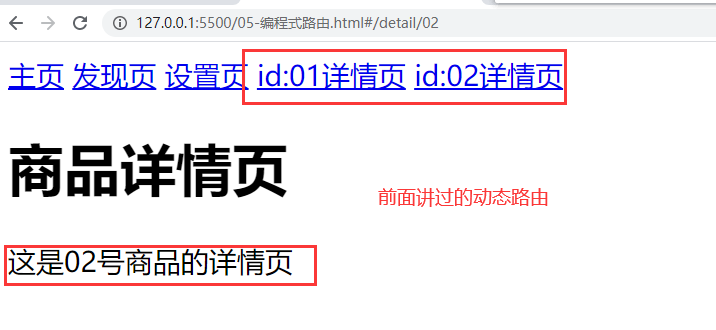
先回顾一下，所谓的动态路由，如下：



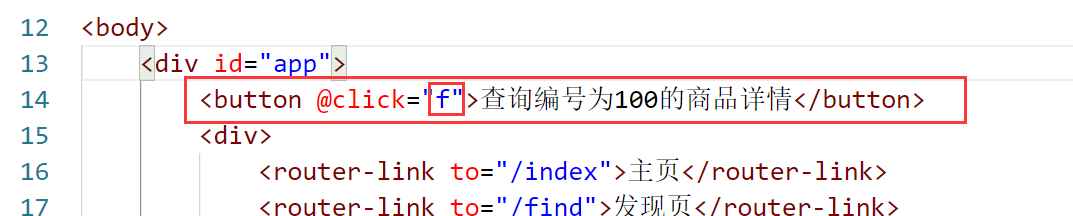
对应的路由规则如下：



效果如下：

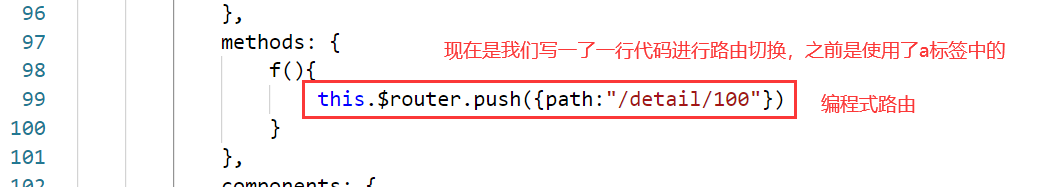


**接着我们换一种思路实现路由，如下：**



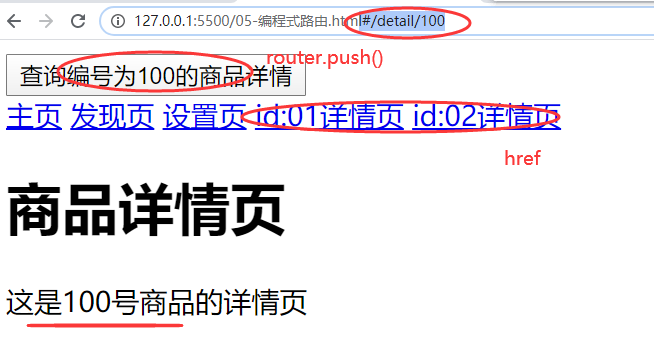
点击上面的按钮，直接打开商品编号为100的详情页。

方法f()如下：



注意：这里是$route**r ，而在获取参数时，** this.$**route**.params.id

效果如下：



**总结：**

* this.$router.push：转到下一个url，会把新转入的url添加到浏览器的history中。push的参数：

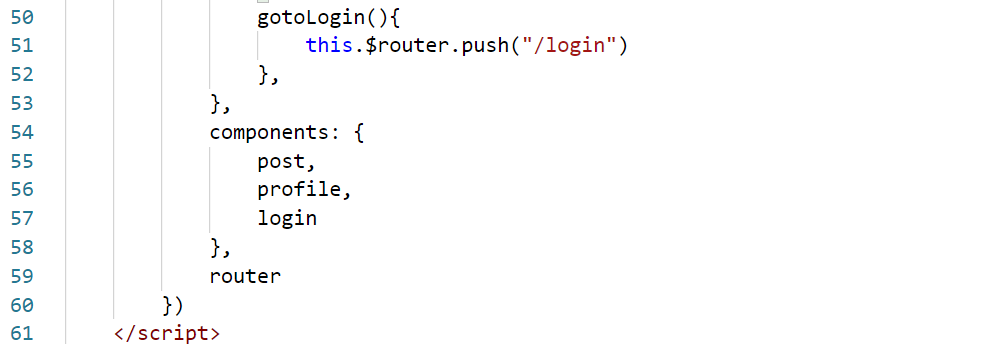
\* 字符串：直接就是路径。

\* 对象：path和name都可以。但是如果使用了path，那么参数就必须要放到path中， 放到params中没有效果。

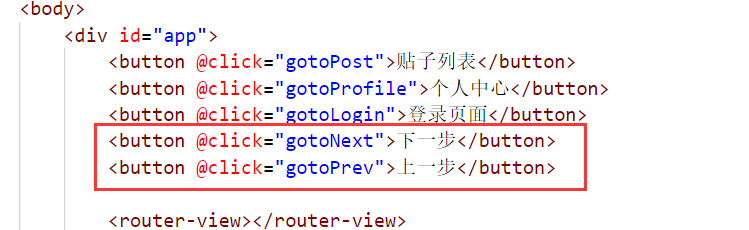
* this.$router.replace：跟push是一样的，只不过是替换当前的页面。
* this.$router.go：往前和往后。

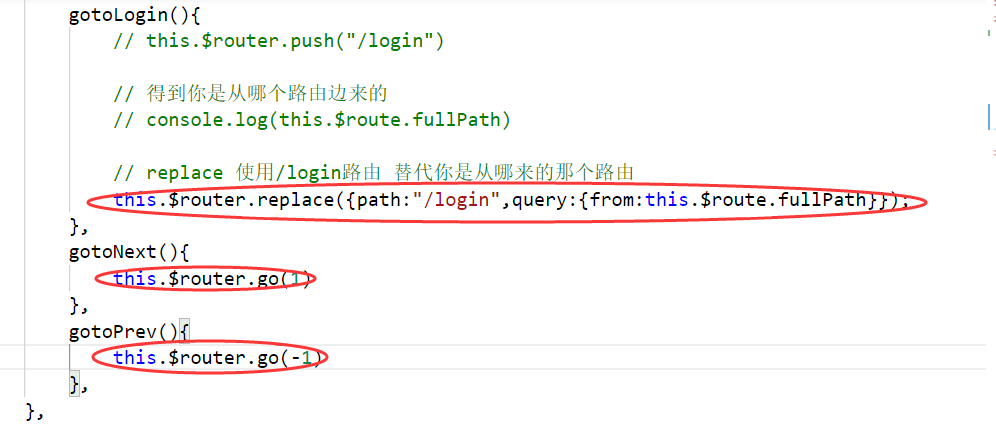
**快速再加强一下：**





下午继续说编程式路由（**通过$router.push()/$router.replace()/$router.go()**）：





## 重定向配置和别名

重定向：当我们在地址栏中访问一个a网址时，会跳转到b网址。

直接通过路由设置来完成：

const router = new VueRouter({

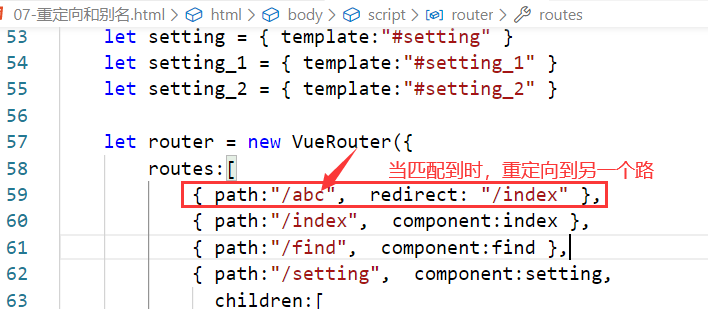
routes: [

{ path: '/a', **redirect**: '/b' }

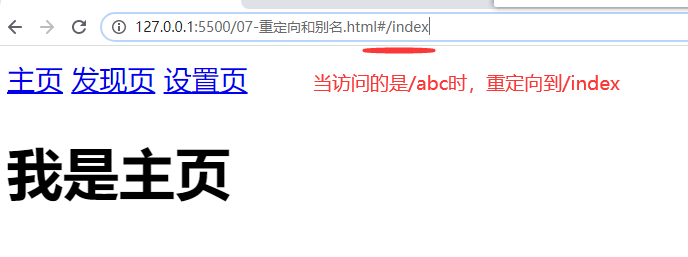
]

