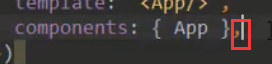
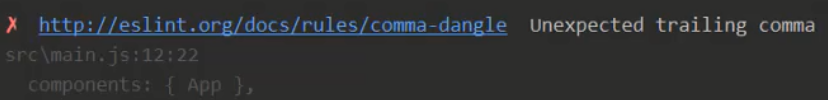
# 脚手架3 VueCLi3

## 安装CLi错误和eslint规范

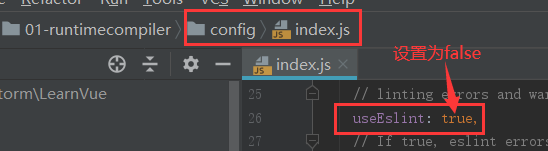
* 安装CLi错误，安装命令npm install @vue/cli -g 。解决方法：
  + 先卸载node，再继续安装CLi
  + 以**管理员身份**打开命令行终端CMD，然后输入命令 npm clean cache -force来清空文件夹，再继续安装CLi。或者手动清空或者删除npm-cache文件夹，路径是C:\Users\Administrator\AppData\Roaming\npm-cache，然后再安装CLi



* Eslint规范
* 在代码最后多了个逗号会报错，但其实按照之前的写法这样写是不会报错的，所以使用Eslint规范有时候也不好。

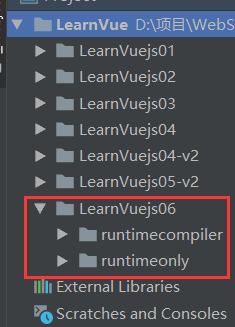
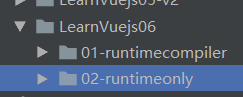


* 如果选择了Eslint规范，但是后面又想关掉Eslint规范，可以这样做，在config文件夹下的index.js文件中，把useEslint设置为false。关闭index.js，在命令行再重新编译npm run dev就不会报错了。

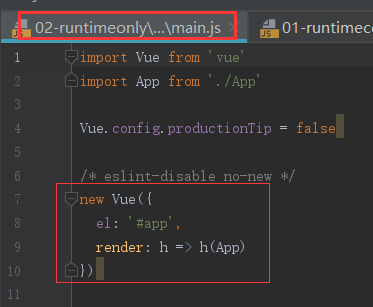


## runtime-compiler和runtime-only的区别

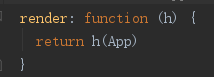
* 新建文件夹LearnVuejs06，并在LearnVuejs06路径的命令行创建2个脚手架项目runtimecompiler（选runtime+compiler）和runtimeonly（选runtime+only），再重命名为

 重命名后

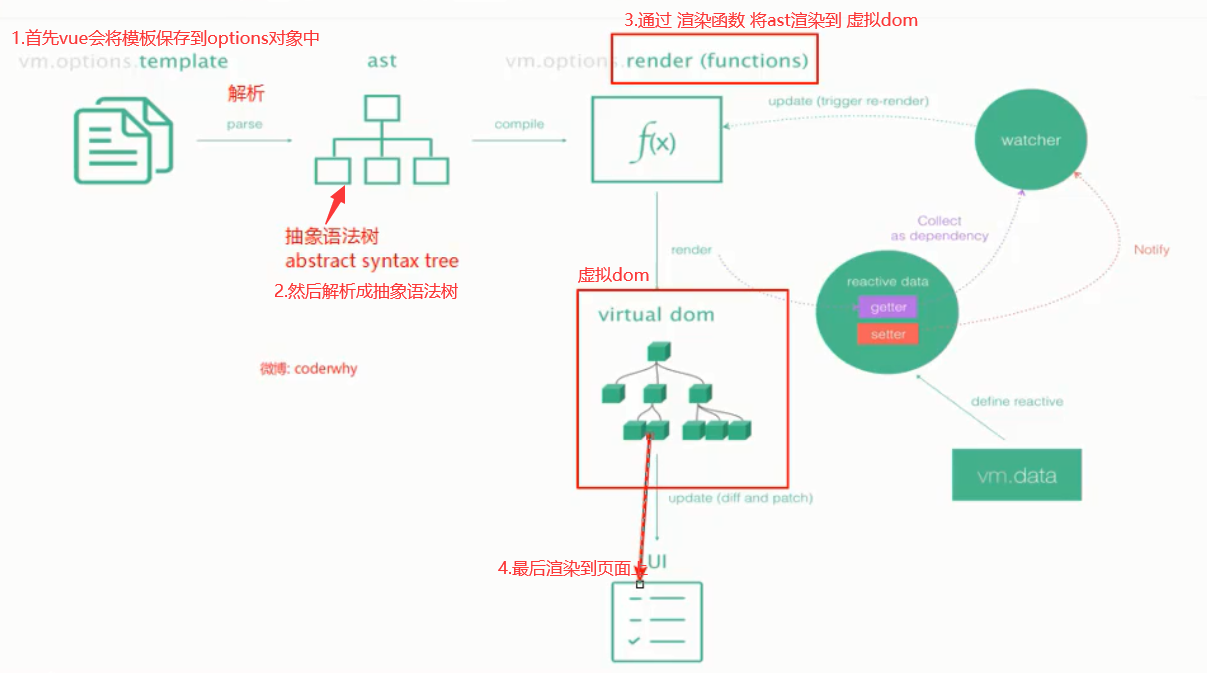
* runtime-compiler和runtime-only的区别，是main.js文件不一样

1. runtime-compiler的main.js中，使用vue时，先注册组件App，再使用模板App
2. runtime-only的main.js中，使用的是箭头函数（相当于是于下面的这个方法），render（渲染）本身就是函数，



