# Vue\_07

# 目标

* 使用vue-router构建单页面应用

# 一、路由

## 1.1、路由概念

路由的本质就是一种对应关系，根据不同的URL请求，返回对应不同的资源。那么url地址和真实的资源之间就有一种对应的关系，就是路由。

路由分为：后端路由和前端路由

* 后端路由：由服务器端进行实现并实现资源映射分发(路由与路由之前切换会刷新页面)
* **前端路由**：根据不同的事件来显示不同的页面内容，取事件与事件处理函数之间的对应关系(前端路由切换可能不会有网络请求存在，也不会有刷新页面)

**SPA**(Single Page Application)单页面应用程序,基于前端路由而起：整个网站只有一个页面，通过监听地址栏中的变化事件，来通过Ajax局部更新内容信息显示、同时支持浏览器地址栏的前进和后退操作。

## 1.2、前端路由实现

前端浏览器路由也有2种模式:

* hash模式(锚链接)

hash模式兼容性很强，刷新浏览器，页面还会存在

hash模式，缺点：地址栏不优雅，有#存在，不利于seo，记忆困难

hash路由模式是这样的：http://xxx.abc.com/**#**/xx。 有带#号，后面就是hash值的变化。改变后面的hash值，它不会向服务器发出请求，因此也就不会刷新页面。并且每次hash值发生改变的时候，会触发hashchange事件。因此我们可以通过监听该事件，来知道hash值发生了哪些变化。

window.addEventListener('hashchange', ()=>{

// 通过 location.hash 获取到最新的 hash 值

console.log(location.hash);

});

* history模式

history模式它是html5提供，兼容性较差，刷新页面，页面会404，需要服务器端配置支持

优点：地址栏更优雅，方便记忆，有利于有seo

HTML5的History API为浏览器的全局history对象增加了该扩展方法。它是一个浏览器（bom）的一个接口，在window对象中提供了onpopstate事件来监听历史栈的改变，只要历史栈有信息发生改变的话，就会触发该事件。

history.pushState({},title,url); // 向历史记录中追加一条记录

history.replaceState({},title,url); // 替换当前页在历史记录中的信息。

window.addEventListener('popstate', function(event) {

console.log(event)

})

注：浏览器地址没有#， 比如(http://localhost:3001/a); 它也一样不会刷新页面的。但是url地址会改变。但它在服务器没有配置的情况下，不能手动刷新，否则返回404页面

## 1.3、Vue Router

网址：<https://router.vuejs.org/zh/>

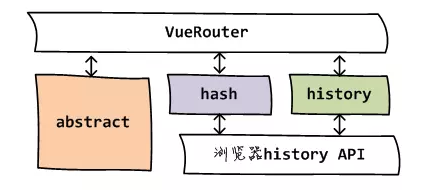
### 1.3.1、介绍

Vue Router 是 Vue.js 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。包含的功能有：

* 嵌套的路由/视图表
* 模块化的、基于组件的路由配置
* 路由参数、查询、通配符
* 基于 Vue.js 过渡系统的视图过渡效果
* 细粒度的导航控制
* 带有自动激活的 CSS class 的链接
* HTML5 历史模式或 hash 模式，在 IE9 中自动降级
* 自定义的滚动条行为

**vue-router路由模式:**

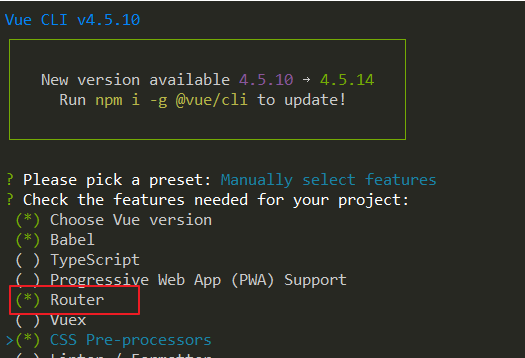
* hash: 使用 URL hash 值来作路由。支持所有浏览器，包括不支持 HTML5 History的浏览器。
* history: 依赖 HTML5 History API 和服务器配置。查看 HTML5 History 模式。
* abstract: 支持所有 JavaScript 运行环境，如 Node.js 服务器端。如果发现没有浏览器的 API，路由会自动强制进入这个模式。



### 1.3.2、安装

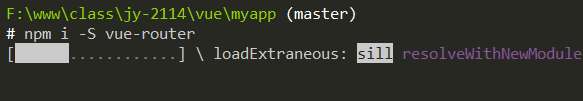
如果在vue-cli创建项目时没有勾选上vue-router选项，此时就需要手动的来安装它

使用vue-cli创建项目时，可以勾选上路由的选项，如果勾选上，则它就会自带路由



如果没有勾选，则需要手动的来安装一下

npm i -S vue-router



### 1.3.3、Vue Router的基本使用

Vue Router的基本使用步骤：

* 引入相关库文件
* VueRouter引入到Vue类中 以插件的方式引入
* 定义路由组件规则并创建路由实例
* 把路由挂载到Vue根实例中
* 添加路由组件渲染容器

# 路由文件

import Vue from 'vue'

import VueRouter from 'vue-router'

// 插件，vue允许开发插件，插入的

Vue.use(VueRouter)

# 定义路由规则

const routes = [

{ path: '/foo', component: Foo },

{ path: '/bar', component: Bar }

]

# 创建路由实例

const router = new VueRouter({

routes

})