})

示例：



如下 ：



别名：在定义路由的时候，可以加一个alias参数，用来表示这个url的别名。以后也可以通过别名来访问到这个组件。

代码如下：

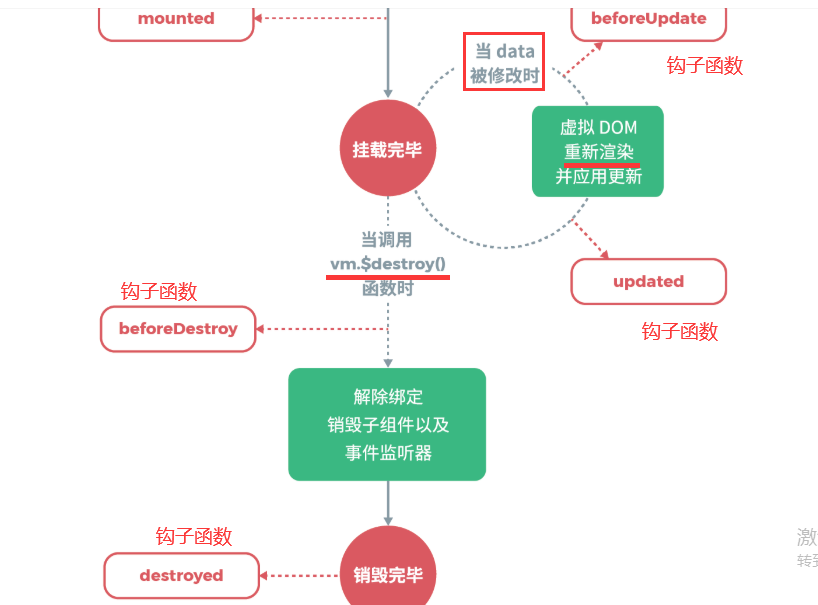
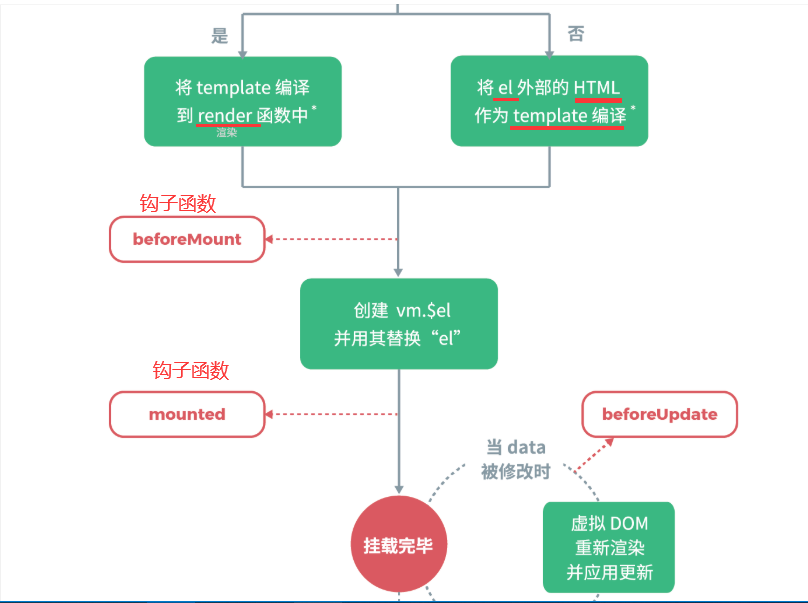
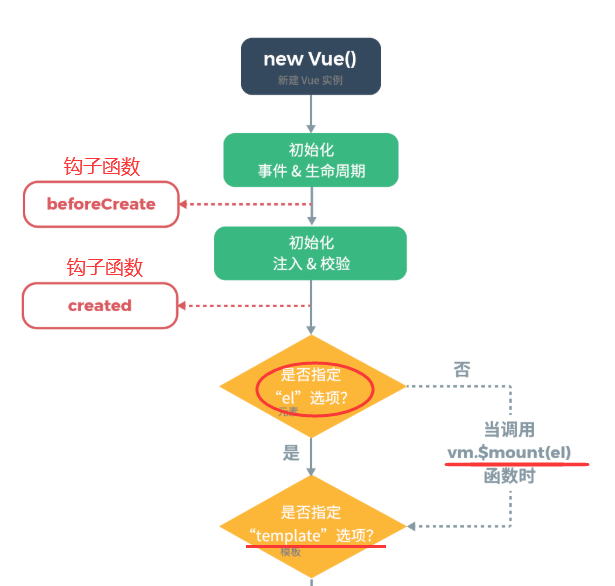


**总结：**

* 重定向：在定义路由的时候，可以加一个redirect参数，用来重定向的到另外一个页面。
* 别名：在定义路由的时候，可以加一个alias参数，用来表示这个url的别名。以后也可以通过别名来访问到这个组件。

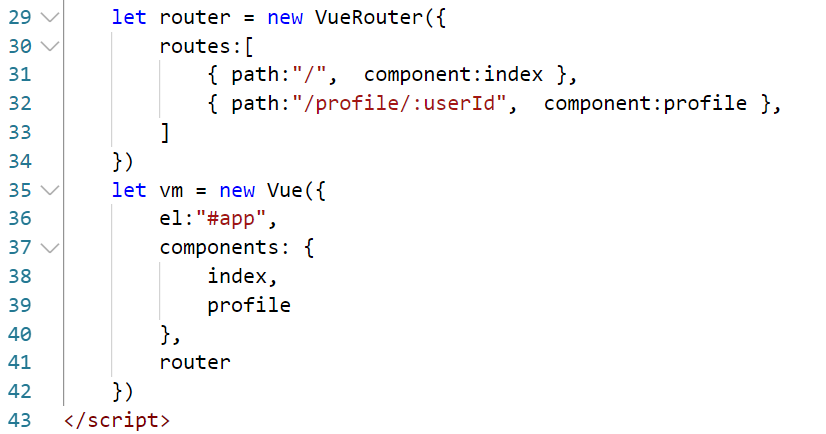
## 组件复用问题

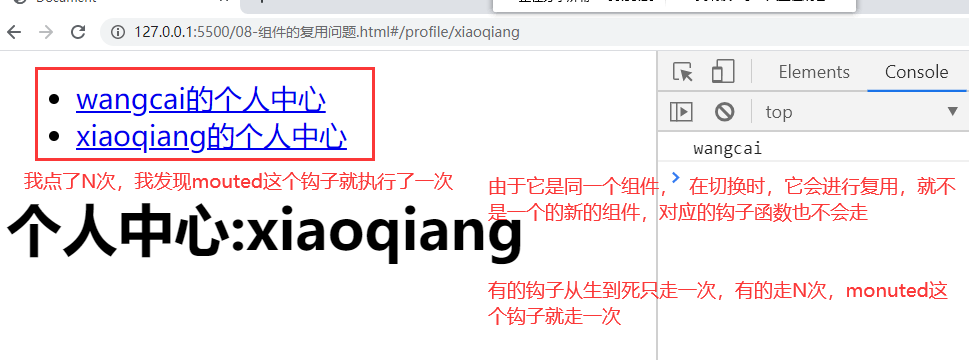
每一个组件都有一些生命周期的钩子函数，不需要我们去调用，vue会在合适的时机自动调用，关于vue组件的生命周期，后面还会细说，现在先了解一下。看一个图：



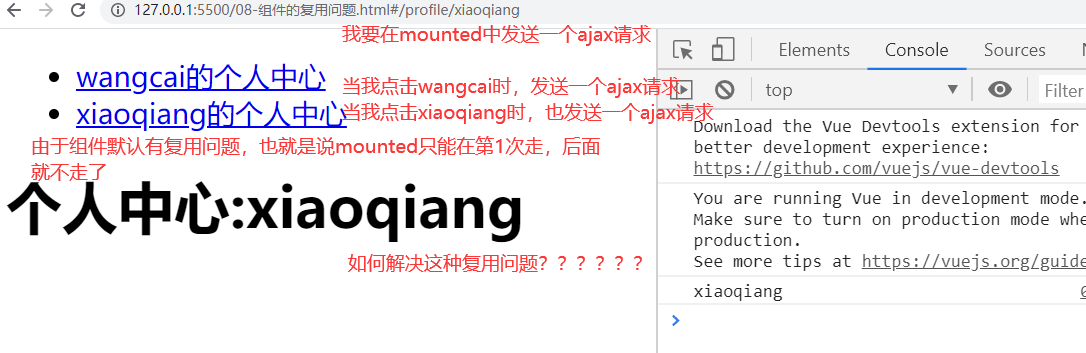
当使用路由参数时，例如从/user/foo导航到/user/bar，原来的组件实例会被复用。因为两个路由都渲染同个组件，比起销毁再创建，复用则显得更加高效。不过，这也意味着组件的生命周期钩子不会再被调用。







