

# Vue学习第五天

---

## 反馈

---

## 回顾

---

## 锚链接与hash

---

锚链接是一种超链接，能快速滚动页面某个位置

1. 可以在url上直接修改hash
2. onhashchange能监听到hash改变
3. location.hash获取到当前页面的hash

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <title>Document</title>
9 </head>
10
11 <body>
12
13   <h2 id="top">这里是页面顶部</h2>
14   <br><br><br><br><br><br><br><br><br><br><br><br><br><br>
15   页面内容.....
16   <br><br><br><br><br><br><br><br><br><br><br><br><br><br>
17   <a href="#top">滚动顶部</a>
18   <h2 id="bottom">这里是页面底部</h2>
```

```
19 <script>
20     // 可以用onhashchange监听到hash改变
21     window.onhashchange = function () {
22         //获取当前页面的hash
23         console.log(location.hash)
24     }
25 </script>
26 </body>
27
28 </html>
```

## SPA与MPA

### 1. SPA(Single-Page Application) 单页应用

一个外壳页面和多个页面片段构成

1. 切换页面不会打开新的页面，URL中只改变hash(hash是实现SPA的方案之一)
2. 首屏加载慢，页面切换快
3. 适就用于做后台管理端

### 2. MPA(Multi-Page Applicatoin) 多页应用

多个完整的页面构成

1. 页面跳转会打开新的页面，URL改变
2. 首屏加载快，页面切换慢

## hash实现SPA

改变hash切换div显示

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-
7 scale=1.0">
8     <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
8 <title>Document</title>
9 <style>
10     .container {
11         display: flex;
12     }
13
14     .left {
15         width: 200px;
16         height: 800px;
17         background-color: #aaa;
18     }
19
20     .left li {
21         height: 50px;
22     }
23
24     .main>div {
25         display: none;
26     }
27 </style>
28 </head>
29
30 <body>
31
32     <div class="container">
33         <div class="left">
34             <ul>
35                 <li><a href="#tab1">导航1</a></li>
36                 <li><a href="#tab2">导航2</a></li>
37                 <li><a href="#tab3">导航3</a></li>
38             </ul>
39         </div>
40         <div class="main">
41             <div id="tab1">内容1</div>
42             <div id="tab2">内容2</div>
43             <div id="tab3">内容3</div>
44         </div>
45     </div>
46
47     <script>
48         window.onhashchange = function () {
49             document.querySelector('#tab1').style.display = 'none'
```

```

50     document.querySelector('#tab2').style.display = 'none'
51     document.querySelector('#tab3').style.display = 'none'
52     document.querySelector(location.hash).style.display = 'block'
53 }
54 </script>
55
56 </body>
57
58 </html>

```

## Vue路由基本使用

### [传送门](#)

Vue路由是用来构建单页应用, 不同hash显示不同的组件

使用方法, 大家只要**学会复制粘贴**就行。

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10
11 <body>
12     <div id="app">
13         <h1>Hello App!</h1>
14         <p>
15             <!-- 菜单 -->
16             <!-- 相当于<a href="#/foo"> -->
17             <router-link to="/foo">Go to Foo</router-link>
18             <router-link to="/bar">Go to Bar</router-link>
19         </p>
20         <!-- 内容 -->
21         <router-view></router-view>

```

```
22 </div>
23 <script src="https://unpkg.com/vue/dist/vue.js"></script>
24 <script src="https://unpkg.com/vue-router/dist/vue-router.js">
</script>
25
26 <script>
27   // 1. 声明组件
28   const Foo = {
29     template: '<div>foo</div>'
30   }
31   const Bar = {
32     template: '<div>bar</div>'
33   }
34
35   // 2. 定义路由
36   // hash和组件的映射关系
37   const routes = [{
38     path: '/foo',
39     component: Foo
40   },
41   {
42     path: '/bar',
43     component: Bar
44   }
45 ]
46
47   // 3. 创建 router 实例
48   // 实例化 路由对象,传递路由规则
49   const router = new VueRouter({
50     routes // (缩写) 相当于 routes: routes
51   })
52
53   // 4. 创建根实例
54   const app = new Vue({
55     el: '#app',
56     // 把路由 和Vue实例关联起来
57     router
58   })
59   //.$mount('#app') 相当于el:'#app'
60
61   // 现在,应用已经启动了!
62 </script>
```