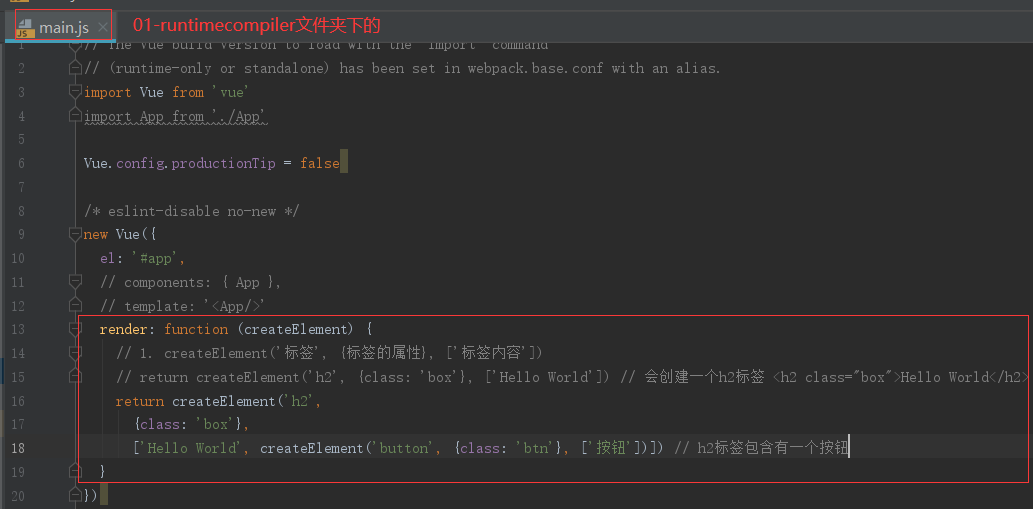
1. runtime-compiler中的程序运行过程： 

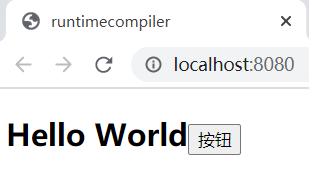


runtime-only的程序运行过程：

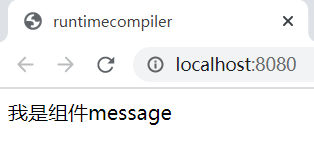
* 所以runtime-only的 ①性能更高、②代码量更少，最后打包出来的文件比runtime-compiler小6KB，开发中用的多的其实是runtime-only
* runtime-compiler的main.js中，其实也可以使用render函数

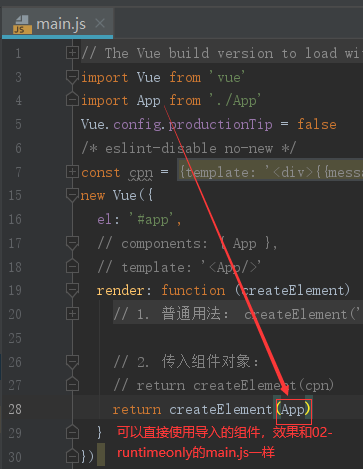
1.普通用法： createElement('标签', {标签的属性}, ['标签内容'])



运行结果

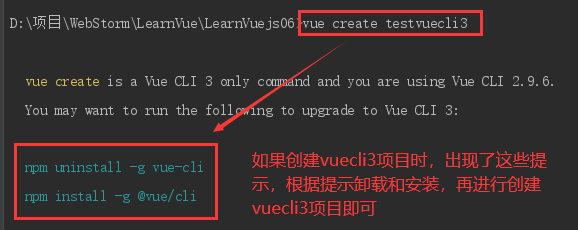
2. 传入组件对象：

运行结果

## VueCLi3创建项目和目录结构

1. 创建VueCLi3项目，命令vue create testvuecli3

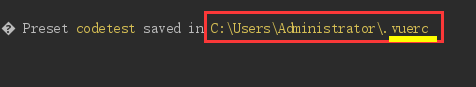


* 创建之后会有一些选择：

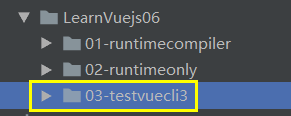
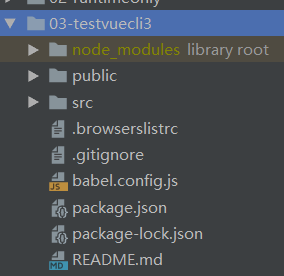
Manually select features手动（Manually）选择特性；

只需要这个，其他的就按上下键和空格来不选中；

选中单独（delicate）放在一个配置文件中；

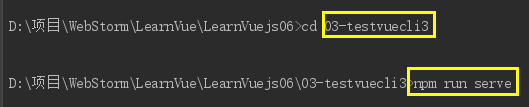
如果想删了codetest可以从这个文件夹删除。

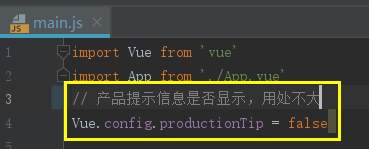
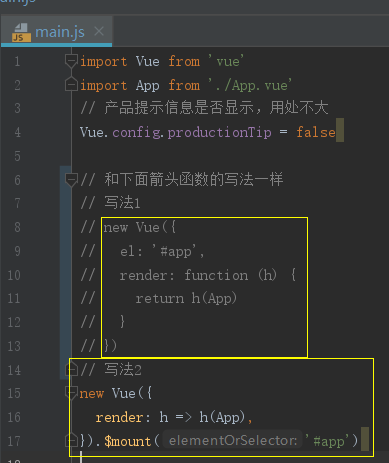
* 创建好VueCLi3项目后重新命名

目录结构比VueCLi2简洁

VCS：version control system（版本控制系统，如git/svn），创建好VueCLi3项目，文件夹下会多出一个.git文件夹，我们就可以使用git命令进行操作 创建的VueCLi3项目。