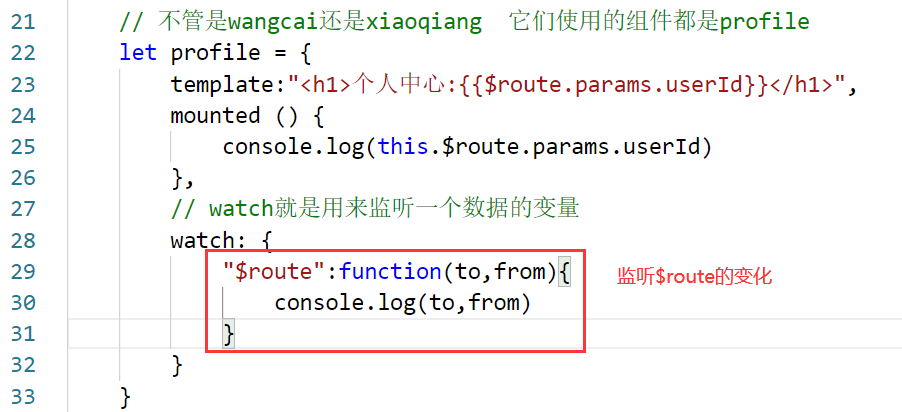
虽然复用可以提升性能，但是有时候不想复用，怎么？？？

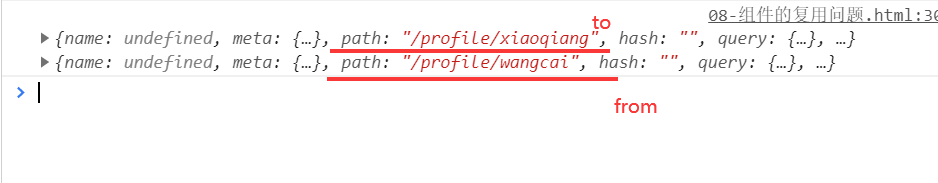


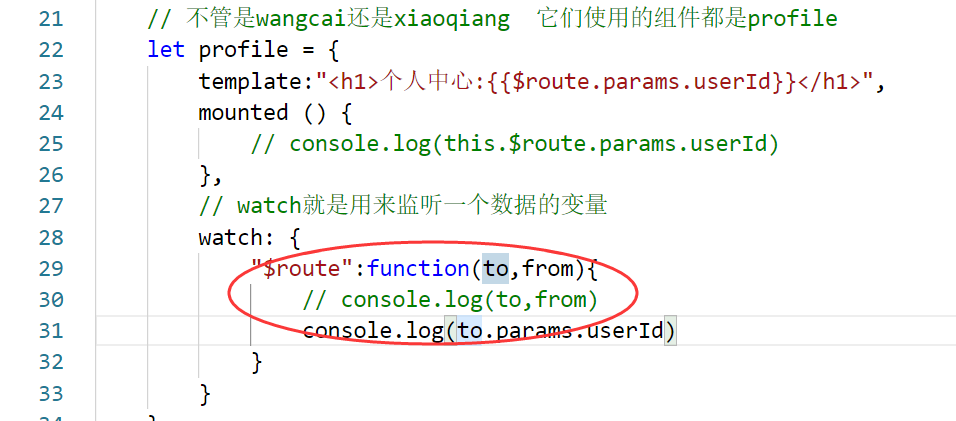
组件复用后，生命周期函数不会被重复调用，那么如果数据更新了，该怎么做相应的处理。有两种解决方案：

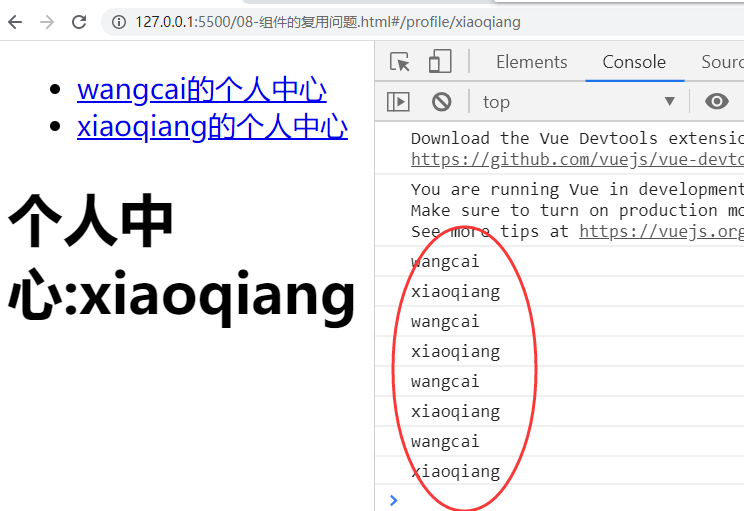
* 监听this.$route属性。通过判断to和from来获取更新的数据。
* 使用导航守卫的beforeRouteUpdate方法，也可以获取to和from，但是这个函数记得调用`next()`，否则页面不会进行更新。