```
63 </body>
64
65 </html>
```

## Vue路由高仿网易云音乐

1. 改变hash显示对应的组件
2. 这里组件就是一个图片

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8   </head>
9   <body>
10    <div id="app">
11      <!-- 导航栏 -->
12      <router-link to="/music">发现音乐</router-link>
13      <router-link to="/me">我的音乐</router-link>
14      <router-link to="/friend">朋友</router-link>
15      <router-link to="/download">下载客户端</router-link>
16      <!-- 内容区 -->
17      <router-view></router-view>
18    </div>
19    <script src="./lib/vue.js"></script>
20    <!-- 导入vue-router.js -->
21    <script src="./lib/vue-router.js"></script>
22    <script>
23      // 定义组件
24      const Music = { template: '' }
25      const Me = { template: '' }
26      const Friend = { template: '' }
27      const Download = { template: '' }
28
29      // 定义路由规则
```

```
30     const routes = [  
31       {  
32         path: '/music',  
33         component: Music  
34       },  
35       {  
36         path: '/me',  
37         component: Me  
38       },  
39       {  
40         path: '/friend',  
41         component: Friend  
42       },  
43       {  
44         path: '/download',  
45         component: Download  
46       }  
47     ]  
48     // 实例化路由对象，传递规则  
49     const router = new VueRouter({  
50       routes  
51     })  
52     //实例化Vue，传递路由对象  
53     const app = new Vue({  
54       el: '#app',  
55       router,  
56       data: {}  
57     })  
58   </script>  
59 </body>  
60 </html>  
61
```

## Demo-高级播放器-路由整合

### 实现步骤

1. 整合路由
  1. 改变hash显示对应的组件

2. 引入vue-router.js
  3. router-view显示组件
  4. 路由使用js里面那一大堆copy过来改
2. 整合组件
    1. 把index隔壁的那些文件，结构，样式全部都整到index.html
    2. 抽取为组件
  3. 为了晚上看代码简洁，删除了除index.html之外所有文件

## 编程式导航

### 传送门

编程式导航和声明式导航都是改变hash.

1. 编程式导航相当于是location.href
2. 声明式导航相当于是
3. [用法：](#)

```
1 router.push('地址')
2
3 <router-link to='/run'>去跑步</router-link>
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-
      scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8   </head>
9   <body>
10    <div id="app">
11      <!-- tab - nav 导航 声明式导航 -->
12      <router-link to="/music1">歌曲1</router-link>
13      <router-link to="/bar">歌曲2</router-link>
14      <router-link to="/run">边唱歌鞭炮</router-link>
15      <h2>编程式导航</h2>
16      <input type="button" value="点我去唱歌" @click="toSing">
```



```
17
18 <!-- tab - content -->
19 <router-view></router-view>
20 </div>
21 </body>
22 </html>
23 <!-- 放上面会影响页面的接在 js加载完毕之后页面是看不到的 -->
24 <script src="https://unpkg.com/vue/dist/vue.js"></script>
25 <script src="https://unpkg.com/vue-router/dist/vue-router.js">
  </script>
26 <script>
27   // 1. 定义组件 简化的写法
28   const Foo = { template: '<div>foo</div>' },
29   const Bar = { template: '<div>bar</div>' },
30   const run = { template: '<div>咋咋咋!!!!的跑!!!!</div>' },
31
32   // 2. 定义规则
33   // url和组件的对应关系
34   // const routes = [
35   const routers = [
36     { path: '/music1', component: Foo },
37     { path: '/bar', component: Bar },
38     { path: '/run', component: run },
39   ].
40
41   // 3. 把路由规则 设置给路由对象
42   const router = new VueRouter({
43     // routes // (缩写) 相当于 routes: routes
44     routes: routers // routers: routers
45   })
46
47   // 4. 创建和挂载根实例。
48
49   const app = new Vue({
50     el: '#app',
51     methods: {
52       // 跳转去唱歌
53       toSing(){
54         // console.log('唱歌去')
55         // 用路由对象跳转
56         // router.push('/bar')
57         router.push('/niubility')
```

```
58   _____.
59   _____.
60   router // router:router
61   _____.
62
63   // 现在，应用已经启动了！
64 </script>
65
```