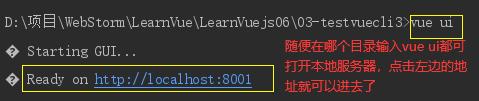
* 通过命令npm run serve运行项目，也可以根据提示信息来运行项目。

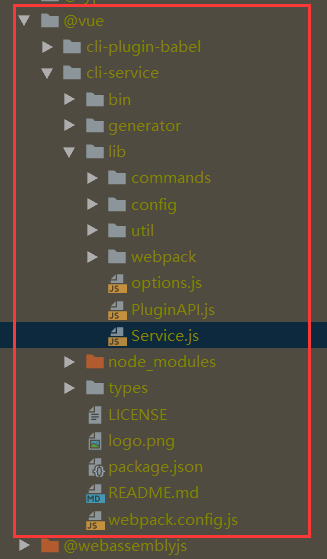
* 
* 

## VueCLi3配置文件的查看和修改

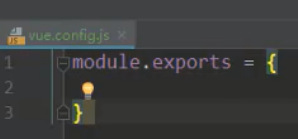
1. 方法一：启动本地服务器vue ui

 本地服务器界面如下

1. 方法二 ：隐藏起来的配置文件放到node\_modules文件夹下的@vue文件夹下的cli\_service文件夹和lib文件夹里了。

  所有的配置文件都隐藏到这了。

1. 方法三：在根目录03-testvuecli3目录下创建一个vue.config.js配置文件（文件名不能乱取，必须是这个），把需要配置的东西写到里面，然后module.exports进行导出，与node\_modules文件夹下的配置文件进行合并。

，并使用git命令添加和提交文件

【git add 文件名】

【git commit -m ‘修改配置文件或其他提示信息’】

## 箭头函数的使用和this指向

1. 箭头函数的定义方式: const 函数名 = (参数列表) => {}，如：

const sum = (num1, num2) => {

return num1 + num2

}

1. 只有一个参数时，括号可以省略不写。如：

const power = num => {

return num \* num

}

1. 若箭头函数 代码块中只有一行代码时,可以写成以下格式

const mul = (num1, num2) => num1 \* num2

const mul = (num1, num2) => {

return num1 \* num2

}

const demo = () => {

console.log('Hello Demo');

}

console.log(mul(20, 30));

const demo = () => console.log('Hello Demo');

console.log(demo());

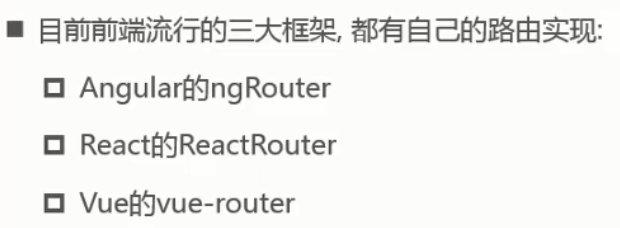
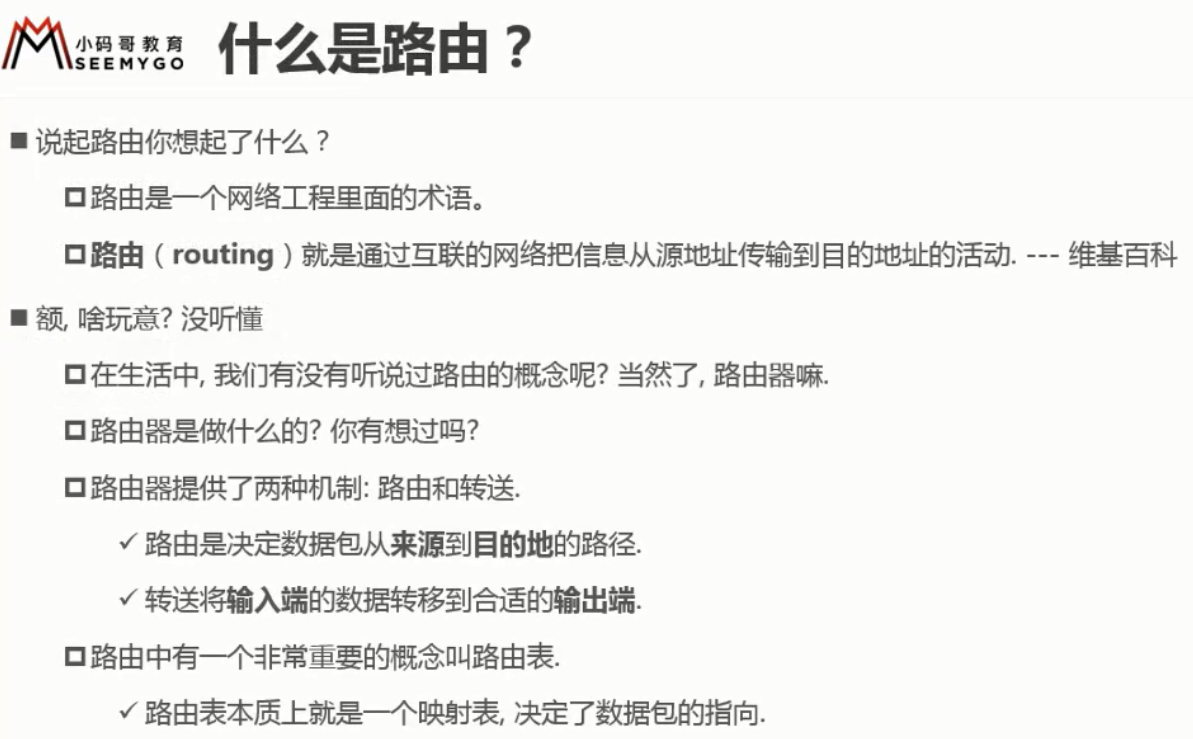
1. 什么时候使用箭头函数：当需要把 一个函数 当做 另一个函数的参数 时使用。
2. 箭头函数中的this是如何查找的?

答案: 向外层作用域中, 一层层查找this, 直到有this的定义.

