

1: 路由的使用

1: cnpm i vue-router -S

2:

NPM

```
npm install vue-router
```

如果在一个模块化工程中使用它，必须要通过 `Vue.use()` 明确地安装路由功能：

```
import Vue from 'vue'
import VueRouter from 'vue-router'

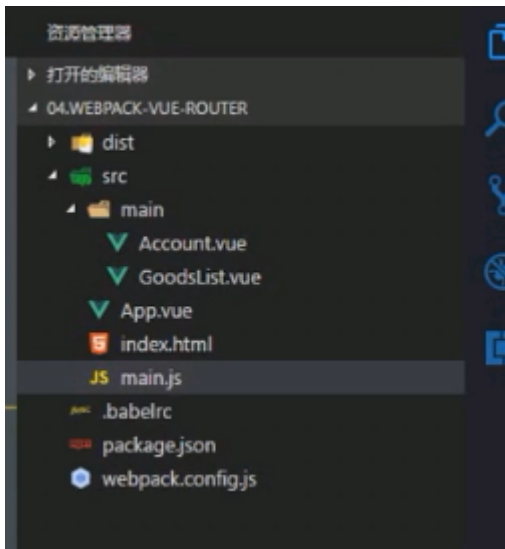
Vue.use(VueRouter)
```

如果使用全局的 `script` 标签，则无须如此（手动安装）。

3:

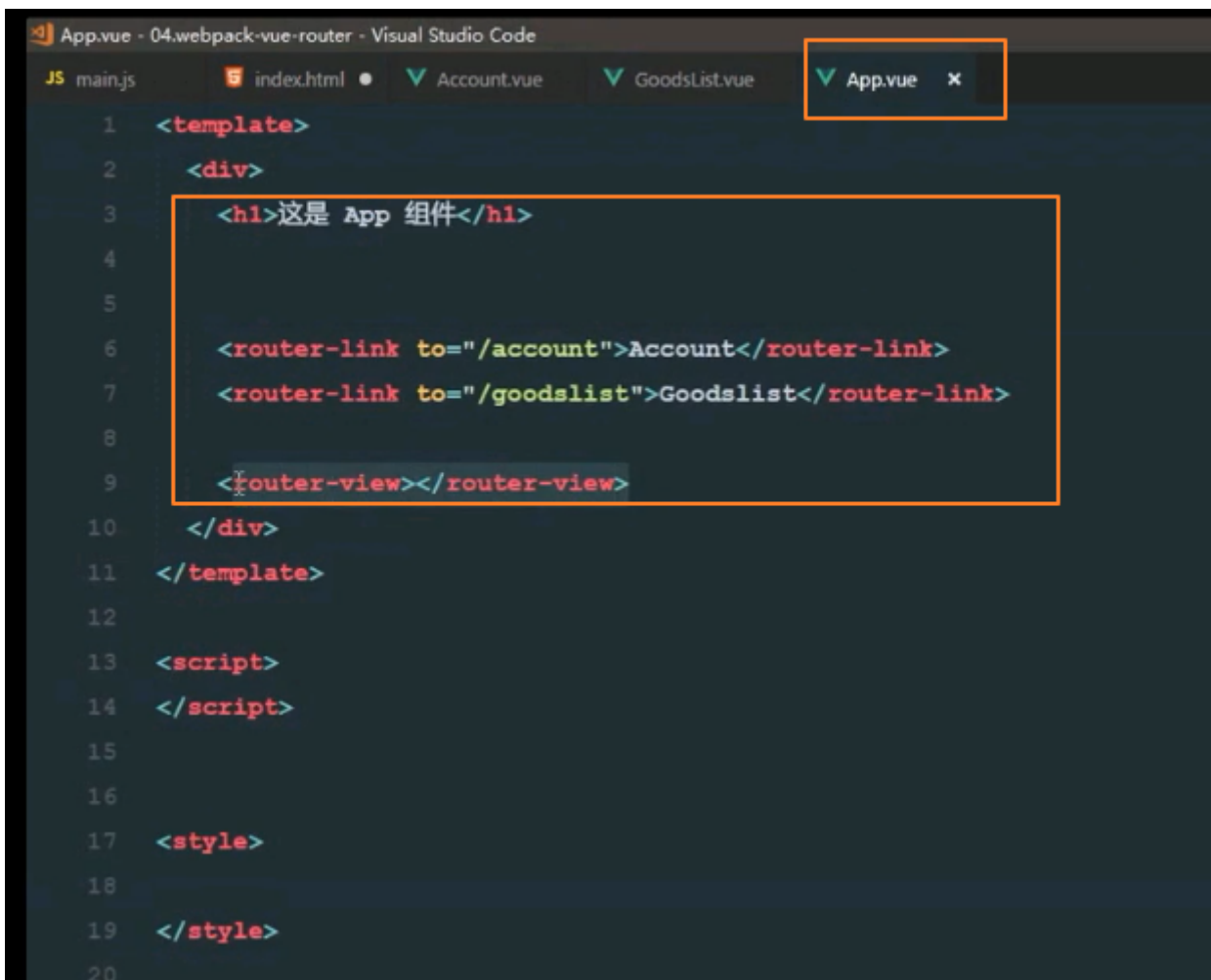
```
main.js - 04.webpack-vue-router - Visual Studio Code
JS main.js Account.vue GoodsList.vue App.vue 用户名:子乐大师

1  import Vue from 'vue'
2  // 1. 导入 vue-router 包
3  import VueRouter from 'vue-router'
4  // 2. 手动安装 VueRouter
5  Vue.use(VueRouter)
6
7  // 导入 app 组件
8  import app from './App.vue'
9  // 导入 Account 组件
10 import account from './main/Account.vue'
11 import goodslist from './main/GoodsList.vue'
12
13 // 3. 创建路由对象
14 var router = new VueRouter({
15   routes: [          4:挂在路由
16     // account goodslist
17     { path: '/account', component: account },
18     { path: '/goodslist', component: goodslist }
19   ]
20 })
21
```