## 动态路由匹配

### 传送门

组件从hash里获取参数

模式、匹配路径及获取参数如下图

模式	匹配路径	\$route.params
/user/:username	/user/evan	{ username: 'evan' }

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8   </head>
9   <body>
10    <div id="app">
11      <h1>Hello App!</h1>
12      <p>
13        <router-link to="/music1">歌曲1</router-link>
14        <router-link to="/bar/Joven">歌曲2</router-link>
15      </p>
16      <router-view></router-view>
17    </div>
18
```

```
19 <script src="https://unpkg.com/vue/dist/vue.js"></script>
20 <script src="https://unpkg.com/vue-router/dist/vue-router.js">
</script>
21 <script>
22   const Foo = { template: '<div>foo</div>' }
23   const Bar = { template: '<div>{{$route.params.name}}</div>' }
24
25   const routes = [{ path: '/music1', component: Foo }, { path:
'/bar/:name', component: Bar }]
26   const router = new VueRouter({
27     routes
28   })
29
30   const app = new Vue({
31     router,
32     methods: {
33       toSing() {
34         router.push('music1')
35       }
36     }
37   }).$mount('#app')
38 </script>
39 </body>
40 </html>
```

## Demo-歌曲搜索

黑云音乐



## 实现步骤

1. 输入关键字，回车/点搜索按钮，展示搜索结果组件
  1. 获取用户输入的关键字 v-model:keywords
  2. 事件 @keyup.enter/@click:queryMusic

### 3. 程式导航

1. router.push('/result')
2. 输入关键字，回车/点搜索按钮，搜索结果组件获取到关键字

#### 1. 动态路由匹配

1. `router.push('/result') => router.push('/result/keywords')`
2. 模式 `'/result' => '/result/:keywords'`
3. 确认组件获取到了关键字 Vue开发工具查看

## 注意点

1. js的方式改变hash，用的是程式导航
2. 给组件传递参数，用的是动态路由匹配

## 生命周期钩子 -created

最早能在created里面获取到data的属性

```
1 beforeCreate() {  
2   console.log(this.message)  
3 },  
4 created() {  
5   // 最早能在created里面获取到data的属性  
6   console.log(this.message)  
7  
8 },
```

## Demo-高级播放器-结果搜索

### 实现步骤

1. 当 result 组件创建出来之后（出现）之后，
  1. 使用生命周期钩子created
  2. 尽可能早一些执行的，让用户早一些看到数据
2. 获取传递过来的关键字 `this.$route.params.键`

3. 通过关键字调用接口， axios.get
4. 接口 `https://autumnfish.cn/search?keywords=海阔天空`
5. 数据获取到之后，渲染到页面上
  1. then
  2. v-for :musicList
6. mvid为0,不展示mv
  1. v-show条件渲染

## 注意点

由切换默认会移除dom，也就是组件重新创建。

## 过滤器基本使用

### 过滤器

过滤器，可被用于一些常见的**文本格式化**

白话：用来做文本格式化的

1. 定义的方式
  1. vue中filters:{}
    2. 过滤器是一个方法
    3. 过滤器默认接收一个参数，参数管道符所作用的数据
    4. 内部处理完毕之后，必须return一个值，这个值就是最终显示的数据
2. 使用
  1. `{{ 数据 | 过滤器 }}`
  2. `|` 管道符
3. 只能用于{{}}和v-bind表达式,大多数时候用于{{}}，作文本格式化。

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
```

```

7     <title>Document</title>
8   </head>
9   <body>
10    <div id="app">
11      <h2>公历生日:{{formatDate1}}</h2>
12      <h2>农历生日:{{formatDate2}}</h2>
13      <h2>过滤器的方式</h2>
14      <h2>公历生日:{{date1|formatDate}}</h2>
15      <h2>农历生日:{{date2|formatDate}}</h2>
16    </div>
17    <script src="./lib/vue.js"></script>
18    <script src="./lib/moment.js"></script>
19    <script>
20      const app = new Vue({
21        el: "#app",
22        data: {
23          date1: '2019-8-13',
24          date2: '2019-7-13'
25        },
26        computed: {
27          formatDate1(){
28            return moment(this.date1).format('YYYY.MM.DD')
29          },
30          formatDate2(){
31            return moment(this.date2).format('YYYY.MM.DD')
32          }
33        },
34        filters:{
35          formatDate(date){
36            return moment(date).format('YYYY.MM.DD')
37          }
38        }
39      });
40    </script>
41  </body>
42 </html>