即查找最近的作用域，如果自己没有，就往上一级找this，直到找到this，这时箭头函数就指向这个this。

## 什么是路由和其中映射关系

### 6.1 认识路由vue-router

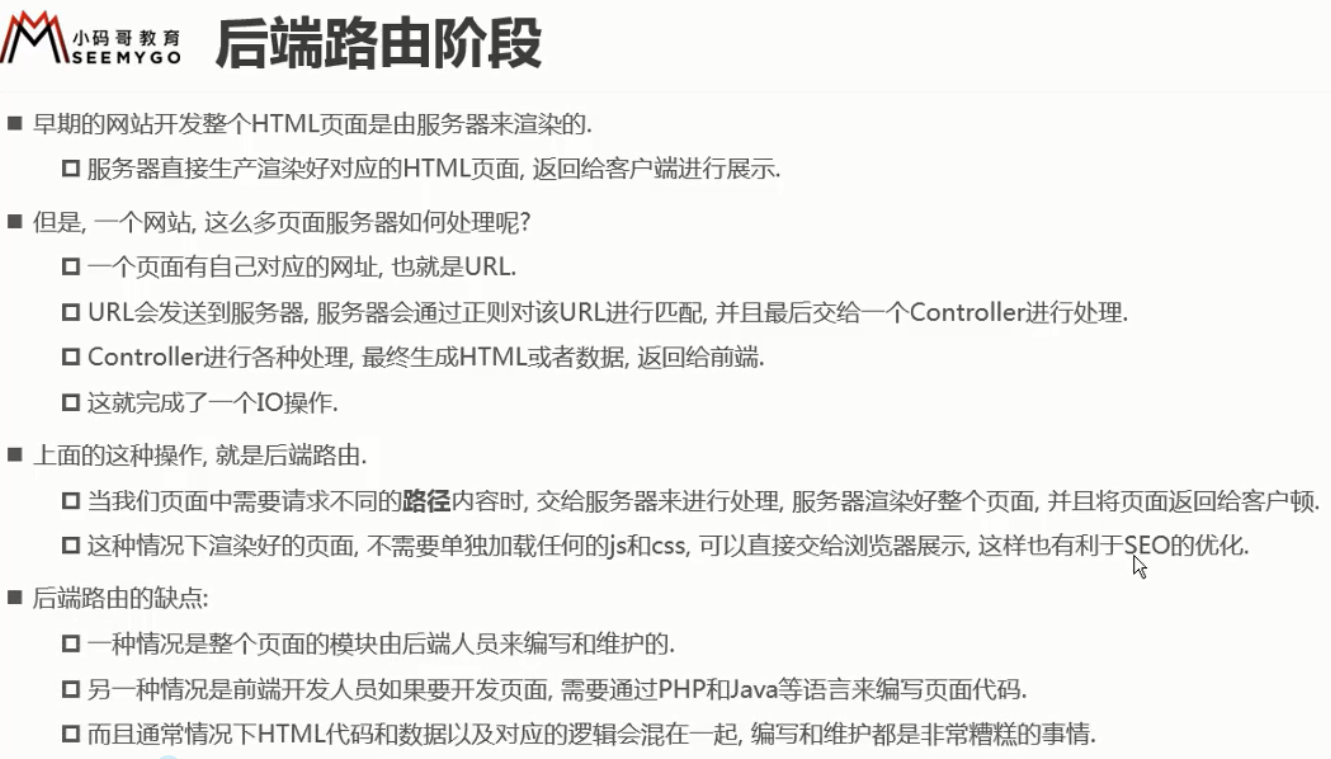
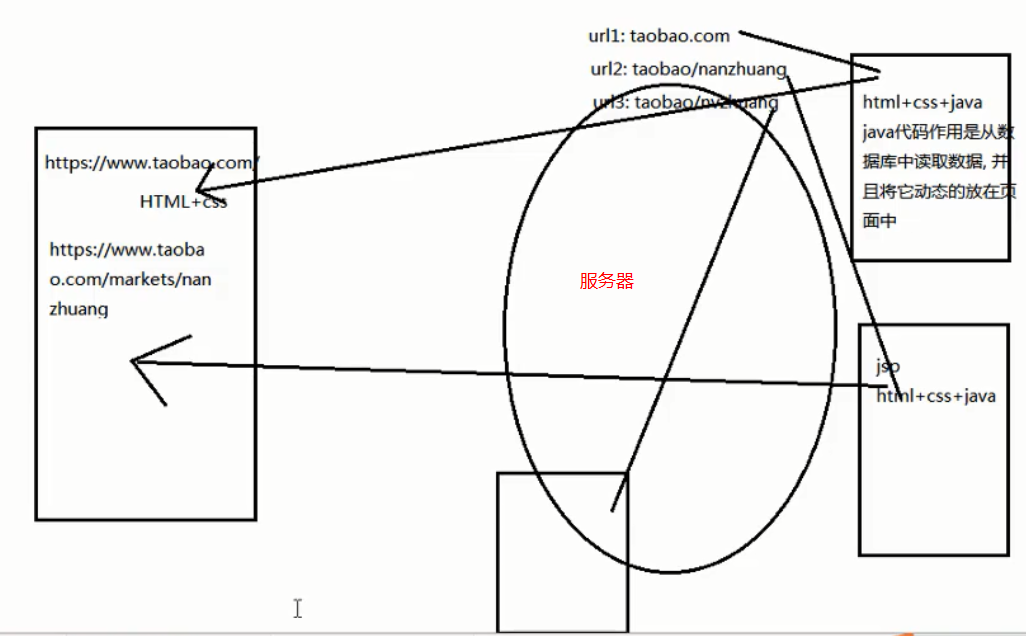