# 挂载根实例

// 记得要通过 router 配置参数注入路由，

// 从而让整个应用都有路由功能

const app = new Vue({

router

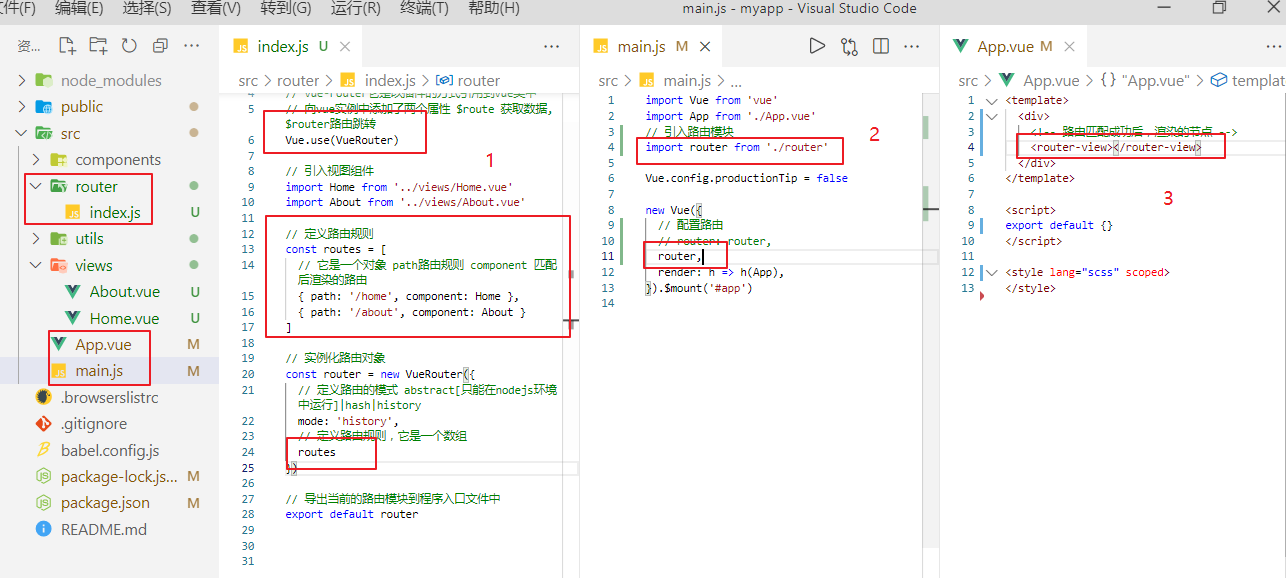
}).$mount('#app')

# html

<div>

<router-view></router-view>

</div>



### 1.3.4、声明式导航

它就是先在页面中定义好跳转的路由规则，vueRouter中通过 router-link组件来完成

<router-link to="path"></router-link>

# to 要跳转到的路由规则 string|object

to=”users”

:to=”{path:’path’}”



### 1.3.5、编程式导航

编程式导航就是通过js来实现路由跳转

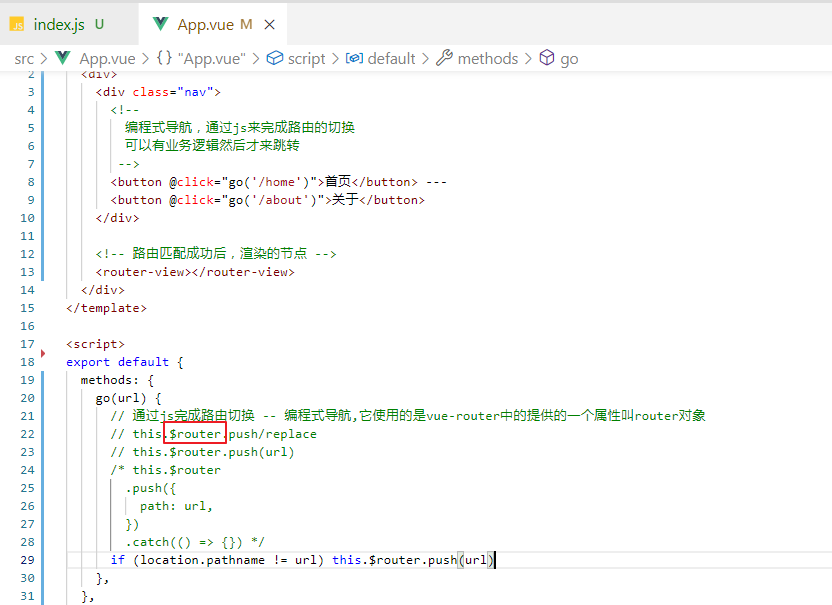
this.**$router**.push("/login");

this.$router.replace({ name:'user' , params: {id:123} });

this.$router.push({ path:"/login" });

this.$router.push({ path:"/login",query:{username:"jack"} });