```

## Demo-过滤器处理result中搜索的结果

## 实现步骤

### 1. 处理时间 毫秒数->4:30

#### 1. 添加过滤器 处理 时间

1. `{{item.duration | formatTime }}`
2. `filters:{ formatTime(time) }`

#### 2. 格式化时间的处理逻辑

1. 毫秒->秒
2. 算出分 60的整数倍 除
3. 剩余的部分作为秒 取余

### 2. 显多个歌手名称

#### 1. 添加过滤器 处理歌手

2. `{{ item.artists | formatSinger }}`
3. `filters:{ formatSinger(arr) }`

#### 2. 过滤器内部逻辑

1. 循环数组，获取歌手的数组
2. 歌手数组.join('/')

## 注意点

1. 时间从毫秒转为 时分秒，
  1. 先除 再取余
2. 过滤器的特点是格式化文本
3. 过滤器的使用 `|`
  1. 这个 `|` 也叫 管道符

## 单文件组件

1. 用一个文件能够包含组件的所有内容
  1. 样式
  2. 结构

### 3. 逻辑

## 2. `.vue`

### 3. 设置三个结构

1. 输入 `<vue>` 就能够自动生成

4. 后续的项目都会使用单文件组件来开发，更加利于编码，利于后期维护，一个文件包含了所有的内容

5. vscode的 `vue 2 snippets` 都升级一下

```
1 <template>
2   <!-- 模板 结构 -->
3 </template>
4
5 <script>
6   // 逻辑
7   export default {
8     // 组件的属性
9     methods: {
10
11     },
12     data() {}
13     // ...
14   }
15 </script>
16
17 <style>
18   /* 样式 */
19
20 </style>
21
```

## Vue-cli

### 基本概念

自动化工具的集合，也叫Vue的脚手架

1. 把.vue翻译成浏览器可以识别的内容
2. 自动刷新浏览器



3. 自动压缩代码
4. 自动的把js翻译为低版本的js
5. 作为代理服务器
6. ....

## 安装

[官网](#)

[安装](#)

1. 找到系统自带的命令行，以管理员身份运行以下命令
2. 更改npm源，使npm安装更快

```
1 | npm config set registry https://registry.npm.taobao.org
```

3. 确定npm源更改成功。显示 `https://registry.npm.taobao.org/` 表示成功

```
1 | npm config list
```

4. 安装vue-cli

```
1 | npm install -g @vue/cli
```

5. 确定vue-cli是否安装成功. 显示版本号表示成功，保证版本号在3.0以上

```
1 | vue --version
```

## 注意点

1. 所有同学都执行一下安装vue-cli，已经安装过，再执行就会更新。由于新旧版本有差异，最好都用最新版本
2. 其他命令行可能稍有差异，最好用系统自带
3. 以管理员身份运行，可能需要权限
4. 清除npm缓存之后，重新安装



