**一.初始化**

**1.vue2也有自己的脚手架工具vue-cli,首先你得全局安装vue-cli:**

$npm install -g vue-cli

**2.使用vue-cli初始化项目,初始化项目时vue-cli官方(https://www.npmjs.com/package/vue-cli)已经给提供了几种模板供大家选择，一般就用webpack**

$vue init  webpack-simple myProject

**3.上一步执行完毕，进入到你的项目里：**

cd myProject

**4.然后安装项目依赖的包，也就是在package.json文件里处在"devDependencies"下的包**

$npm i

**5.再安装生产环境下需要用到的包，也就是在package.json文件里处在"dependencies"下的包**

$npm i --production

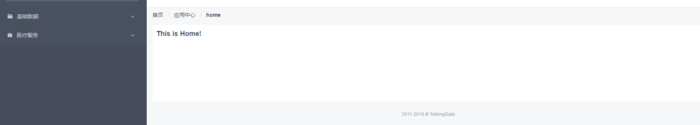
**6.就紧接着再运行，这一步依赖于package.json文件里"scripts"下的内容**

$npm run dev

**7.就会看到浏览器里面自动打开一个页面本地页面 localhost：8080**

此时一个简单的vue2 demo已经搭建成功了！

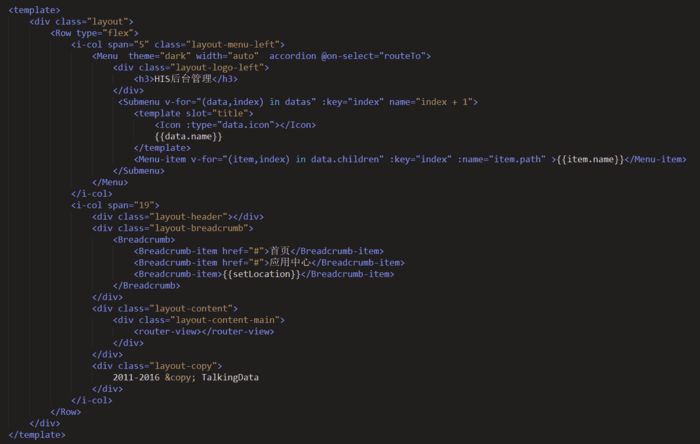
**二.搭建完整的项目**



页面展示

首先得新建一个项目，步骤同简单demo的初始化。

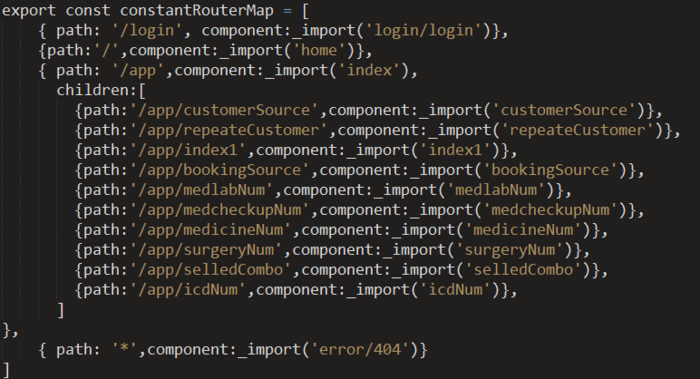
搭建项目，首先构建你的整体框架，此项目只有有一个入口页面，左侧菜单始终显示在页面上，而后根据理由跳转决定右侧显示和内容



入口页面代码

其中menu部分可以根据权限动态生成，而中间变动的部分在<router-view>标签内显示。

**1.路由**



router.js

在调用时需要写为:<router-link to='/baseData/dictionary'></router-link>

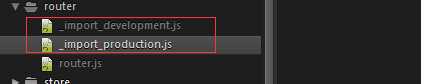
很多人在这里会想到用懒加载来加载组件，代码如下：

{path:'/medicalSvc',name:'医疗服务',component:resolve => require(['../components/medicalData/medicalServe'], resolve),},

但是如果你构建的是系统类项目，有上百个路由，这样的话 当你用webpack hot进行代码热更新的时候，速度就会超级慢，此时解决方法就是让他在开发环境下不要懒加载，在生产环境下再进行懒加载。我们首先要定义一个变量

const \_import = require('./\_import\_' + process.env.NODE\_ENV);

process.env是读取系统环境变量的，比如你在启动服务的时候，设置环境变量为production或者development，那么在程序里面就可以通过process.env.NODE\_ENV获取。此时你就需要再写两个文件(注意文件的名称命名)分别导出不同环境下的文件目录如：module.exports = file => () => import('@/pages/' + file + '.vue')