this.$router.go( n );//n为数字 负数为回退



### 1.3.6、路由重定向/404

用户在访问地址A的时候，强制用户跳转到地址C ，从而展示特定的组件页面。

var router = new VueRouter({

// routes是路由规则数组

routes: [

// 每个路由规则都是个配置对象，至少有path和component两个属性

// path表示当前路由规则匹配的hash地址

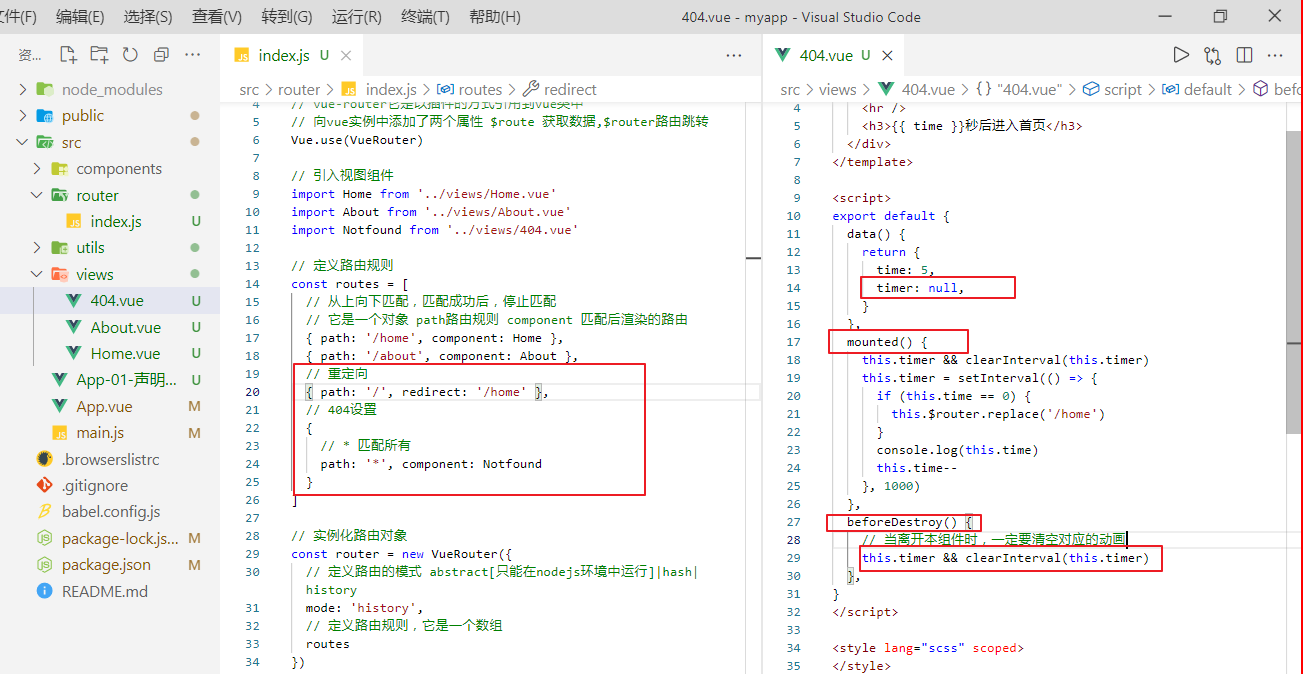
// component表示当前路由规则对应要展示的组件

{ path: '/', redirect: '/user' },

{ path: '/user', component: User }

]

})



### 1.3.7、嵌套路由

嵌套路由最关键在于理解子级路由的概念：

比如我们有一个/users的路由，那么/users下面还可以添加子级路由，如:

/users/index、/users/add等等，这样的路由情形称之为嵌套路由。

routes: [

{

path: "/user",

component: User,

//通过children属性为/user添加子路由规则

children:[

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

{ path: "/user/add", component: Add },

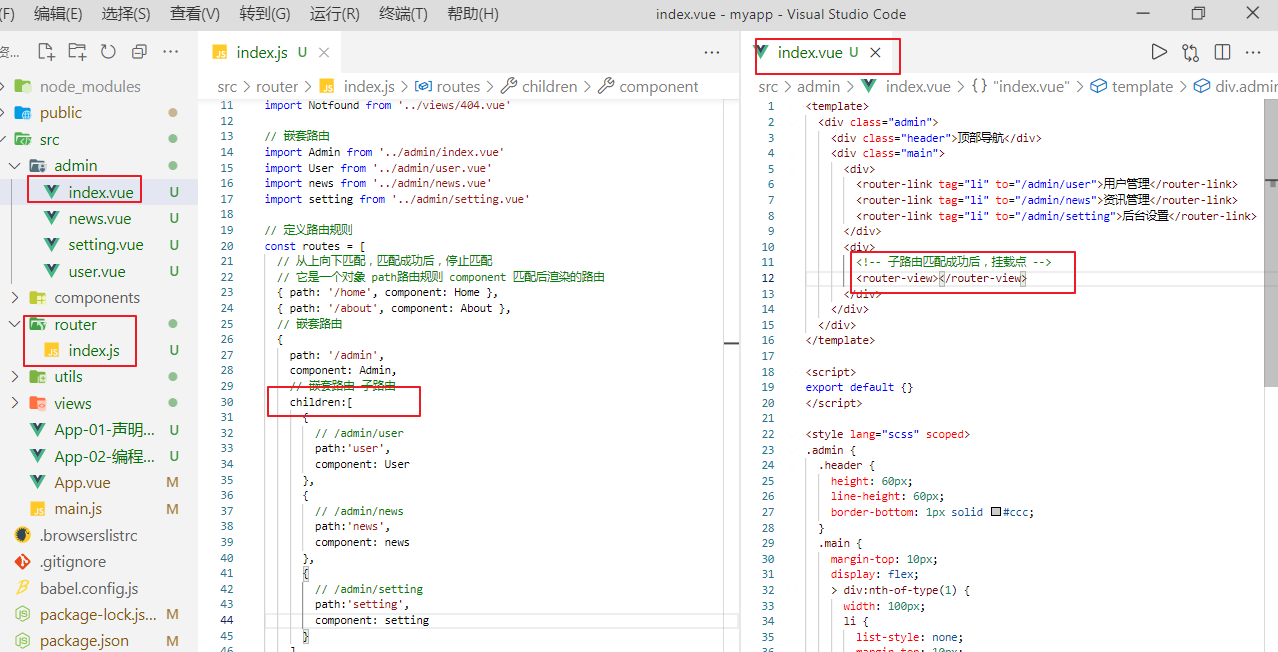
]

}

]

# 需要在 User组件中定义一个router-view 用于嵌套路由的渲染显示

<router-view></router-view>



### 1.3.8、动态路由匹配

所谓动态路由就是路由规则中有部分规则是动态变化的，不是固定的值，需要去匹配取出数据(即路由参数)。

// 传递参数id

var router = new VueRouter({

// routes是路由规则数组

routes: [

{ path: '/user/:id', component: User },

]

})

// 组件中获取id值

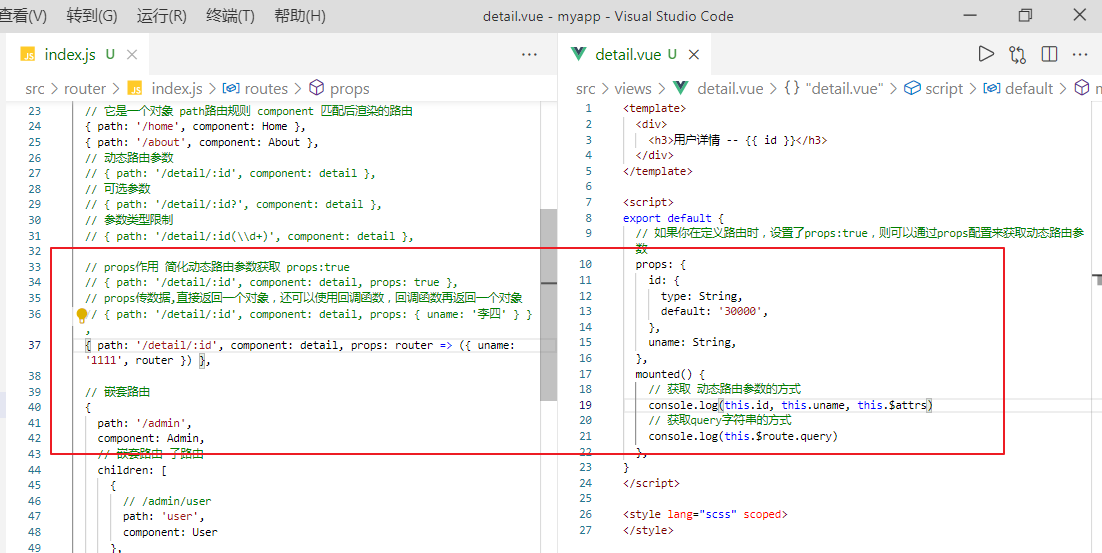
const User = {

template: '<div>User ID is {{$route.params.id}}</div>'