**方案一，监听$route是否变化，有一个to和from，分析代码如下：**

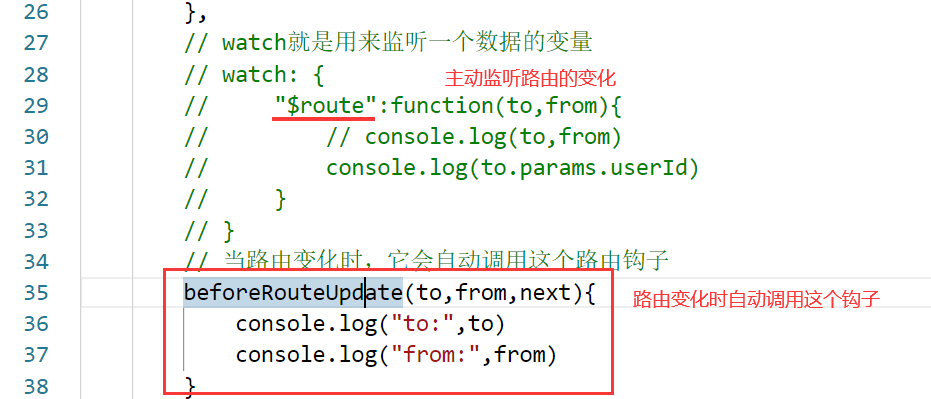


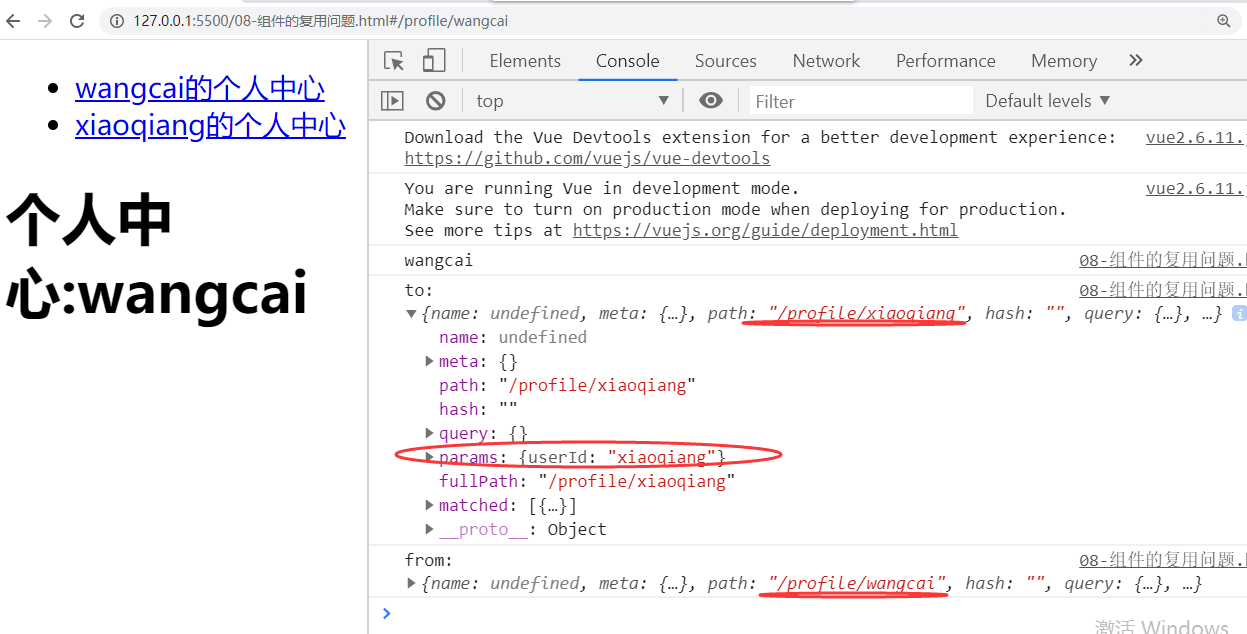




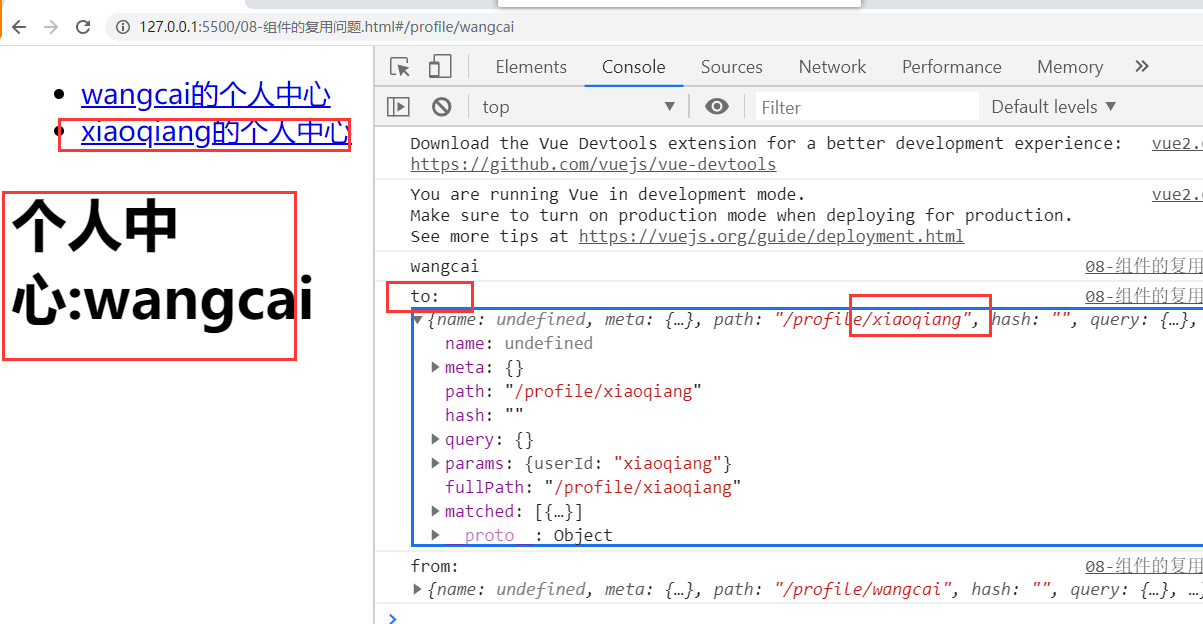


**方案二：使用导航守卫的beforeRouteUpdate**

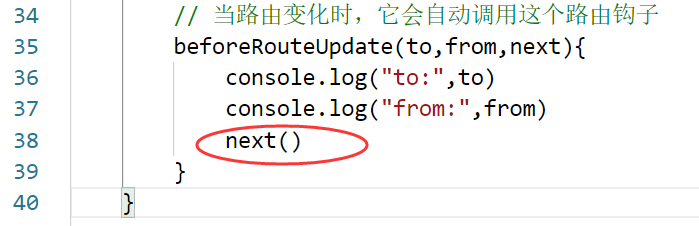




当我们点了xiaoqiang，此时页面中还是wangcai的个人中心，为什么？



原因是在路由的钩子，需要调用next，如下：

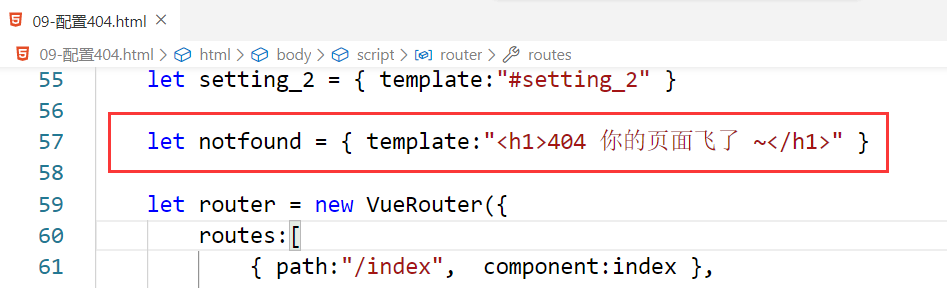


## 404配置

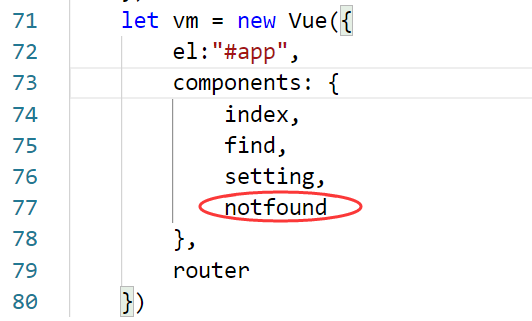
**什么是404配置：**

* 前端的页面配置：在所有路由后面增加一个`\*`的url，让这个url映射到一个404的组件。
* 数据不存在的处理：这种情况，前端是没法判断存不存在的，只能通过访问服务器来判断存不存在。如果服务器返回不存在，那么我们可以通过`this.$router.replace`，跳转到404页面。

创建一个404组件，如下：



注册之：



配置路由规则：



测试如下：



## 命名视图

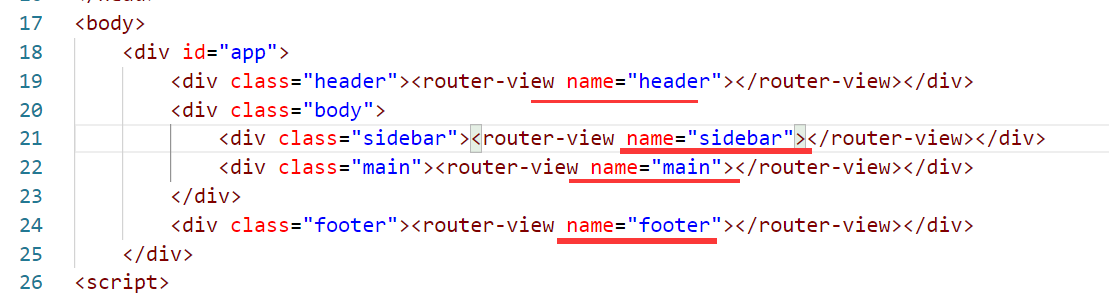
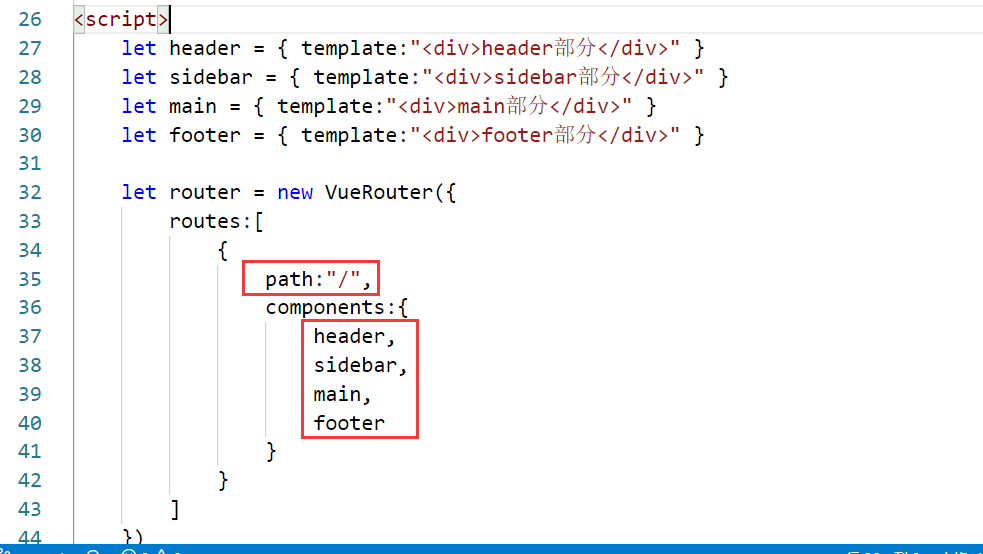
**前面是一个path对应一个组件，能不能让一个path对应多个组件呢？**