1. 先创建一个CLi2的项目learnvuerouter

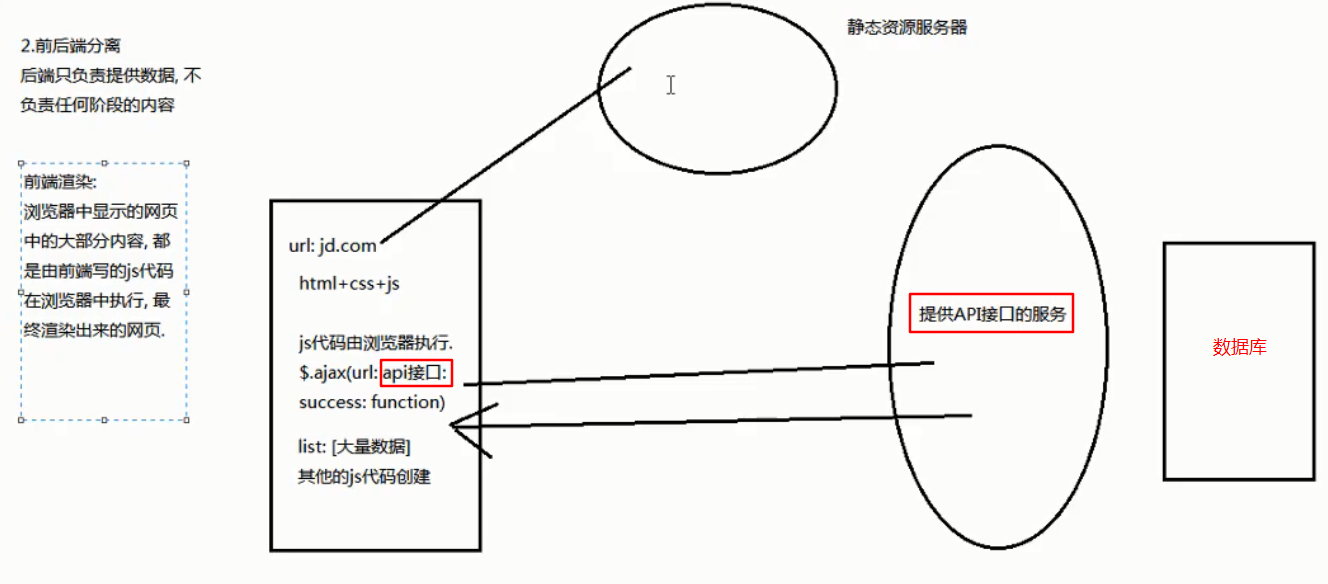


## 前端渲染、后端渲染和前端路由

* 后端渲染：以.jsp文件 .php 进行渲染，网页每传一个URL到服务器，将url由一个对应的.jsp文件来处理，转成html+css+java，在后端就已经有页面了，再将结果返回到页面中；
* 后端路由：后端处理URL和页面之间的映射关系。



* 前端渲染：浏览器中显示的网页中大部分内容，都是由前端写的js代码在浏览器中执行，最终渲染出来的网页。

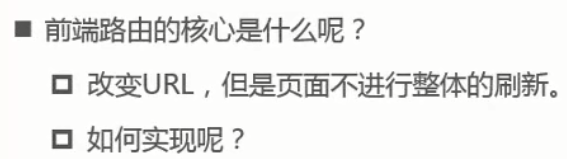
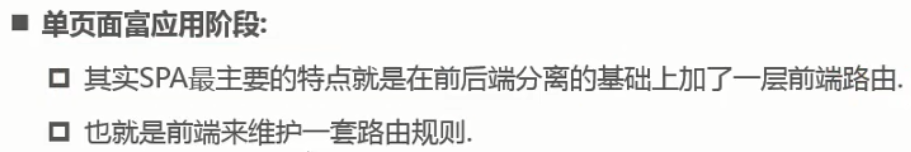
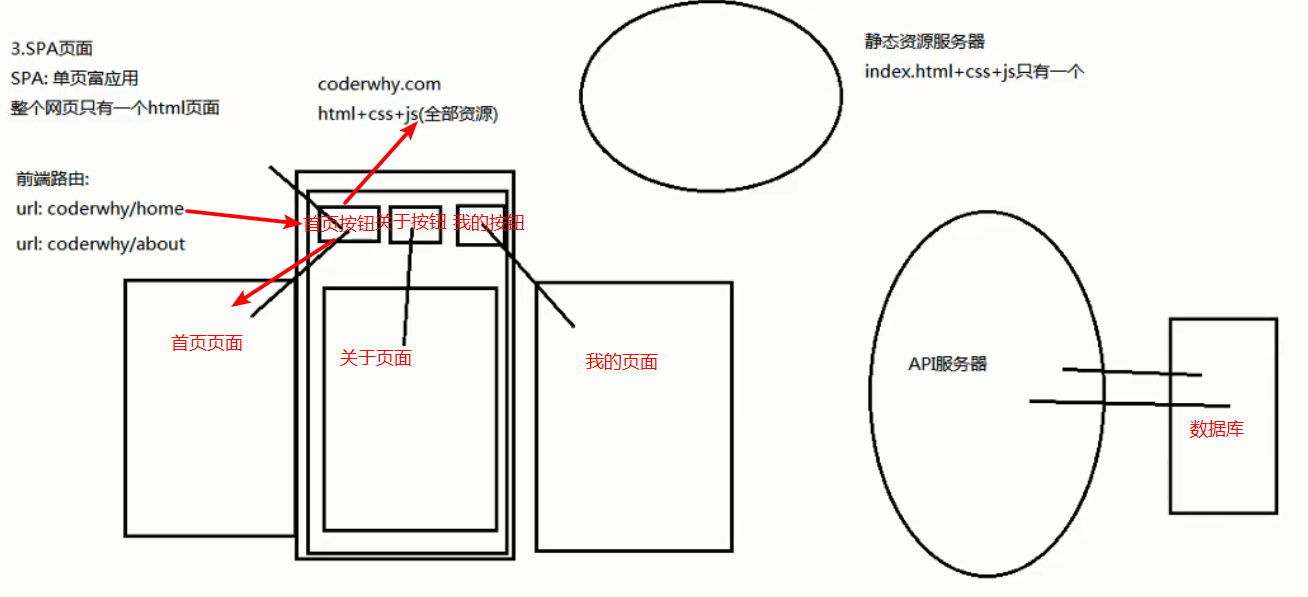