}

在vue中的路由传递参数的3个形式

* 动态路由参数  根据id查询详细，这样的需要所用 先定义再使用
* query字符串 ?id=10&name=lisi 搜索 无需定义，直接在地址栏中书写就可以
* props属性传值 隐式（数据传递时不让用户看到 - 埋点）



### 1.3.9、命名路由

通过一个名称来标识一个路由显得更方便一些，特别是在链接一个路由，或者是执行一些跳转的时候。

# 路由

const router = new VueRouter({

routes: [

{

path: '/user/:id',

name: 'user',

component: User

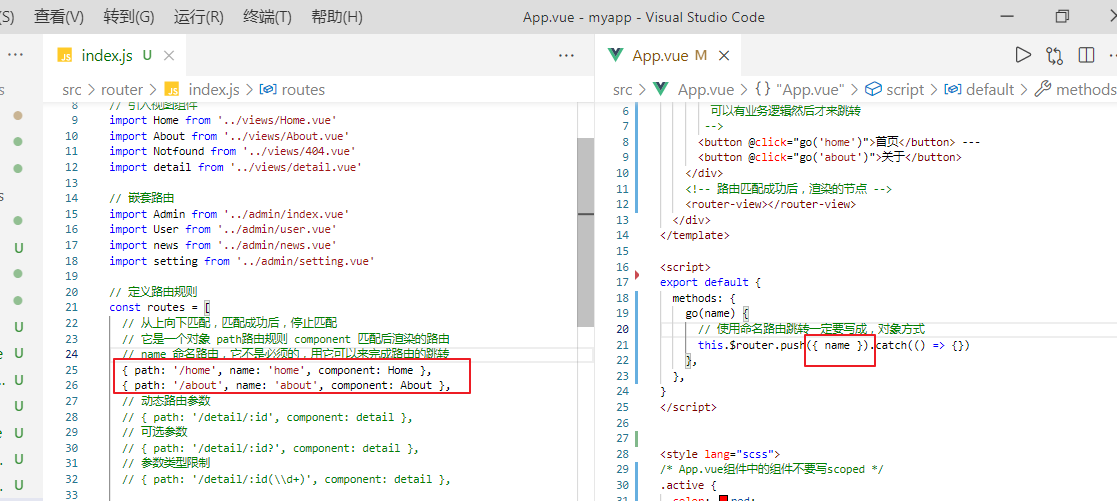
}

]

})

# 声明路由

<router-link :to="{ name: 'user', params: { id: 123 }}">User</router-link>



### 1.3.10、导航守卫

导航守卫就是路由跳转过程中的一些钩子函数，这个过程中触发的这些函数能让你操作一些其他的事儿的时机，这就是导航守卫。

* **全局守卫**

# 全局前置守卫

//当一个导航触发时,触发前置守卫，

to: Route: 即将要进入的目标 路由对象

from: Route: 当前导航正要离开的路由

next: Function: 一定要调用该next方法，否则路由不向下执行。

router.beforeEach((to, from, next) => {

// next函数必须要调用，否则路由不会向下执行

**next()**

})

//全局解析守卫

router.beforeResolve((to, from, next) => {

next()

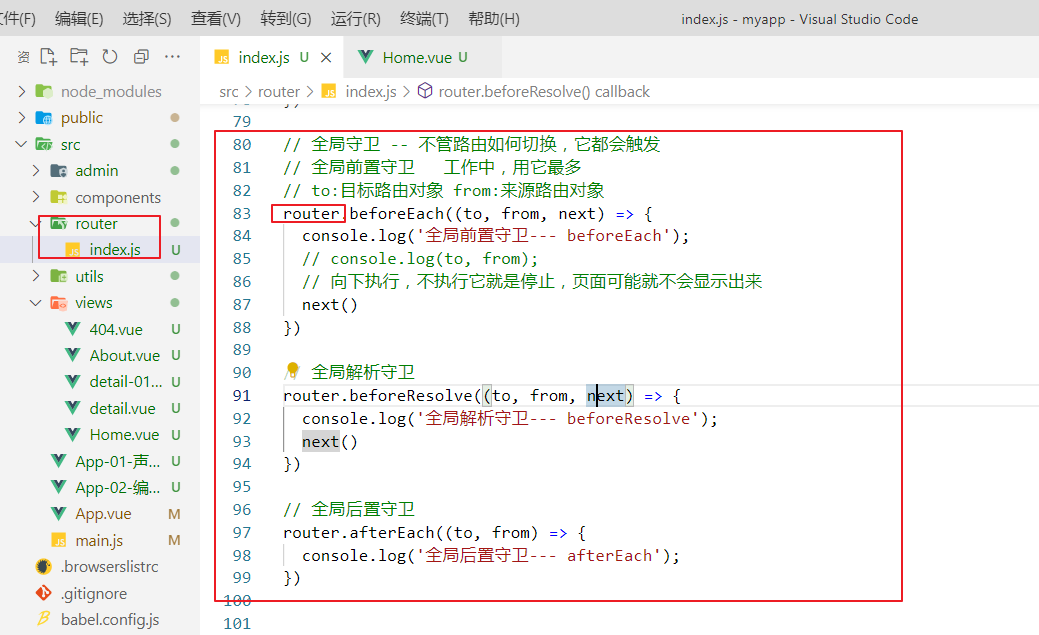
})

# 全局后置钩子

// 此钩子不会接受 next 函数也不会改变导航本身

router.afterEach((to, from) => {

})



* 路由独享的守卫

它的执行的顺序比beforeEach要晚，但早后beforeResolve

const router = new VueRouter({

routes: [

{

path: '/foo',

component: Foo,

beforeEnter: (to, from, next) => {

// ...