答：可以，使用命名视图。

在一个页面中，可以通过命名视图展示多个组件。在实现的时候，有以下几个步骤：

* 在定义路由的时候，需要传递`components`，然后把所有需要展示的路由都放到这个里面。`components`是一个对象，{name:组件}的映射。
* 在模板中，就是通过`<router-view name="组件名"></router-view>`来实现。

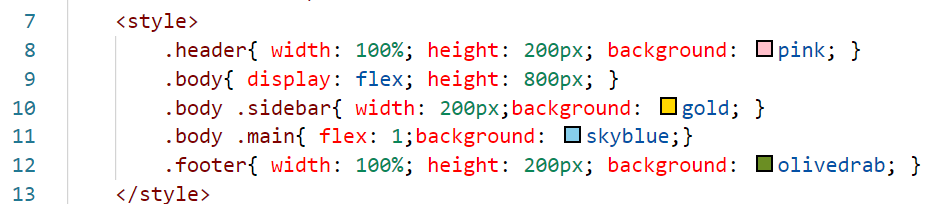
代码如下：



效果：



**样式：**



## 导航守卫之全局导航守卫

**在`VueRouter`上实现的。总体来讲有两个函数，一个是`beforeEach`、`afterEach`。**

* 1. `beforeEach(to,from,next)`：`to`代表的是上一个路由对象，`from`代表的是下一个路由对象。next代表的是控制下一步路由该怎么走。

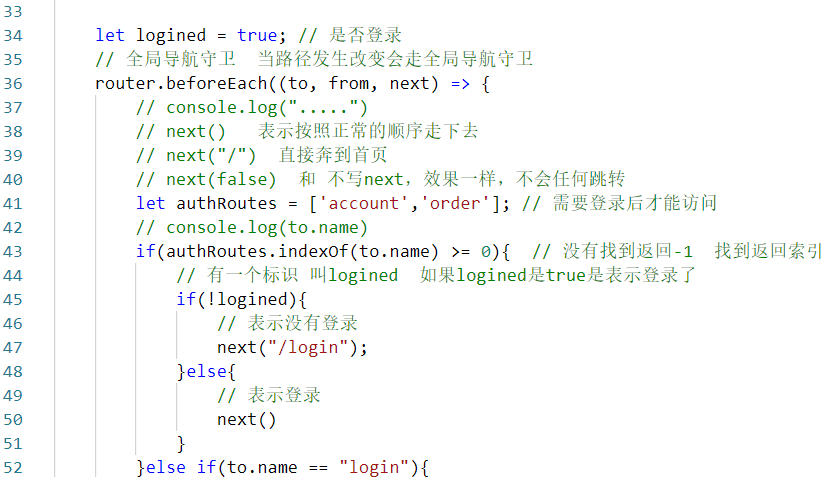
\* next()：按照正常的流程来走。

\* next("/")：之前的路由被断掉了，重新走到/中去。

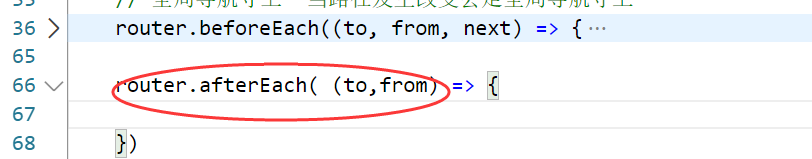
\* next(false)或者是没有调用：不会导向任何路由。

* 2. `afterEach(to,from)`：路由完成后的回调。

代码如下：



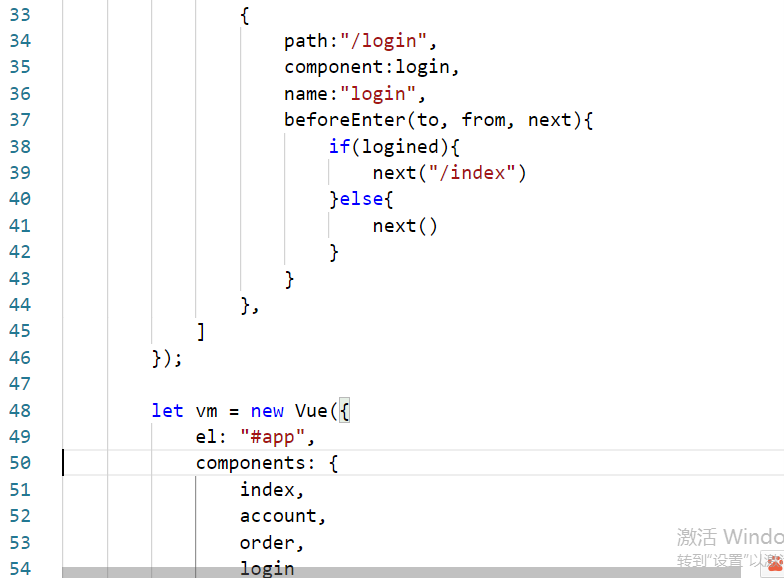
对于afterEach，自己打印出来，研究一下：



## 导航守卫之路由导航守卫

在定义路由的时候，可以传递一个`beforeEnter(to,from,next)`参数来实现。里面的参数跟之前是一样的。

代码如下：

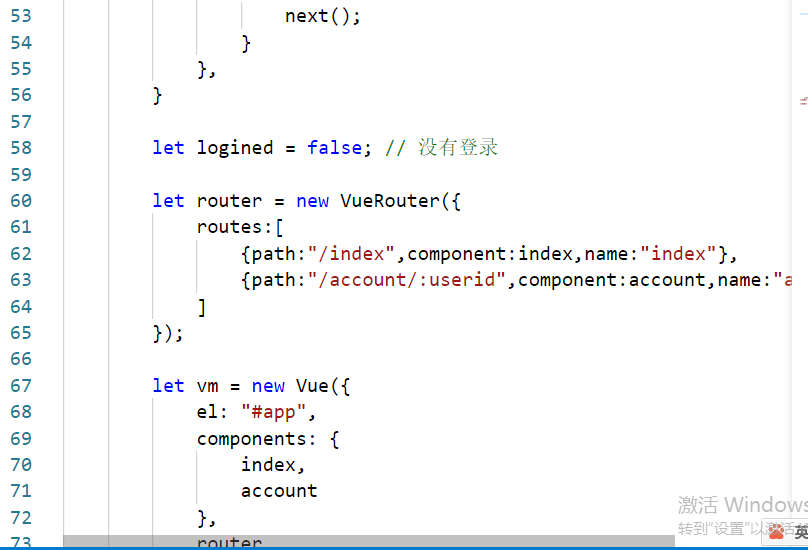


## 导航守卫之组件导航守卫

**组件导航守卫：**

* beforeRouteEnter(to,from,next)：当前页面被进入之前调用。
* beforeRouteUpdate(fo,from,next)：当前页面被复用了，参数改变了，会调用这个函数。
* beforeRouteLeave(fo,from,next)：当前页面即将离开了，会调用这个。