* 前后端分离：后端只负责提供数据，不负责任何阶段的内容。



* SPA页面（simple page web application 单页富应用）：整个网站只由一个html页面。

每个请求的url有一个对应页面，由前端路由来进行管理。

改变URL但是页面不刷新，可以通过url的hash和HTML5的history

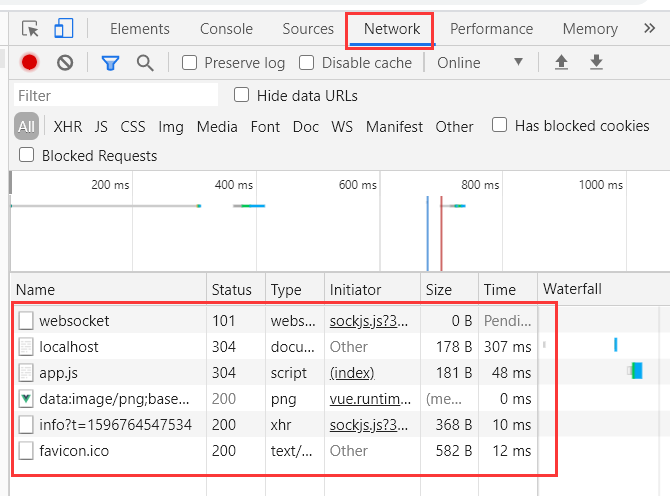
## url的hash和HTML5的history



1. 运行项目，打开页面的网址

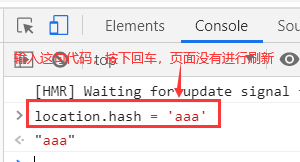
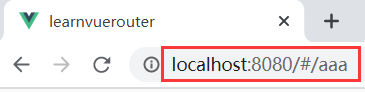


1. 刷新页面，可以看到页面请求了以下的资源

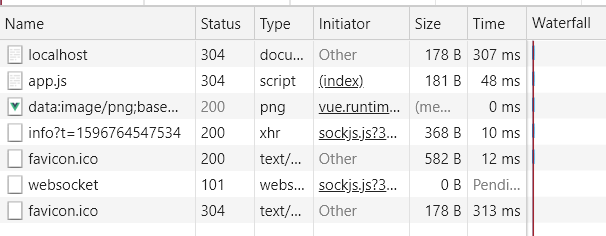


1. 改变URL，但是不让页面进行刷新（很多情况下，改变URL，页面是会自动刷新的）

方法一：URL的hash

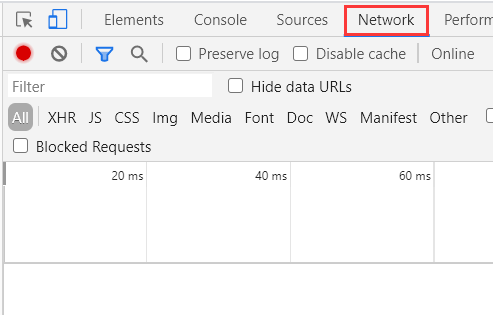
*  

可以看到页面的URL改变了，但是页面没有进行刷新，资源也没有重新请求

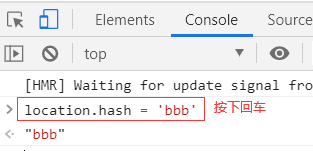
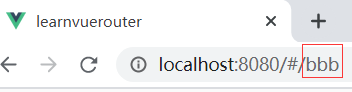


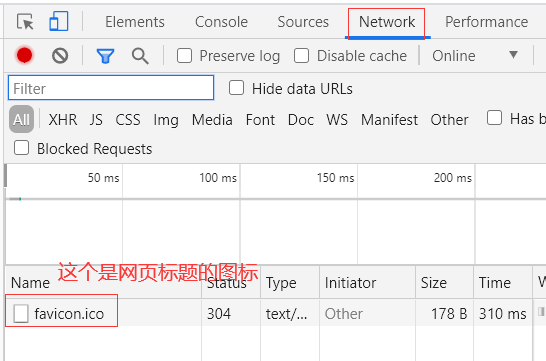
* 清空资源请求



清空后👉

通过hash修改URL的值

 页面的URL变为

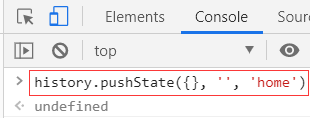
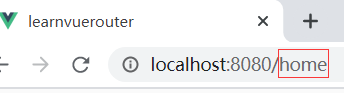
页面的资源请求清空👉

方法二：HTML5的history

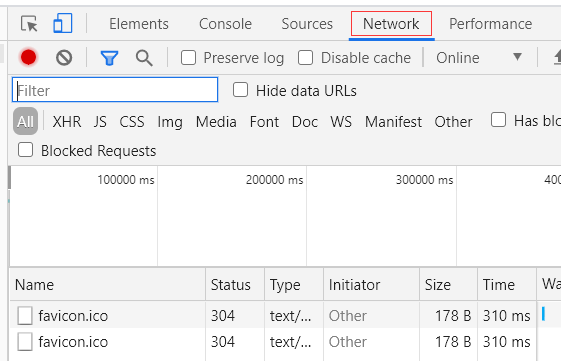
 

通过history改变URL

pushState类似于栈结构，先进后出，通过pushState修改URL，页面总是显示后面更改的URL地址，即当前pushState修改URL是home，页面URL就显示home。可以通过history.back()命令回到上一步。pushState相当于入栈，back()相当于出栈。