需要注意的是：

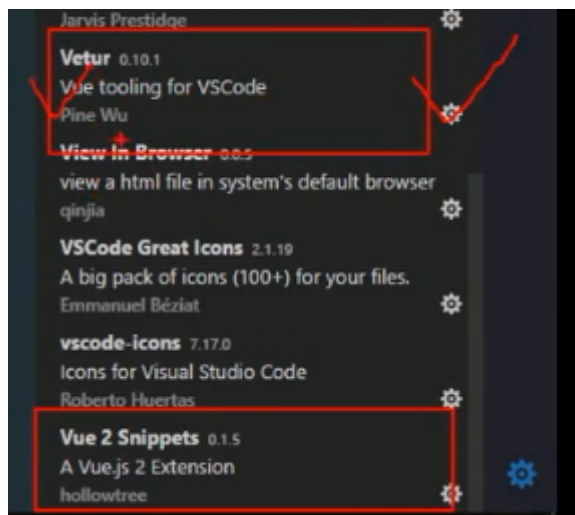
App 这个组件，是通过 VM 实例的 render 函数，渲染出来的，render 函数如果要渲染组件，渲染出来的组件，只能放到 el: '#app' 所指定的元素中；Account 和 GoodsList 组件，是通过路由匹配监听到的，所以，这两个组件，只能展示到属于路由的 `<router-view></router-view>` 中去；



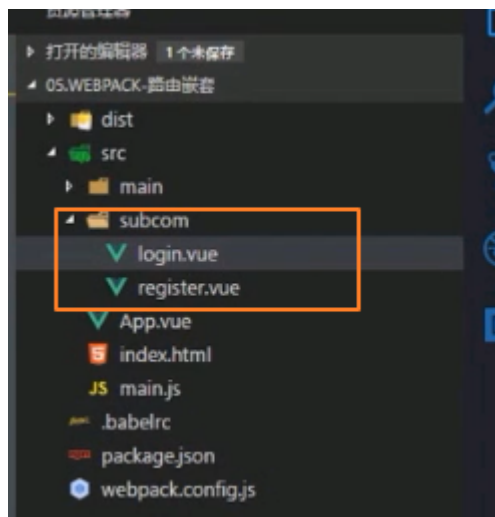
```
var vm = new Vue({
  el: '#app',
  render: c => c(app),
  // render 会把 el 指定的容器中，所有的内容都清空覆盖，所以 不要 把 路由的 router-view
  // 和 router-link 直接写到 el 所控制的元素中
  router // 4. 将路由对象挂载到 vm 上
})
```

2: 路由嵌套

1: 两个辅助的插件



2: 目录结构



```
main.js - 05.webpack-路由嵌套 - Visual Studio Code
JS main.js x
9 // 导入 Account 组件
10 import account from './main/Account.vue'
11 import goodslist from './main/GoodsList.vue'
12
13 // 导入Account的两个子组件
14 import login from './subcom/login.vue'
15 import register from './subcom/register.vue'
16
17 // 3. 创建路由对象
18 var router = new VueRouter({ ...
31 })
32
33 var vm = new Vue({
34   el: '#app',
35   render: c => c(app), // render 会把 el 指定的容器中，所有的内容都清空覆盖，所以 不要 把
   路由的 router-view 和 router-link 直接写到 el 所控制的元素中
36   router // 4. 将路由对象挂载到 vm 上
37 })
38
39 // 注意：App 这个组件，是通过 VM 实例的 render 函数，渲染出来的，render 函数如果要渲染 组
   件，渲染出来的组件，只能放到 el: '#app' 所指定的 元素中
```