1. `npm cache clean -f`
2. 重新执行安装的命令
5. 再不行的话，开手机4G网安装或者用讲师的网络

# Vue-cli 项目创建

[传送门](#)

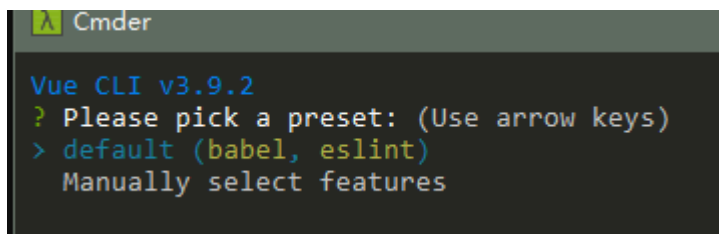
## 创建的流程

### 1. 合适的目录下，创建项目

1. 执行后会创建一个项目文件夹，所以执行命令时的路径要选择好
2. 项目名不要有中文，不要有大写字母，尽可能有意义

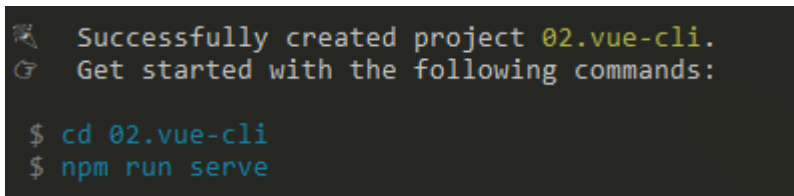
```
1 | vue create 项目名
```

### 2. 弹出的对话框先选择默认的选项



```
Cmder
Vue CLI v3.9.2
? Please pick a preset: (Use arrow keys)
> default (babel, eslint)
   Manually select features
```

### 4. 稍等一会，等进度条走完 提示如下画面说明成功了



```
Successfully created project 02.vue-cli.
Get started with the following commands:

$ cd 02.vue-cli
$ npm run serve
```

### 5. 进入项目文件夹

1. `cd 项目名` 直接根据提示即可

### 6. 运行项目

1. `npm run serve`

### 7. 稍等片刻，出现如下效果说明成功了

```
DONE Compiled successfully in 4556ms

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.17.58:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

## 报错的原因

1.

```
Unknown command creat.
Did you mean create?
```

创建的命令输入错误 create 输入成了 creat

2.

```
npm ERR! path C:\Users\Administrator\
npm ERR! code EPERM
npm ERR! errno -4048
npm ERR! syscall unlink
npm ERR! Error: EPERM: operation not
allowed, unlink 'C:\Users\Administrator\
node_modules\postcss-840ea3e5\lib'
```

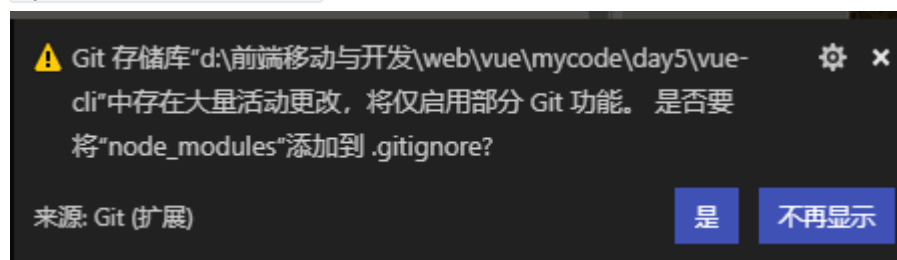
2. 终端的权限问题；新建管理员模式的终端

3. 当前这个文件夹，这个文件被其他软件占用：关闭所有可能影响的软件（重启）

4. npm包管理工具的问题:

2. 执行 `npm cache clean -f` 在重新创建项目

3.



创建项目是，又到了第三方模块，文件太多了git人为没有必要管，提示你一下

vue-cli创建项目是，已经设置了git忽略文件 就在 `.gitignore` 中

## 总结