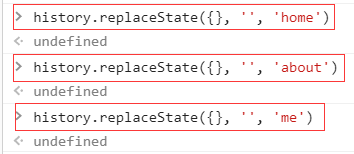
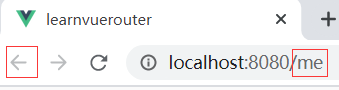
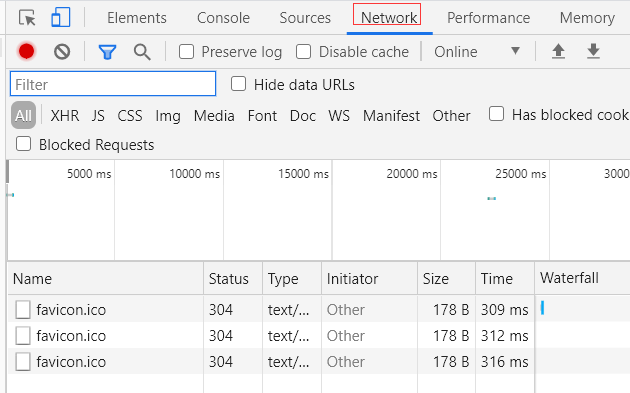
URL被修改了👉 

但是页面没有刷新，资源也没有重新请求



方法三

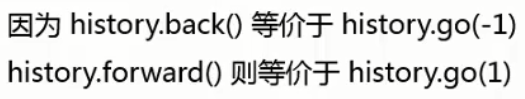
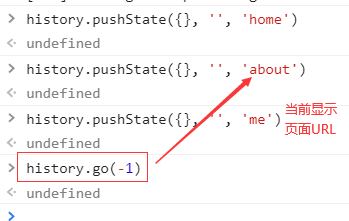


直接替换URL，不能回退到上个页面，也没请求资源 

方法四：

history.go(N)里面是负值则回退到第N页面等同于history.back()。history.go(N)里面是正值则往前N个页面等同于history.forward()

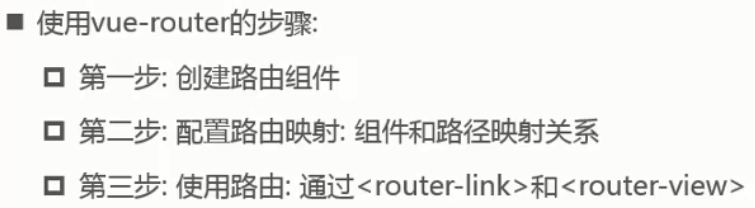


## vue-router安装和配置方式



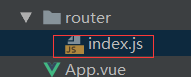
## 路由映射配置和呈现出来

### 10.1 vue-router基本使用



创建项目时已经选择安装了路由，就不用再安装

1. 删除router文件夹，再新建router文件夹和index.js来配置路由相关信息。

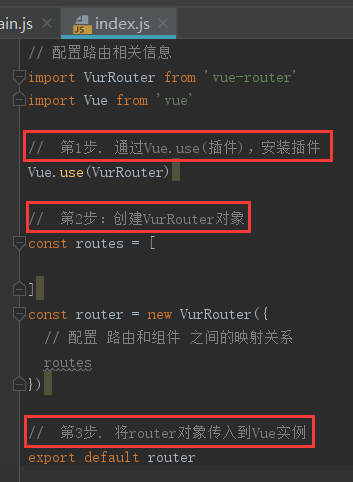
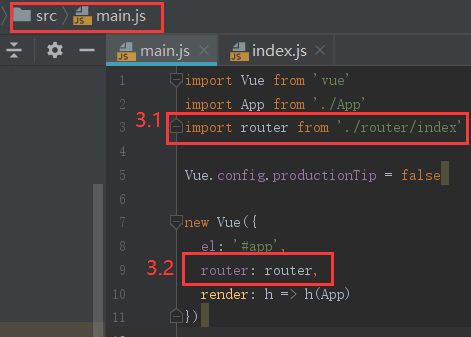
 

### 10.2 搭建路由框架

1. 导入路由

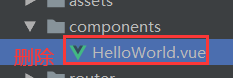


1. 使用路由，3个步骤：

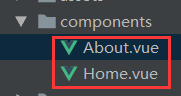
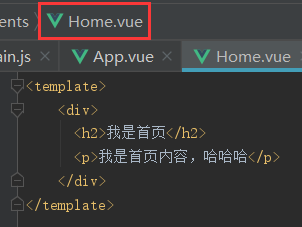
 挂载路由👉 

### 10.3 配置路由映射关系和呈现出来

配置 路由和组件之间的映射关系，一个URL对应一个组件，这就是一个映射关系。一个映射关系就是一个对象。

删除app.vue中引用helloworld.vue的部分后：

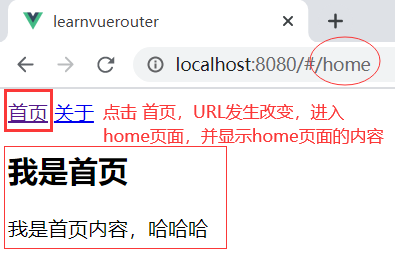
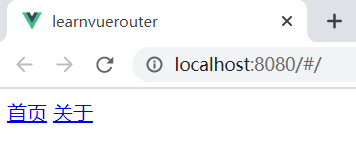
1. 创建组件，并添加一些网页元素