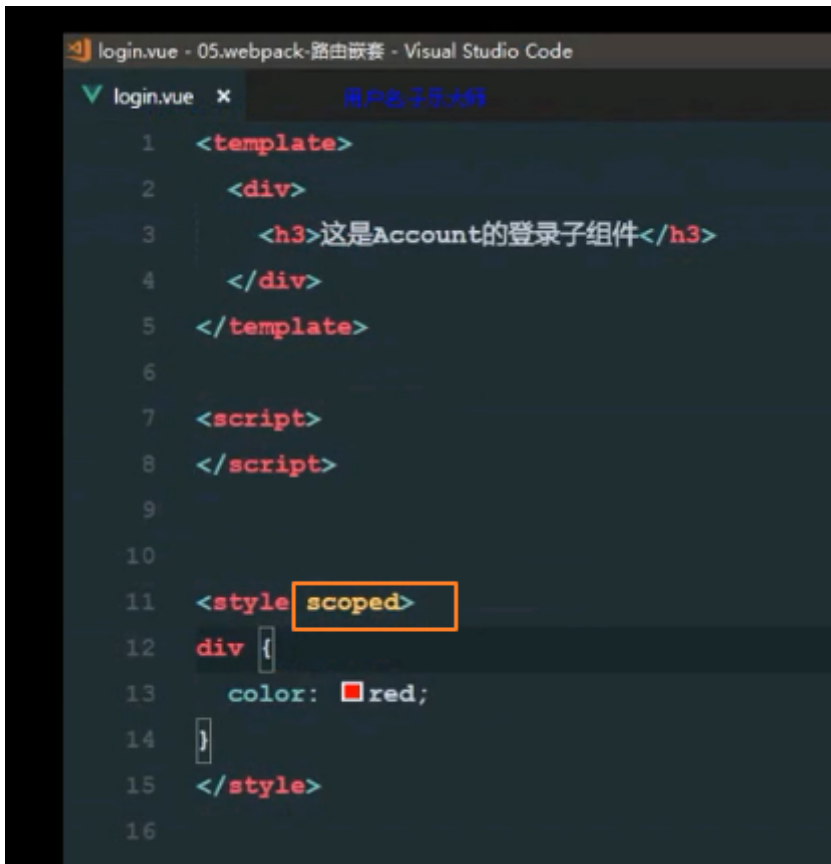
```
Account.vue - 05.webpack-路由嵌套 - Visual Studio Code 用户名:子东次郎
JS main.js Account.vue x
1 <template>
2   <div>
3     <h1>这是 Account 组件</h1>
4
5     <router-link to="/account/login">登录</router-link>
6     <router-link to="/account/register">注册</router-link>
7   </div>
8 </template>
9
10
11 <script>
12 </script>
13
14 <style>
15
16 </style>
```