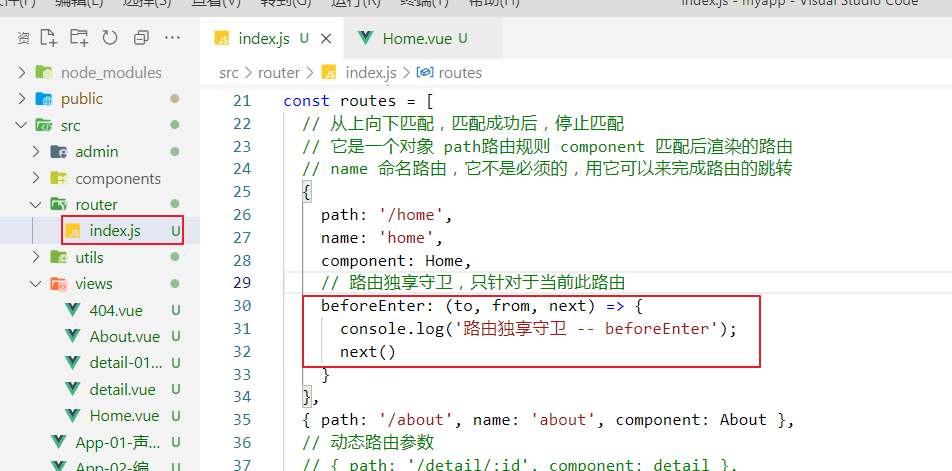
next()

}

}

]

})



