# 一、跨域问题

## 1. 什么是跨域

这是浏览器对js的一种安全限制，也就是说浏览器的页面内去访问其他服务器上的资源的时候，就会出现跨域。

同源策略明确了什么情况是属于跨域。所谓的同源策略，指的是协议、域名、端口完全相同，才是安全的。

## 2.跨域的解决方案

md文档中提供了不同环境下的解决方案，参考即可。

# 二、路由（组件之间的跳转）

路由器的功能：在数据通信时 帮你选择通信的路线

在vue中的路由，能够帮助我们在一个vue组件中实现其他组件的相互切换。

也就是说，可以通过路由模块，再创建路由表，将制定的组件显示在路由视图中。 如何实现路由跳转？

## 1. 安装路由模块

npm install vue-router -s

## 2.设计路由界面

## 3.创建路由表

|  |
| --- |
| **import** Home **from './views/Home' import** Products **from './views/Products'  export const** routes = [  {  **path**:**'/Home'**,  **component**:Home  },  {  **path**:**'/Products'**,  **component**:Products  } ] |

## 4.在main.js中使用路由模块以及注册路由表

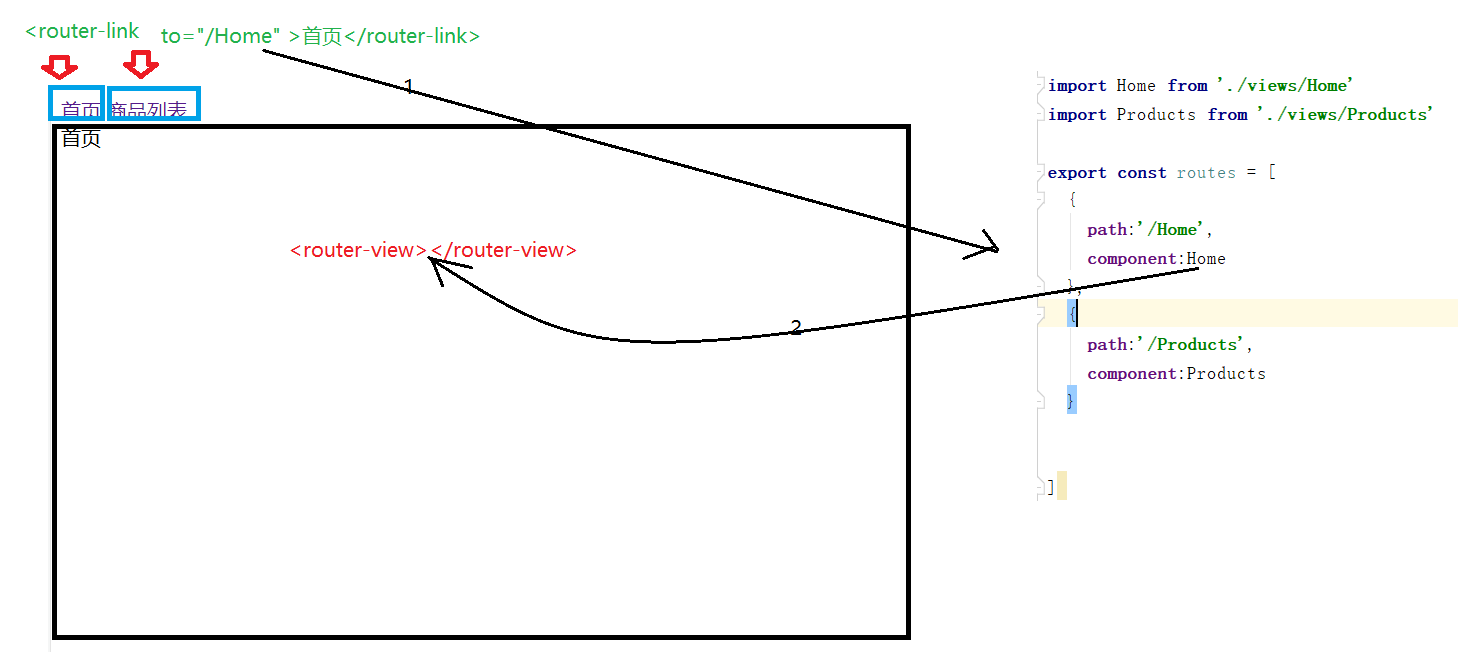
|  |
| --- |
| **import** Vue **from 'vue' import** App **from './App.vue' import** VueRouter **from 'vue-router'** *//1.引入路由模块* **import** {routes} **from './routes'** *//2.引入静态路由表* Vue.use(VueRouter)*//3.使用路由模块  //4.创建一个VueRouter模块的实例* **const** router = **new** VueRouter({  **routes**:routes });  **new** Vue({  **el**: **'#app'**,  router,*//5.把router实例放入到vue实例中* render: h => h(App) }) |

## 5.创建路由链接和路由视图

在App.vue里

|  |
| --- |
| <**template**>  <**div id="app"**>  <**div**>  <**span**>  <**router-link to="/Home"**>首页</**router-link**>  </**span**>  <**span**>  <**router-link to="/Products"**>商品列表</**router-link**>  </**span**>  </**div**>  <**router-view**></**router-view**>  </**div**> </**template**> |

路由过程分析：



# 三、路由之间的参数传递

## 1.设参

通过路由表，设置路由参数

|  |
| --- |
| {  **path**:**'/Products/:mid'**,  **component**:Products } |

## 2.传参

通过router-link的to访问路径时携带参数

|  |
| --- |
| <**li role="presentation"**><**router-link to="/Products/1"**>手机</**router-link**></**li**> |