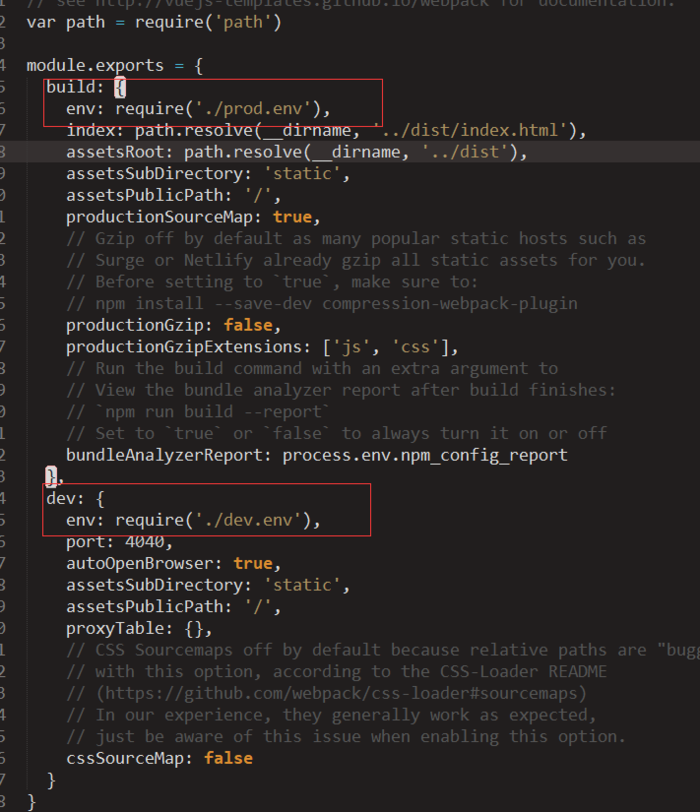


import文件

然后将组件路径定义为：

{path:'/app/customerSource',component:\_import('customerSource')},

那么又是怎么设置环境变量为production或者development值的呢，是在配置文件index.js下



index.js

另外注意：404页面一定要最后加载，如果放在constantRouterMap一同声明了404，它后面的所有页面都会被拦截到404，详细的问题见[addRoutes when you've got a wildcard route for 404s does not work](https://link.jianshu.com/?t=https://github.com/vuejs/vue-router/issues/1176)

**2.权限控制**



main.js

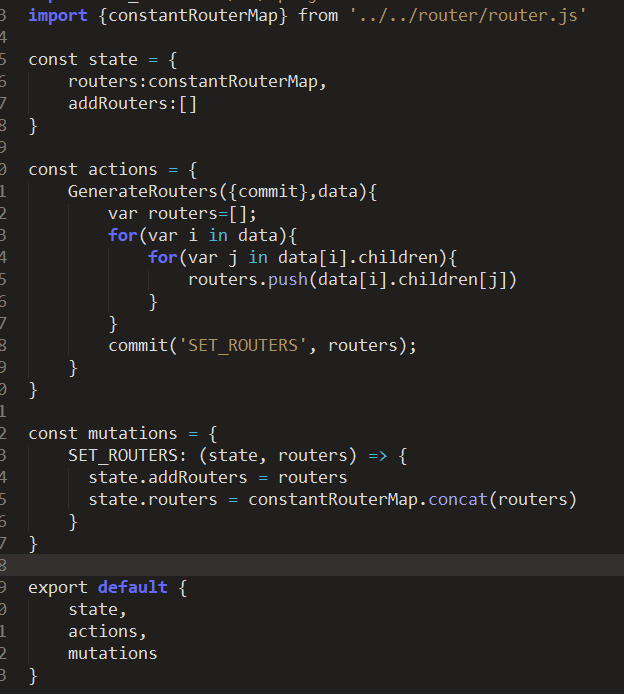
这里使用了router.beforeEach注册一个全局的before钩子，来在导航触发之前判断是否有权限进入改导航页面。其中的“token”是用户登录成功，有服务器端返回的标识，如果它存在，则证明用户已经登录，所以

1.当其再访问登录页面（to.path === '/login'）时，可直接重定向到首页。

2.当其访问其他路由时，需要判断用户是否具有访问这些路由的权限，首先将用户所有的权限转换城路由挂载到router上，接着当访问路由上未挂载的页面时便会定向到404页面，访问挂载上的页面时，则会正常跳转。

3.设置白名单是避免一些页面浪费时间做重定向，比如404，login.

4.注意：这里还有一个小hack的地方，就是router.addRoutes之后的next()可能会失效，因为可能next()的时候路由并没有完全add完成，我们简单的通过next(to)巧妙的避开这个问题