* 组件内守卫

它比全局前置要慢，比路由独享要慢，快于全局解析路由守卫

你可以在路由组件内直接定义以下路由导航守卫

const Foo = {

template: `...`,

beforeRouteEnter (to, from, next) {

// 在渲染该组件的对应路由被 confirm 前调用

// 不！能！获取组件实例 `this`

// 因为当守卫执行前，组件实例还没被创建

},

beforeRouteUpdate (to, from, next) {

// 在当前路由改变，但是该组件被复用时调用

// 举例来说，对于一个带有动态参数的路径 /foo/:id，在 /foo/1 和 /foo/2 之间跳转的时候，

// 由于会渲染同样的 Foo 组件，因此组件实例会被复用。而这个钩子就会在这个情况下被调用。

// 可以访问组件实例 `this`

},

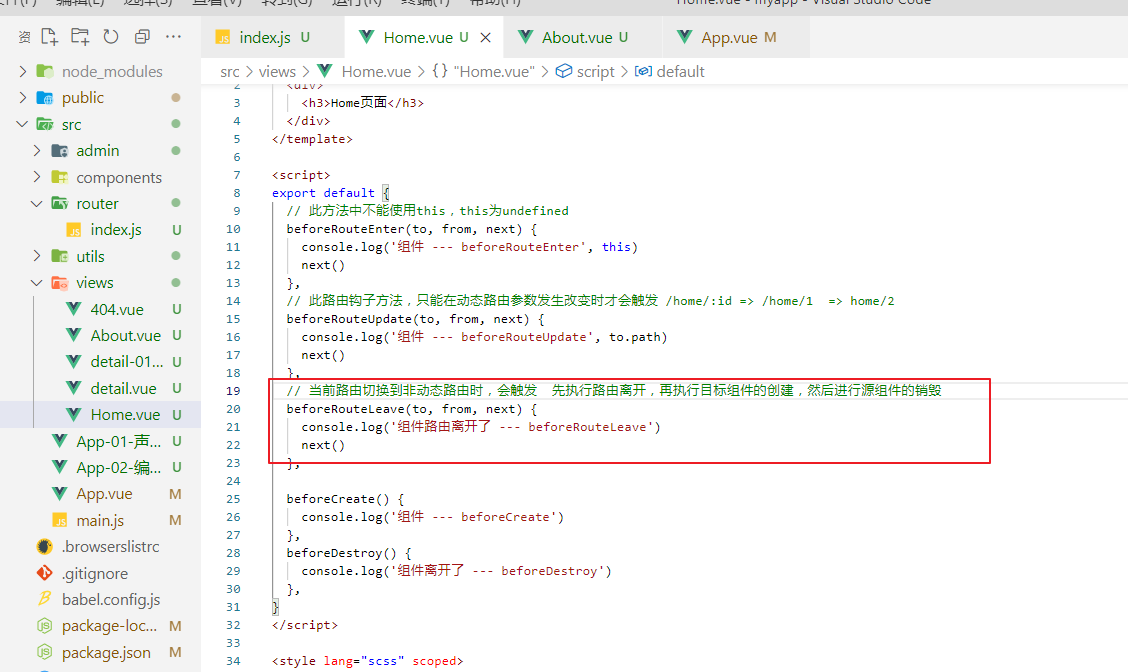
beforeRouteLeave (to, from, next) {

// 导航离开该组件的对应路由时调用

// 可以访问组件实例 `this`

}

}



### 1.3.11、路由元信息

定义路由的时候可以配置 meta 字段：

const router = new VueRouter({

routes: [

{

path: '/admin',

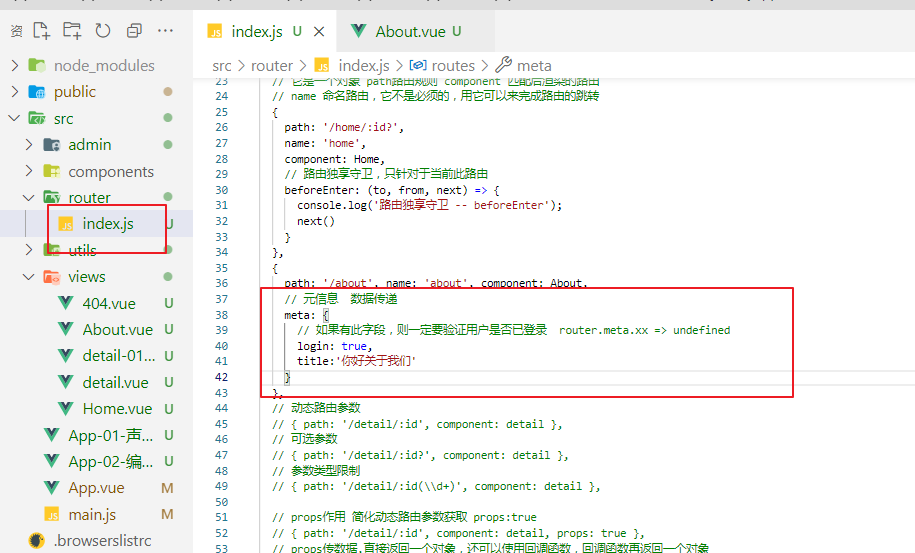
component: Admin,

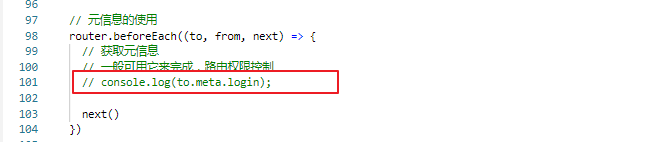
meta: {login:true}

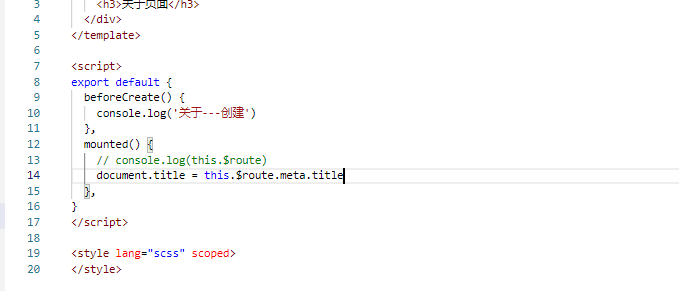
}

]

})







### 1.3.12、路由懒加载代码拆分

当打包构建应用时，js包会变得非常大，影响页面加载。如果我们能把不同路由对应的组件分割成不同的代码块，然后当路由被访问的时候才加载对应组件，这样就更加高效了。

const Foo = () => import(/\* webpackChunkName: "group-foo" \*/ './Foo.vue')

const router = new VueRouter({

routes: [{ path: '/foo', component: Foo }]

})

# 二、千锋电影项目

## 2.1、项目效果



## 2.2、开发项目流程

产品立项 (需求分析、技术选型、项目人员确定)

产品原型 (设计产品原型图— 进行ui设计)

项目开发 (前端 与 后端)

项目测试

项目上线

## 2.3、开发环境

开发工具：Vscode

开发环境：Window/Linux/Mac

项目运行环境：node v10.x以上版本

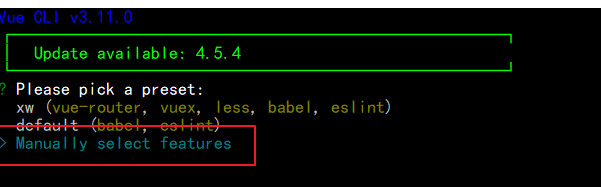
Vue脚手架： vue-cli 4.x

代码版本工具：Git

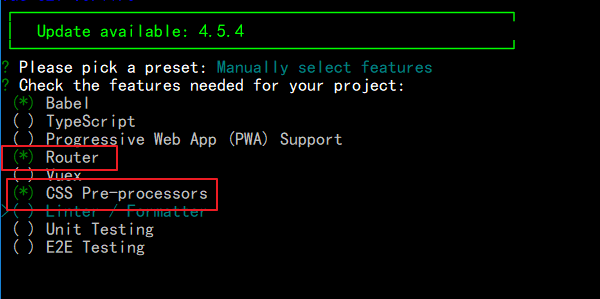
## 2.4、初始化项目

使用vue脚手架创建项目

选择手动模式



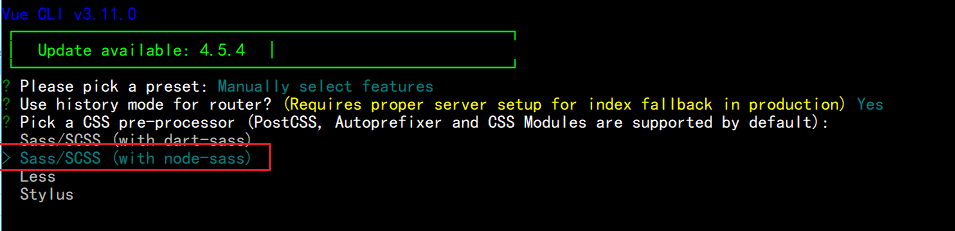
安装vue扩展



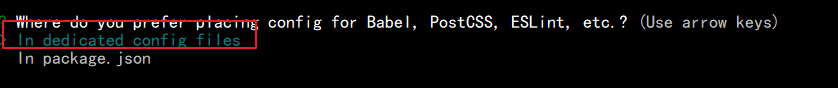
路由模式



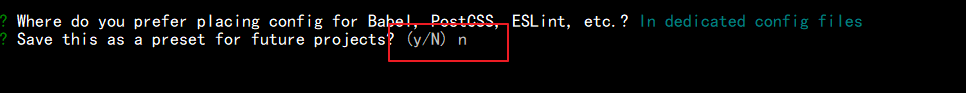
css预处理



每个配置都用单文件设置



不保存刚刚的安装配置



创建完成后，cd进入到项目中

## 2.5、vant组件库

网址：<https://vant-contrib.gitee.io/vant/#/zh-CN/>

## 2.6、路由规划

如果项目中所有的路由都写在入口文件中，那么将不便于编写项目和后期维护。因此路由需要进行模块化处理。

可以先行添加以下几个空的路由模块：

* 电影模块
* 影院模块
* 电影资讯
* 个人中心模块

## 2.7、网络请求

项目中使用axios进行网络请求，vue脚手架默认没有安装，需要自行安装才能使用。

* 安装ss

npm i -S axios

* 设置统一请求域名

// 导入axios

import axios from "axios"

// 设置请求的基础域名

axios.defaults.baseURL = 'https://api.iynn.cn/film/api/v1/'