1. 配置路由映射关系

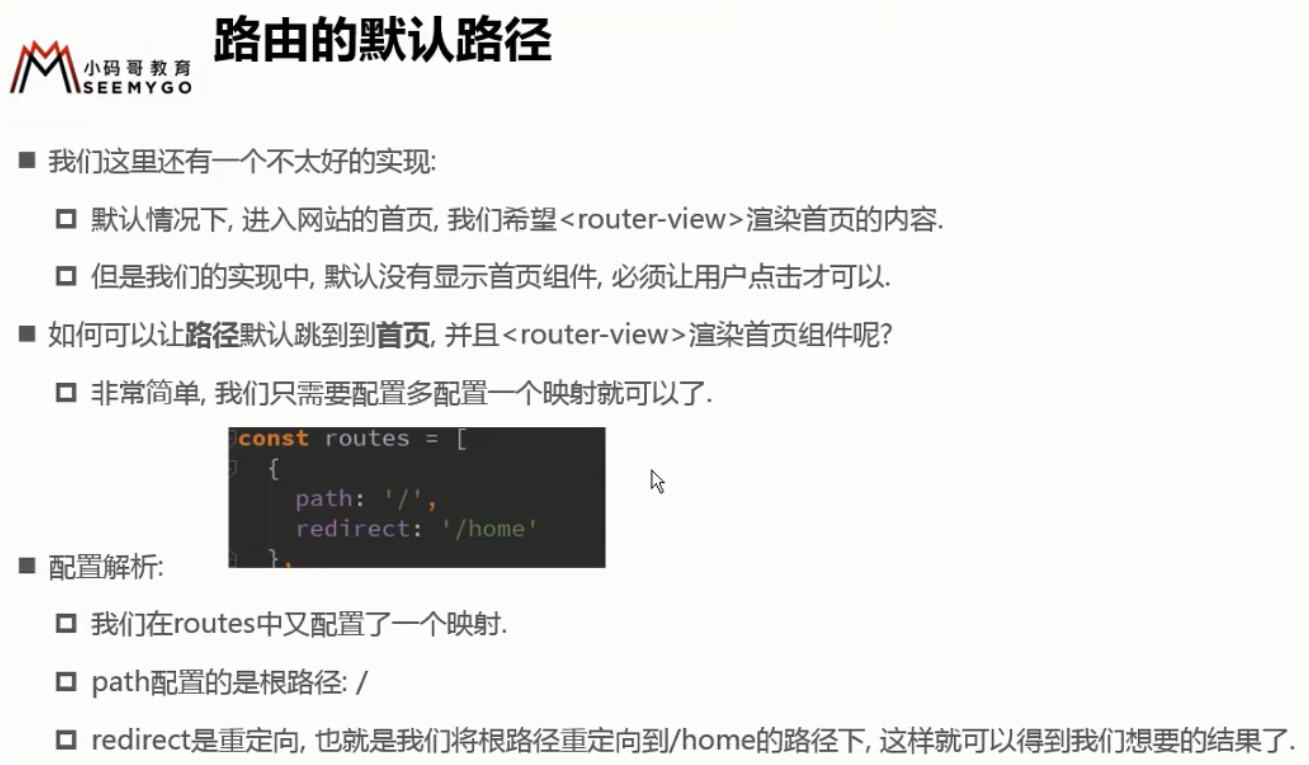


1. 使用组件，router-link用于显示标签，router-view用于显示组件内容渲染的位置

保存并打开网址

## 路由的默认值和修改为history

### 11.1路由的默认路径



正确的页面，应该是已进入页面就显示首页的内容，而不是点击了首页标签才再显示内容



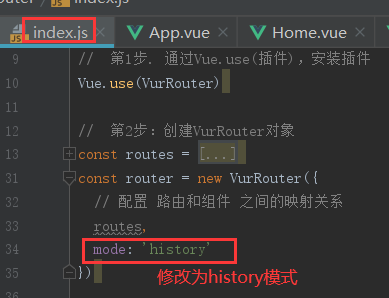
### 11.2 将URL路径改为history模式



hash的URL中带有#，改成history让路径中没有#

hash模式的URL

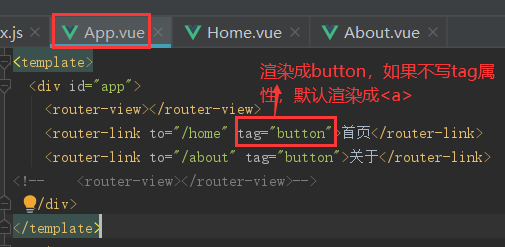
history模式的URL



## router- link的其他属性补充



1. tag属性

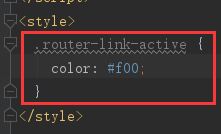


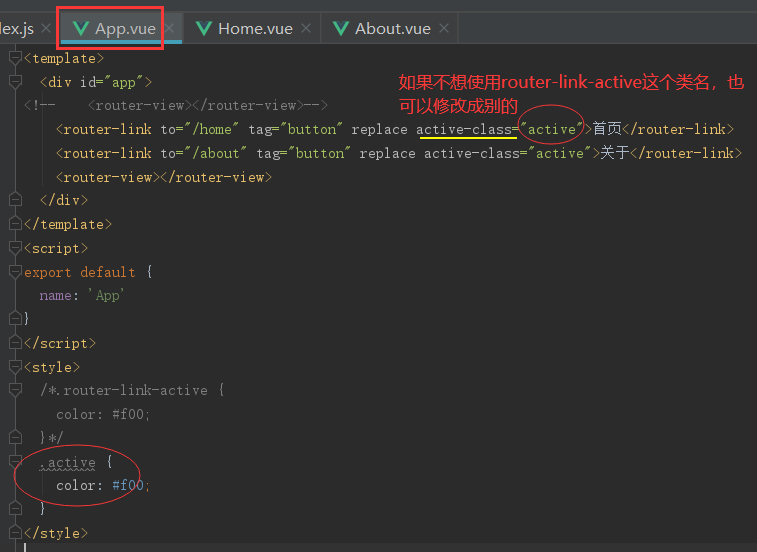
1. replace属性



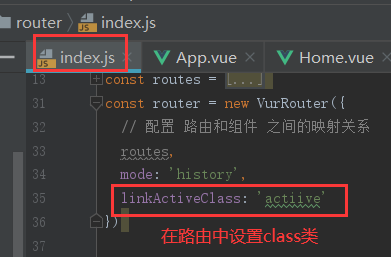
1. active-class

点击哪个按钮时，哪个按钮就添加一个类，不被点击的按钮不添加该类

不写active-class属性，则默认类名是router-link-active，并可修改这个样式。



但是如果多个标签都需要修改类名，可以通过在路由里面修改类名，就不需要给每个标签都要多添加一个class属性了，如果要修改样式，就改active样式即可。



## 通过代码跳转路由使用$router

1. 通过this.$router.push(‘路由URL’)，可以前进和后退页面
2. 通过this.$router.replace(‘路由URL’) ，不可以前进和后退页面