代码如下：

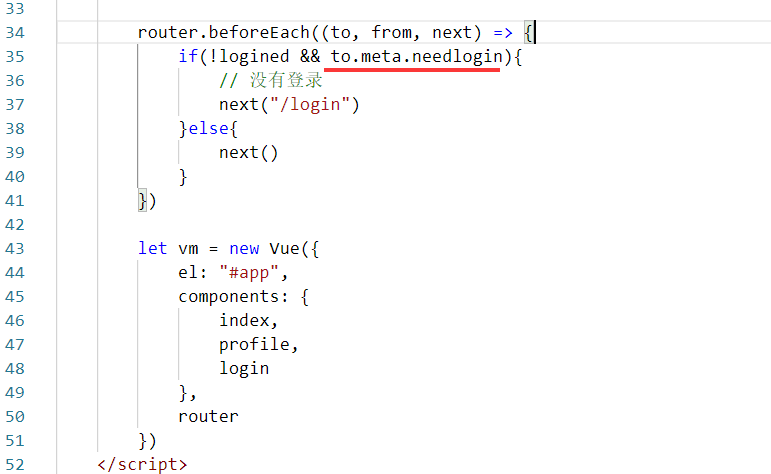
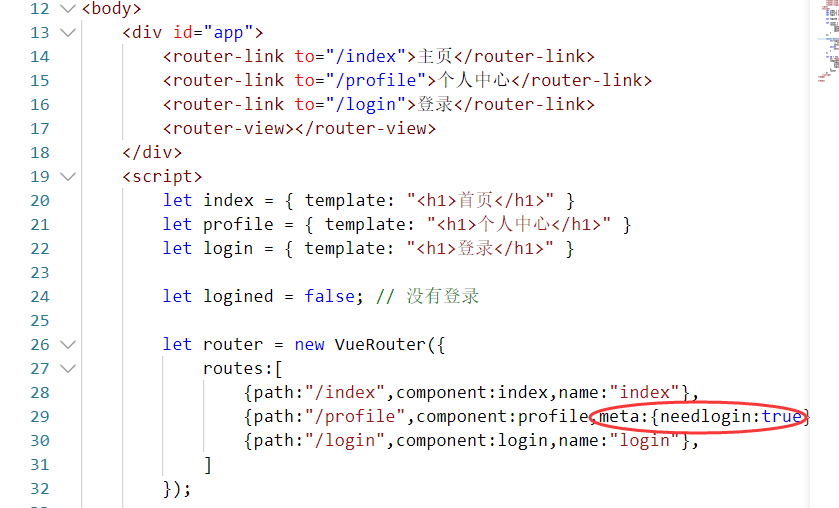


**导航守卫执行的流程：**

* 1. 导航被触发。
* 2. 在失活的组件里调用离开守卫。
* 3. 调用全局的 beforeEach 守卫。
* 4. 在重用的组件里调用 beforeRouteUpdate 守卫 (2.2+)。
* 5. 在路由配置里调用 beforeEnter。
* 6. 解析异步路由组件。
* 7. 在被激活的组件里调用 beforeRouteEnter。
* 8. 调用全局的 beforeResolve 守卫 (2.5+)。
* 9. 导航被确认。
* 10. 调用全局的 afterEach 钩子。
* 11. 触发 DOM 更新。
* 12. 用创建好的实例调用 beforeRouteEnter 守卫中传给 next 的回调函数。

## 路由的元信息

直接上代码如下：



# Axios

## 什么是axios？

**中文网：**<http://www.axios-js.com/>

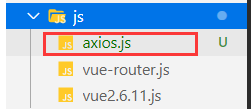
**npm:** <https://www.npmjs.com/package/axios>

**源码：**https://unpkg.com/axios/dist/axios.min.js

基于**Promise**的HTTP请求客户端。用于向服务器发送请求的一个工具。

promise 是用于处理异步请求。

和vue-router一样，先把axios下载下来，如下：



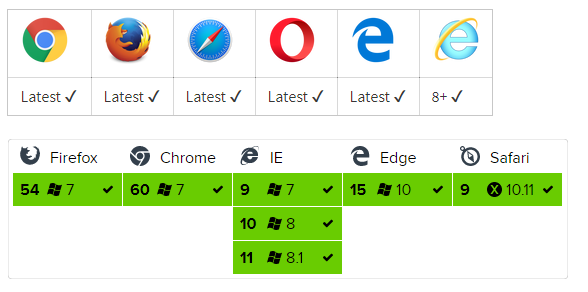
## 功能特性

在浏览器中发送 **XMLHttpRequests**（在后面学习ajax时会去专门讲到） 请求

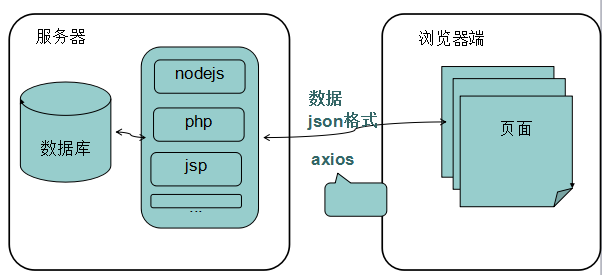
转换请求和响应数据

自动转换 **JSON** 数据

它在浏览器端运行。



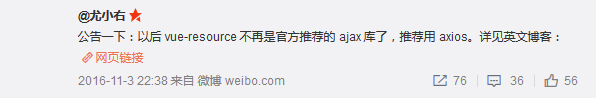
## 作用



我们从前端向服务器请求数据：

1. 查询一个商品的详情信息。
2. 查询本地的天气预报。
3. 查询一个用户是否被占用。
4. ........

我们可以通过很多的工具向服务器端发请求：

1. jquery.ajax
2. ~~vue-resource~~
3. **Axios:它是一个完全独立的js库，与其它库没有任何的关系。**
4. ....

## Json

JavaScript Object Notation是一种轻量级的数据交换格式。



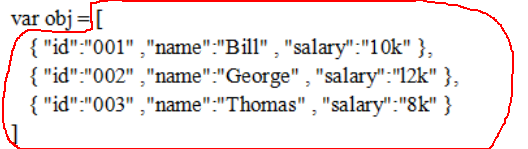
---- 用javascript中的对象的格式来表示数据。

### 特点

* JSON 指的是 JavaScript 对象表示法（JavaScript Object Notation）
* JSON 是轻量级的**文本数据交换格式**
* JSON 独立于语言文本数据格式。**与具体的程序设计语言没有关系**，各种语言中都支持json格式的数据。各类**程序设计语言**都支持json。
* JSON 具有自我描述性，更易理解
* JSON 具有层级结构（值中存在值）

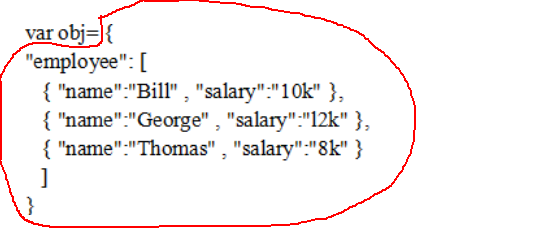
### 两种格式：数组，对象

#### 示例：数组表示的 json



红色部分才表示是符合json格式要求的数据。

#### 示例：对象表示的 json



红色部分才表示是符合json格式要求的数据。

### 语法规则

* 数据在**键/值对**中
* 数据由逗号分隔
* 花括号保存对象
* 方括号保存数组
* **键名必须使用””包含（不能省略，也不能写成’’)**
* 值可以是：
  + 数字（整数或浮点数）
  + 字符串（在双引号中）
  + 逻辑值（true 或 false）
  + 数组（在方括号中）
  + 对象（在花括号中）
  + Null
  + **不能是函数**

**与js中的对象相比，只有两点不同：**

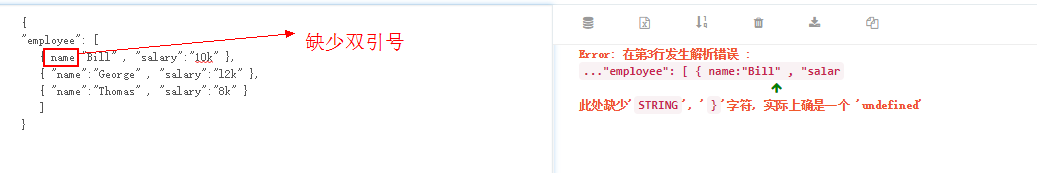
1. 在书写上，属性名必须一定要用””包含。
2. 属性的值，不能是函数。因为json是用来表示数据的，而函数是动态的，它必须要调用才能起效果。