* 反向代理可以进行跨域解决

在项目根目录下面创建一个vue.config.js文件，写下如下代码

module.exports = {

devServer: {

overlay: false,

// vue项目代理请求

proxy: {

// 规则

// axios中相对地址开头的字符串

'/api': {

// 把相对地址中的域名 映射到 目标地址中

// localhost:8080 => localhost:3000

target: 'http://localhost:3000',

// 修改host请求的域名为目标域名

changeOrigin: true,

// 请求uri和目标uri有一个对应关系

// 请求/api/login ==> 目标 /v1/api/login

pathRewrite: {

'^/api': '/v1/api'

}

}

}

}

}

# 三、电影模块开发

## 3.1、列表顶部导航

* 创建电影列表顶部导航组件src/components/FilmListTopNav.vue

<template>

<nav>

<ul>

<router-link tag="li" to="/film/nowplaying" active-class="active">

<span>正在热映</span>

</router-link>

<router-link tag="li" to="/film/comingsoon" active-class="active">

<span>即将上映</span>

</router-link>

</ul>

</nav>

</template>

<style lang="scss" scoped>

nav {

width: 100%;

height: 50px;

line-height: 50px;

border-bottom: 1px solid #ccc;

ul {

display: flex;

text-align: center;

li {

flex: 1;

display: flex;

justify-content: center;

&.active {

span {

width: 40%;

border-bottom: 2px solid red;

display: block;

color:#fe5100;

}

}

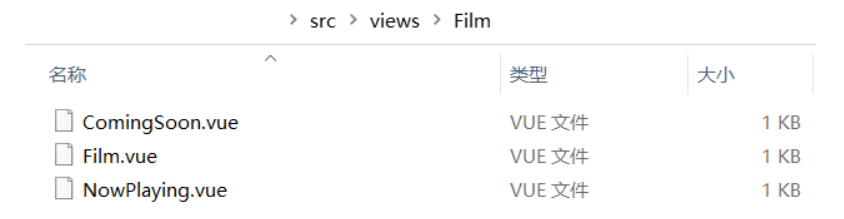
}

}

}

</style>

* 创建顶部导航对应的2个列表视图组件src/views/Film/目录下的Nowplaying.vue和Comingsoon.vue



* 在电影路由模块文件中定义上一步两个视图组件对应的路由（子路由）

// 电影路由

export default {

// 页面请求uri

path: '/film',

component: () => import('@/views/Film/Film'),

// 子路由 嵌套路由

children: [{

// /film/nowplaying

path: 'nowplaying',

component: () => import('@/views/Film/NowPlaying')

},

{

path: 'comingsoon',

component: () => import('@/views/Film/ComingSoon')

},

{

path: '/film',

redirect: 'nowplaying'

}

]

}

* 在电影视图组件src/views/Film/Film.vue中导入顶部导航组件

<template>

<div>

<FilmListTopNav></FilmListTopNav>

<router-view></router-view>

</div>

</template>

<script>

import FilmListTopNav from '@/components/FilmListTopNav'

export default {

components: {

FilmListTopNav

}

}

</script>

## 3.2、正在热映

修改src/views/Film/NowPlaying.vue文件

* 实现步骤

- 请求卖座网的正在热映电影数据

- 循环数据进行展示

**静态页面**

<div class="item">

<div class="left">

<img src="https://static.maizuo.com/pc/v5/usr/movie/fad9f1e9a5c38f3883d7f9b5e561e5fe.jpg?x-oss-process=image/quality,Q\_70"/>

</div>

<div class="middle">

<div>我和我的家乡</div>

<div>

<span>观众评分</span>

<span>7.6</span>

</div>

<div>主演：闫非 彭安宇 宁浩 徐峥 陈思诚</div>

<div>中国大陆 | 150分钟</div>

</div>

<div class="right">

<span>购票</span>

</div>

</div>

<style>

.item {

margin-top: 15px;

padding-bottom: 15px;

display: flex;

color: #797d82;

font-size: 13px;

border-bottom: 1px solid #ededed;

.left {

width: 77px;

height: 100px;

margin-left: 20px;

img {

width: 66px;

height: 100%;

}

}

.middle {

display: flex;

flex-direction: column;

justify-content: space-between;

width: 55%;

div:nth-of-type(1) {

color: #191a1b;

font-size: 16px;

}

div:nth-of-type(3) {

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap;

}

}

.right {

width: 15%;

display: flex;

align-items: center;

margin-right: 20px;

span {

border: 1px solid #ff5f16;

background: white;

color: #ff5f16;

border-radius: 2px;

height: 25px;

line-height: 25px;

font-size: 13px;

width: 50px;

text-align: center;

}

}

}

</style>

* **顶部菜单的吸顶效果**

实现思路：给头部导航添加一个class动态根据滚动条距离顶部的位置控制是否吸顶

<template>

<div>

<FilmListTopNav :class="fixed ? 'fixed' : ''" />

<router-view></router-view>

</div>

</template>

<script>

import FilmListTopNav from "@/components/FilmListTopNav";

export default {

data() {

return {

// 默认不吸顶

fixed: false,

};

},

components: {

FilmListTopNav,

},

mounted() {

window.addEventListener(

"scroll",

(e) => {

let top = document.documentElement.scrollTop;

if (top > 150) {

// 吸顶

this.fixed = true;

} else {

// 还原

this.fixed = false;

}

},

true

);

},

};

</script>

<style lang="scss" scoped>

.fixed {

position: fixed;

top: 0;

background: #ffffff;

}

</style>

## 3.3、即将上映

参考正在热映完成

## 3.4、电影详情

电影详情页面需要依据上一页传递过来的数据（电影ID，电影ID是可变的数字）来显示对应电影的具体信息（动态路由参数）

**实现步骤**

* + 创建电影详情组件
  + 定义对应的电影详情路由模块并注册
  + 依据动态参数filmId去获取对应的电影信息并展示
  + 其他必要处理

### 3.4.1、基本信息展示

**实际操作**

* 创建电影详情组件

创建位置位于src/views/Film/Detail.vue，基本结构代码参考如下

<template>

<div class="detail">

<div class="img">

<img src="https://static.maizuo.com/pc/v5/usr/movie/5607daea349314f6e67004348fd49e6c.jpg?x-oss-process=image/quality,Q\_70" />