## 3.接参

在目标vue中，通过vue中的$route.params.mid

|  |
| --- |
| <**template**>  <**div**>  {{**id**}}  {{$route.params.id}}  家电  </**div**> </**template**>  <**script**> **export default** {  **name**: **"Products.vue"**,  data(){  **return** {  **id**:**this**.**$route**.**params**.id  }  }  } </**script**>  <**style scoped**>  </**style**> |
|  |

# 四、路由跳转的两种方式

## 1.通过HTML中的路由<router-link to="路由地址">链接进行跳转

<**li role="presentation"**><**router-link to="/Products/1"**>手机</**router-link**></**li**>

## 2.通过js实现路由的跳转

this.$router.push("/Products/1");

# 五、Vue中组件样式表的作用范围

如果vue组件中的style标签没有带上 scoped属性，那么这个style的样式将会作用在整个页面中，而不是当前组件中，那么因此加上scoped，让样式只作用在当前组件中。

# 六、关于资源打包

开发一个vue项目，标准的步骤：

1）用vue-cli拉取一个项目骨架

2）安装依赖

3）使用npm run dev 进入开发者模式

此时在开发中模式中，各种修改都能看到实时效果，那么这些内容实际上是由vue-cli进行打包并发布在node.js上的。那么，最后开发完以后，要上生产，这些资源是需要我们手动部署在自己的服务器上的。那么，哪些资源是需要我们手动部署的，因此，通过npm run build命令来构建资源。

4）使用npm run build命令来构建资源。

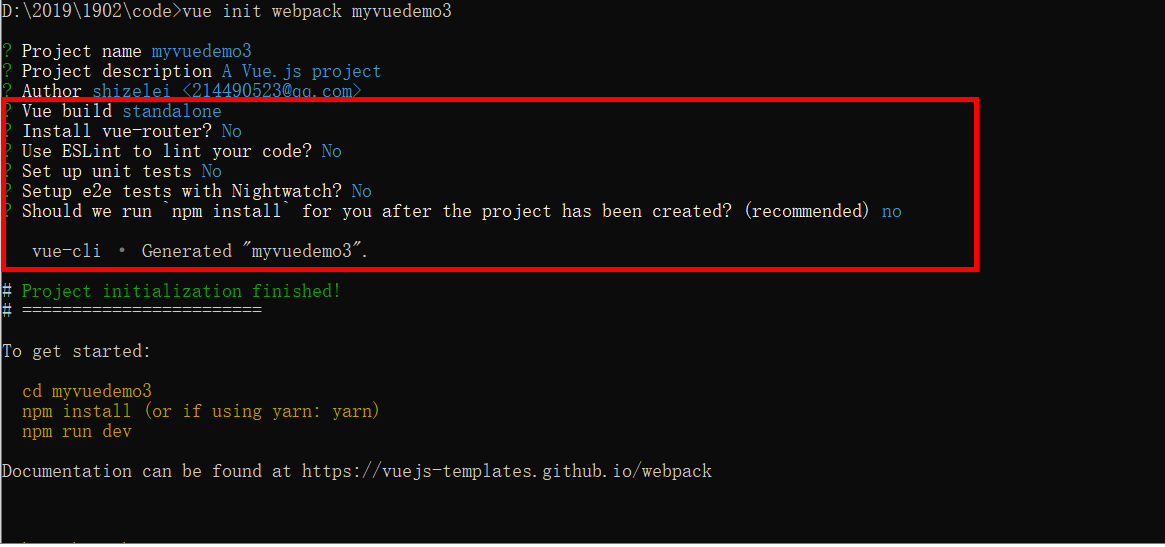
会产生一个dist文件夹。 里面包含静态资源。

因此，在开发环境下，要想使用静态资源，首先通过npm run build命令来创建dist文件夹，然后将静态资源放入，再在vue的代码中，去该/dist/文件夹内取资源即可。

# 七、创建Webpack项目

## 1.下载和使用webpack骨架

vue init webpack myvuedemo3



## 2.安装路由模块、element-ui模块、axios模块

# 安装 vue-router

npm install vue-router --save-dev

# 安装 element-ui

npm i element-ui -S

# 安装axios

npm install --save axios vue-axios

安装完以后，记得安装依赖： npm install

## 3.配置路由

1） 创建路由表

在src下创建router文件夹，里面创建index.js,此时该文件就是一个路由表

|  |
| --- |
| **import** Vue **from 'vue' import** Router **from 'vue-router'  import** Login **from '../views/Login'** *// 安装路由* Vue.use(Router);  *// 配置路由* **export default new** Router({  **routes**: [  {  *// 路由路径* **path**: **'/Login'**,  *// 路由名称* **name**: **'Login'**,  *// 跳转到组件* **component**: Login  }  ] }); |

2）在main.js中配置路由表

|  |
| --- |
| *// The Vue build version to load with the `import` command // (runtime-only or standalone) has been set in webpack.base.conf with an alias.* **import** Vue **from 'vue' import** App **from './App' import** VueRouter **from 'vue-router' //全局使用路由模块** *// 导入上面创建的路由配置表* **import** router **from './router'** Vue.use(VueRouter); */\* eslint-disable no-new \*/* **new** Vue({  **el**: **'#app'**,  *// 配置路由* router,  **components**: { App },  **template**: **'<App/>'** }) |

## 4.在App.vue中创建路由视图

|  |
| --- |
| <**template**>  <**div id="app"**>  <**router-view**></**router-view**>  </**div**> </**template**> |

## 5.在项目中使用element-ui模块