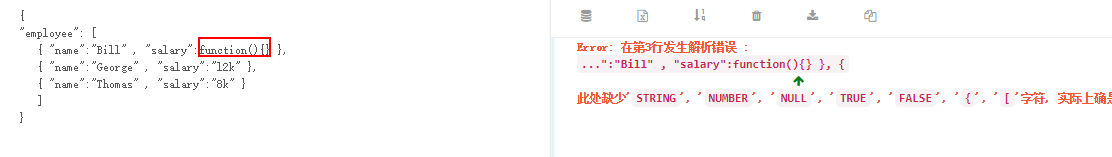
### 在线验证

检查当前的字符串（文本）是不是json的格式的。

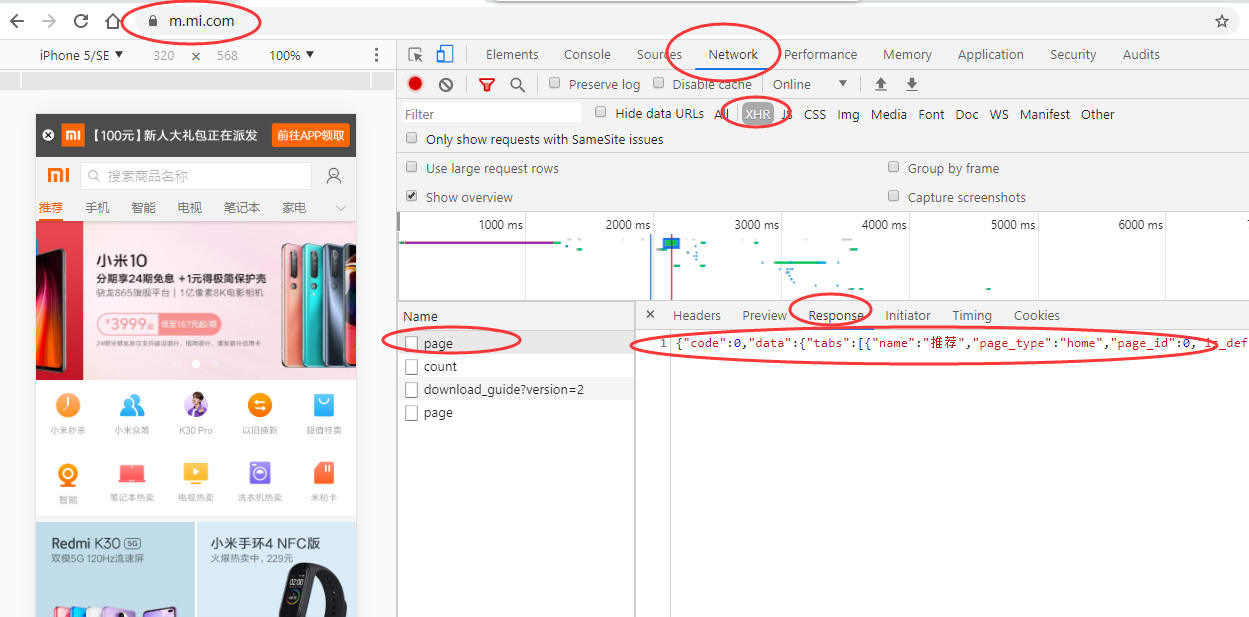
<http://www.json.cn/>





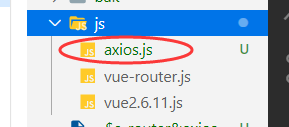


### 例子



## 如何使用axios

### 下载，引入js文件。



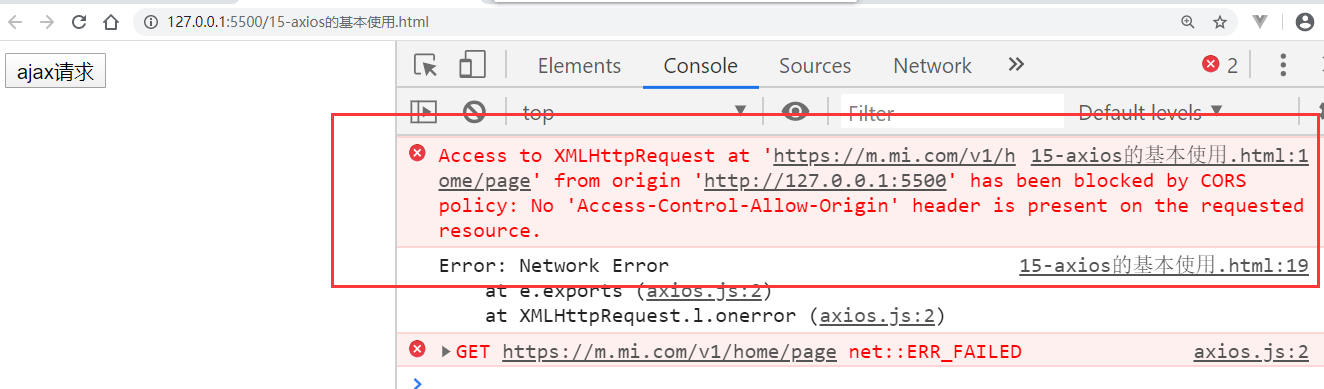


* get是指请求数据，它还有另一种方式post。这两个方式见于form中的methods属性。
* url，是服务器的接口地址。一般是服务器公开的接口。

https://m.mi.com/v1/home/page

* then()如果请求数据成功，则进入then，其中有一个回调函数，它的参数value中就保留了axios成功请求之后的所有的数据信息。
* catch（）如果请求失败，则会执行它的回调函数。类似于try..catch

运行代码，点击按钮后，发现报错：



关键字：**Access-contro-allow-Origin**

你现在在**跨域** ，这是不允许的！

### 跨域

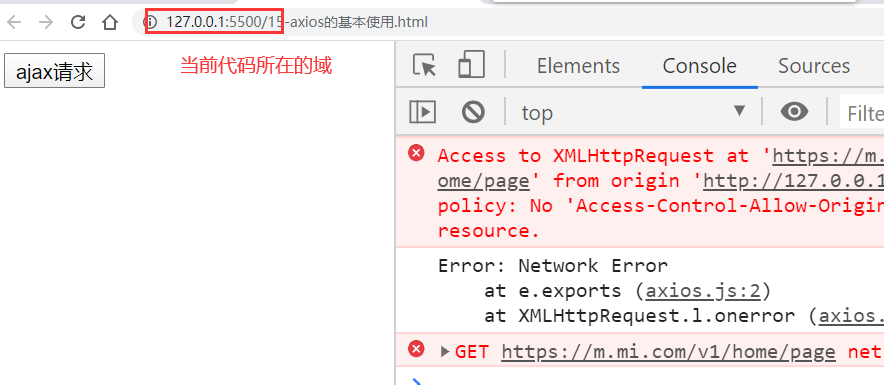
跨：超过了，出界了

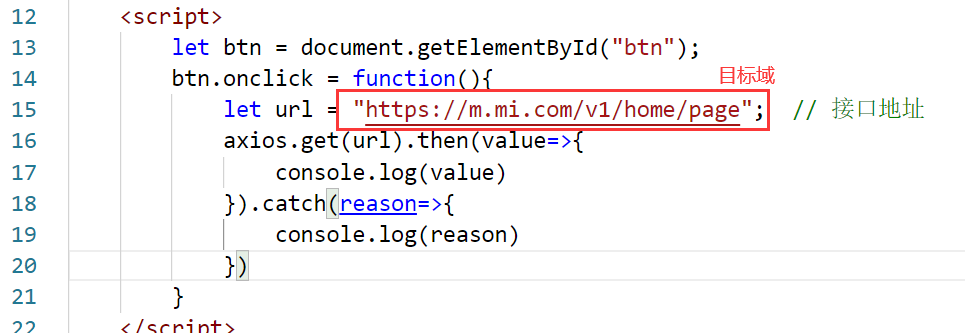
域：区域，域名。

理解上面的问题：

现在有两个域：

* 你请求的那个服务器的地址。
* 你当前所在网页的地址。





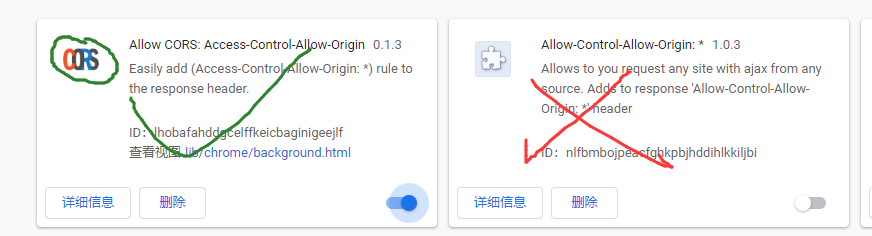
两个地址如上，很明显，它们不是同一个区域的地址，这就是跨域。在浏览器端有一个同源策略：浏览器不允许你通过**js代码**去访问另一个域中资源。