</div>

<div>

<div>我和我的家乡</div>

<div>剧情 | 喜剧</div>

<div>2020-10-01上映</div>

<div>中国大陆 | 120 分钟</div>

<div>

电影《我和我的家乡》定档2020年国庆，延续《我和我的祖国》集体创作的方式，由张艺谋担当总监制，宁浩担任总导演，张一白担任总策划，宁浩、徐峥、陈思诚、闫非&彭大魔、邓超&俞白眉分别执导五个故事。

</div>

</div>

</div>

</template>

<style lang="scss" scoped>

.detail {

.img {

width: 100%;

height: 260px;

img {

width: 100%;

height: 100%;

}

}

}

</style>

* 定义对应的电影详情路由模块并注册

{

path: '/film/:id,

name: 'filmDetail',

component: () => import('@/views/film/Detail')

}

* 动态展示
* 解决上映时间戳转年月日的问题

# 安装

npm i -S moment

# 在当前组件中引入moment模块

import moment from 'moment'

# 创建过滤器处理时间戳

filters: {

parsePremiereAt: function(value){

// 时间戳的单位是秒，需要乘以1000转化成毫秒进行处理

return moment(value \* 1000).format('YYYY-MM-DD')

}

}

# 在模板中使用过滤器

<div>{{ film.premiereAt | parsePremiereAt }}上映</div>

### 3.4.2、底部菜单移除

* **步骤**

- 定义事件总线

- src/main.js

- 在详情中发布

- Detail.vue

- 在根组件中订阅

- App.vue

* **操作实现**
* 在src/main.js文件中实例化一个事件总线对象

Vue.prototype.eventBus = new Vue();

* 在详情组件中的created和beforeDestroy方法中完成发布命令

# 进入（创建完成）详情时隐藏导航，离开（销毁）详情时显示导航

export default {

......

created() {

this.eventBus.$emit("footernav", false);

},

beforeDestroy() {

this.eventBus.$emit("footernav", true);

},

}

* 在根组件中订阅详情中的发布并应用到导航上

①在data中定义is\_show，用于控制是否显示导航组件

②定义订阅操作，获取详情发送过来的数据并动态控制is\_show的值

created() {

this.eventBus.$on("footernav", (flag) => {

this.is\_show = flag;

});

}

## 3.5、影院列表

**步骤**

* 修改封装好的请求
* 添加影院数据请求地址
* 添加影院数据请求方法
* 在影院组件中调用封装好的方法并展示数据
* 编写样式
* 获取&展示数据

列表滚动加载数据插件：**better-scroll**

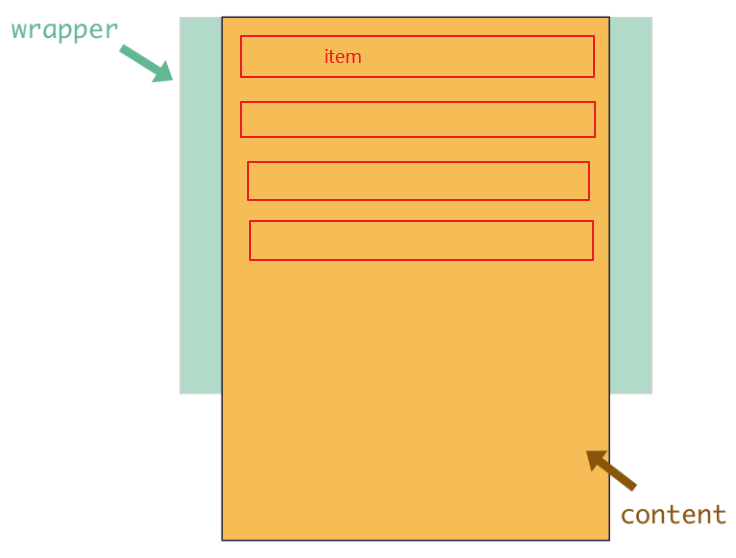
BetterScroll 是一款重点解决移动端（已支持 PC）各种滚动场景需求的插件。

better-scroll：<https://better-scroll.github.io/docs/zh-CN/guide/>

**安装**

npm i -S better-scroll

better-scroll运行原理图，绿色部分为 wrapper，也就是父容器，它会有固定的高度。黄色部分为 content，它是父容器的第一个子元素，它的高度会随着内容(item)的大小而撑高。那么，当 content 的高度不超过父容器的高度，是不能滚动的，而它一旦超过了父容器的高度，我们就可以滚动内容区了，这就是 BetterScroll 的滚动原理。。



<div class="scroll" :style="{height: height + 'px'}">

<ul>

<li></li>

</ul>

</div>

import BScroll from "better-scroll";

export default {

data() {

return {

height: 0,

bs: null

};

},

mounted() {

// 获取可视高度

this.height = document.documentElement.clientHeight - 100

this.$nextTick(() => {

this.bs = new BScroll('.scroll', {

// 激活上滑动的事件监听

pullUpLoad: true,

// 激活下滑的事件监听

pullDownRefresh: true,

// 默认情况下使用bs后，它会禁止浏览器的点击事件

click: true

})

// 上拉刷新

this.bs.on('pullingUp', () => {

console.log('上拉刷新')

// 获取数据

//本次pullup动作已经完成，继续准备下一次pullup

this.bs.finishPullUp()

})

this.bs.on('pullingDown', () => {

console.log('下拉刷新')

// 获取数据

//本次pulldown动作已经完成，继续准备下一次pulldown

this.bs.finishPullDown()

})

})

},

beforeDestroy() {

// 离开清理资源

this.bs = null

}

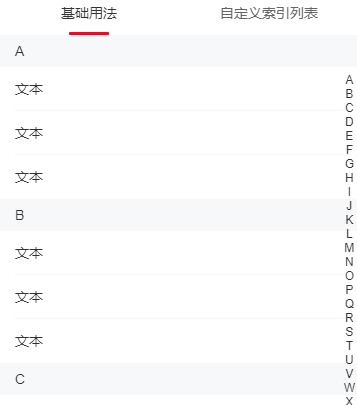
}

.scroll {

overflow: hidden;

}

## 3.6、城市列表开发



## 3.7、电影搜索