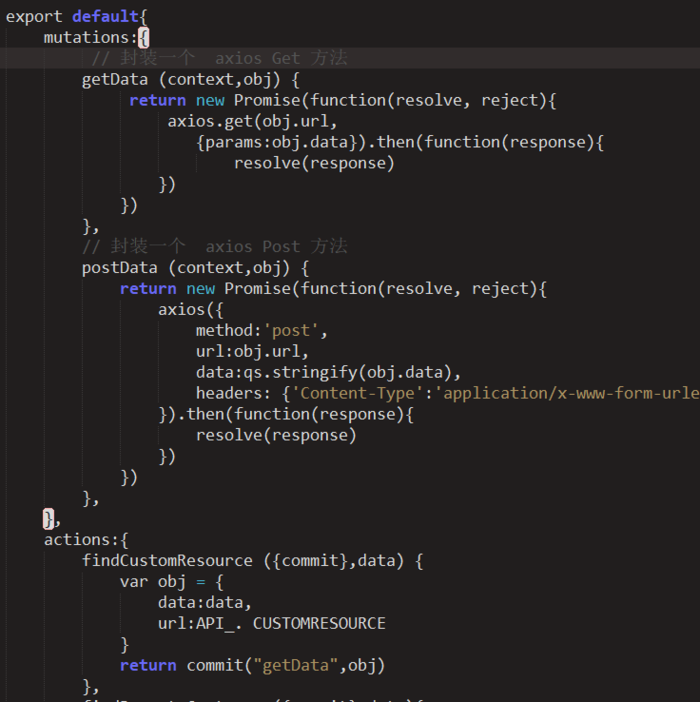


permission.js

permission.js里面代码只干了一件事情：那就是将用户获得的权限转化成该用户可以访问的路由。具体的转换方法要看后台返回什么样的数据。

**3.http服务--axios**

先得安装axios : npm install axios --save



http.js

Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中，我在项目中用Promise对象对其进行了封装，目的是为了保证异步书写代码时，数据这间的衔接性。axios既可以通过发送post，get请求也可以通过API访问服务器。

**4.vuex**

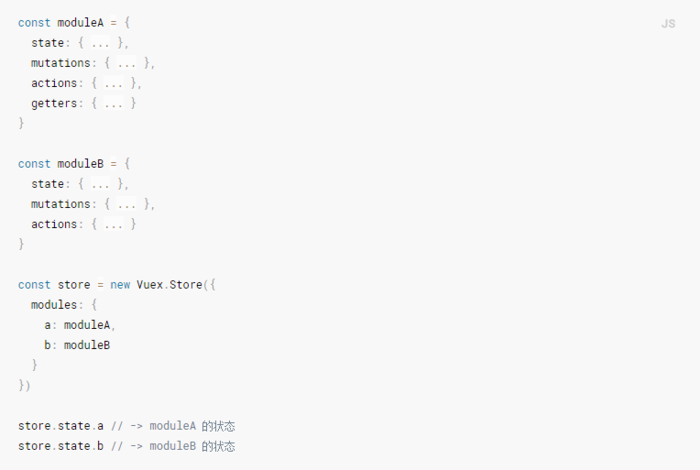
Vuex 是一个专为 Vue.js 应用程序开发的**状态管理模式**。个人理解vuex就相当于angular里面的service，用来存放状态和数据

一个项目里面有一个store存储器,在没有store以前，我们的vue组件可以理解成由state、view、action组成，三者之间的联系是@**state(data)**，驱动应用的数据源；**@view(template)**，以声明方式将**state**映射到视图；@**actions(methods)**，响应在**view**上的用户输入导致的状态变化。但是当你的好多组件要共享一个数据时，你得通过层层嵌套的父子组件来获取并修改数据，这样做太繁琐了，所以就想到要将多个组件共享的数据提出来，以一个全局单例模式管理，这便有了vuex。

store存储器里包含

new Vuex.Store({               state:{...},mutations:{...},actions:{...},getters:{...}               });

但是由于使用单一状态树，当你的项目足够大时，所有的状态都集中在一起，会 导致代码臃肿，不易维护。于是 Vuex 允许我们将 store 分割成**模块（module）**。每个模块拥有自己的 state、mutation、action、getter。

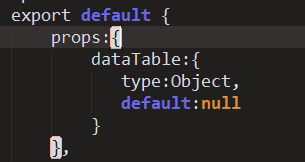


**5.组件之间的传值**

父组件向子组件传值：通过为子组件绑定属性

父：<table-hospital :dataTable="dataTable" :multiple="multiple" @changePage='changePage'></table-hospital>

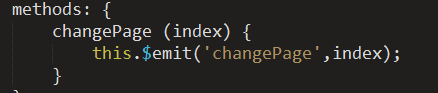
子：



子组件

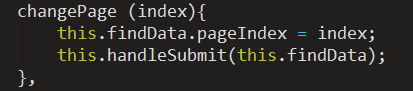
子组件向父组件传值：通过向上广播传值

子：



子组件

父：<table-hospital :dataTable="dataTable" :multiple="multiple" @changePage='changePage'></table-hospital>



父组件

非父子组件之间通信时，（1）在简单的场景下，可以使用一个空的 Vue 实例作为：

varbus =newVue()，

// 触发组件 A 中的事件

bus.$emit('id-selected',1)

// 在组件 B 创建的钩子中监听事件

bus.$on('id-selected',function(id){.............. ...})

（2）可以通过vuex状态管理模式来实现。

**6.jQuery的引用**

(1).用import的房是引入，那么哪个压面用到$,哪个页面就得import jquery;

(2)在index.html页面用<script>标签引入，这样的话人和组件都不需要再次引入，可以直接使用

总结：vue我也是初次接触，说的不正确的地方，欢迎大家留言指出，让我们在前端的道路上 互相伤害！