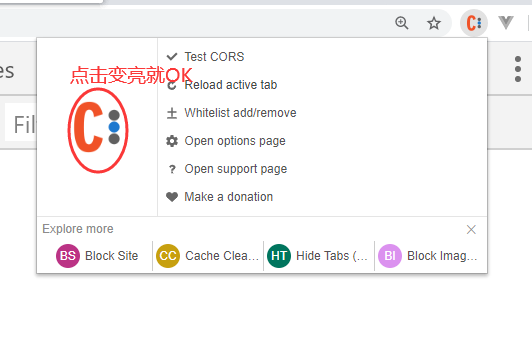
### 如何解决跨域的问题

**常见的方案：**jsonp cors 代理 ngnix

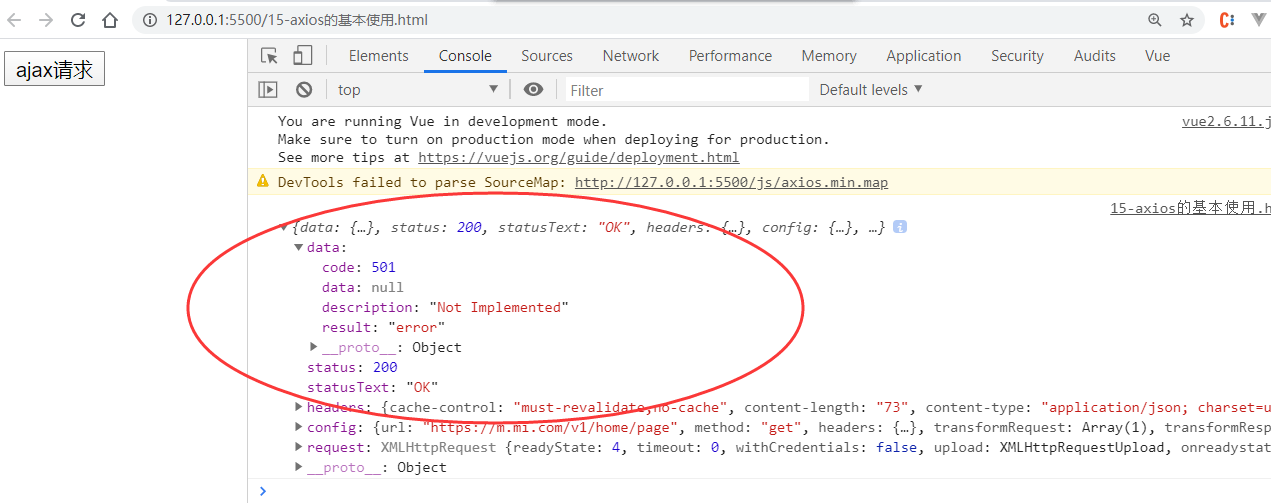
一个最笨的，最简单的方法：在浏览器安装一个插件：

chrome浏览器中的cors插件。

1. 墙
2. 下载安装：  
   
3. 启用



**这个插件有时候不太好使。调用小米app接口，效果如下：**

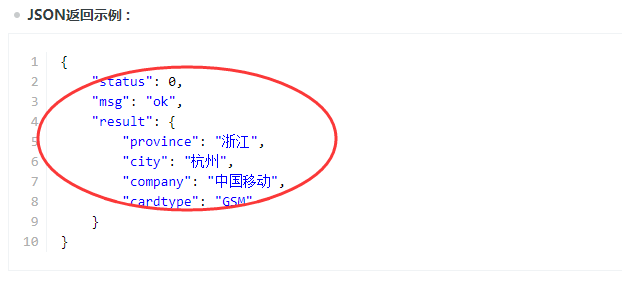


使用网上的api接口数据：

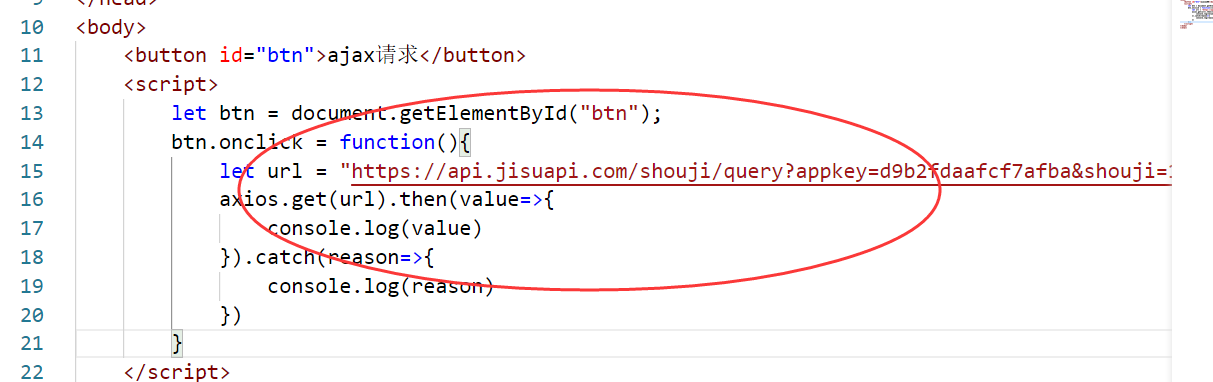


使用方式：

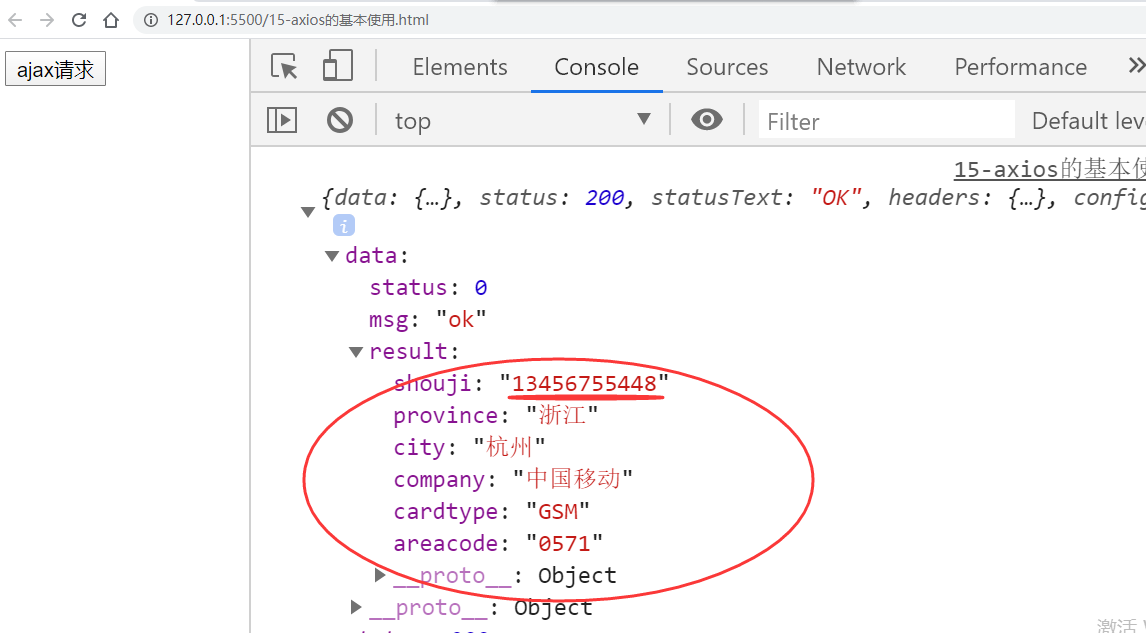




玩一下：



效果如下：



示例：通过公开的api查询手机号信息

[https://tcc.taobao.com/cc/json/mobile\_tel\_segment.htm?tel=](https://tcc.taobao.com/cc/json/mobile_tel_segment.htm?tel=13617685969)**[13617685969](https://tcc.taobao.com/cc/json/mobile_tel_segment.htm?tel=13617685969)**

示例：通过卖座网的公开接口，去访问电影信息

http://m.maizuo.com/v4/api/film/now-playing?\_\_t=1489757848979&page=1&count=5

# Vue-cli

# vue中的UI组件库