子组件样式：关键字 `scoped` 这样样式就只会给自己子组件加了，不然就会给所有的选择的标签去加

还有`lang` 属性：

普通的 `style` 标签只支持 普通的 样式，如果想要启用 `scss` 或 `less` ，需要为 `style` 元素，设置 `lang` 属性

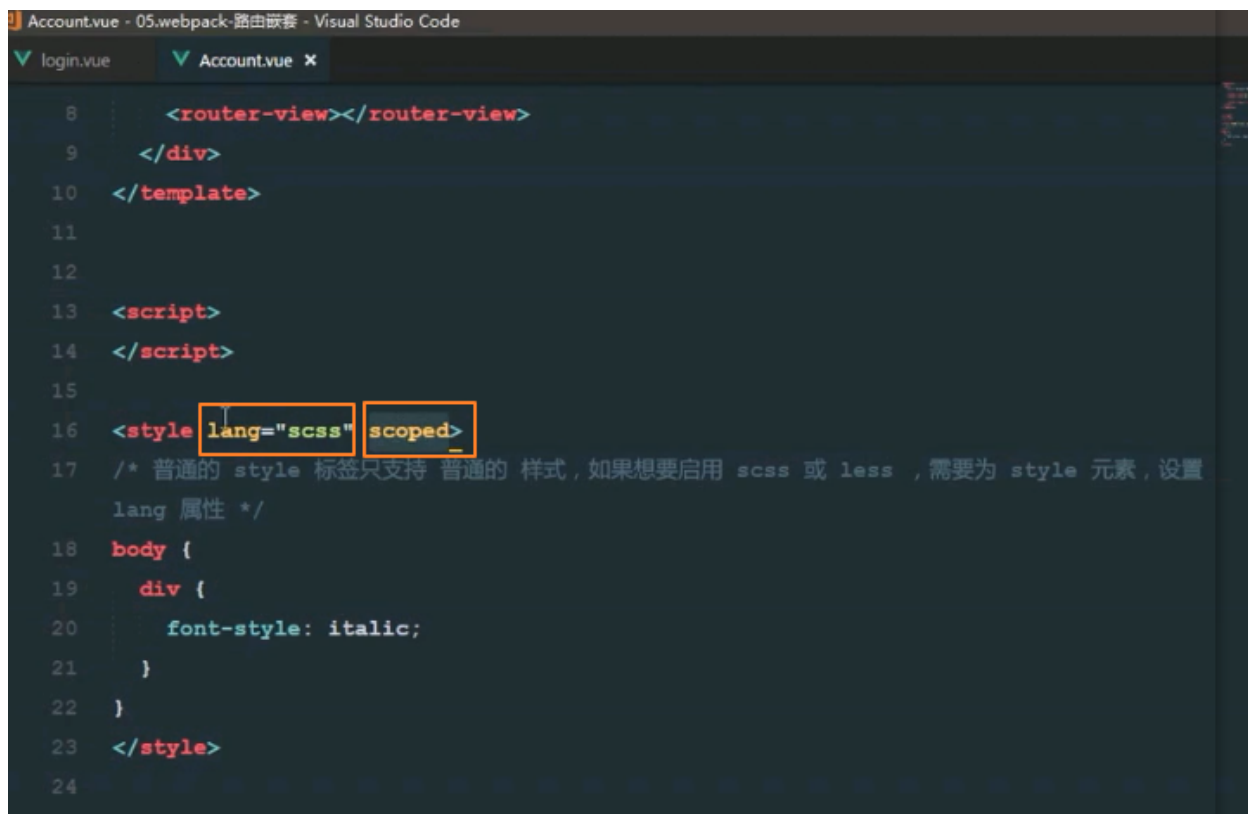
只要咱们的 style 标签，是在 .vue 组件中定义的，那么，推荐都为 style 开启 scoped 属性



The screenshot shows the 'login.vue' file in Visual Studio Code. The file has a tab labeled 'login.vue' and a title '用户名登录大略'. The code is as follows:

```
1 <template>
2   <div>
3     <h3>这是Account的登录子组件</h3>
4   </div>
5 </template>
6
7 <script>
8 </script>
9
10
11 <style scoped>
12   div {
13     color: red;
14   }
15 </style>
16
```

The `<style scoped>` tag on line 11 is highlighted with an orange box.



The screenshot shows the 'Account.vue' file in Visual Studio Code. The file has a tab labeled 'Account.vue' and a title '用户名登录大略'. The code is as follows:

```
8   <router-view></router-view>
9 </div>
10 </template>
11
12
13 <script>
14 </script>
15
16 <style lang="scss" scoped>
17   /* 普通的 style 标签只支持 普通的 样式，如果想要启用 scss 或 less ，需要为 style 元素，设置
18      lang 属性 */
19   body {
20     div {
21       font-style: italic;
22     }
23   }
24 </style>
```

The `lang="scss"` and `scoped` attributes on line 16 are highlighted with orange boxes.

3：抽离路由模块

把导入需要依赖的模块

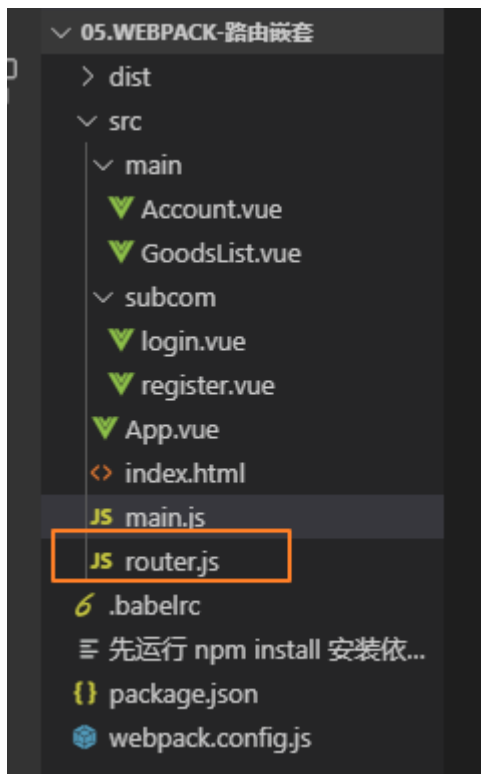
和 创建的路由对象模块抽离出去就可以



```
15 main.js
9 // 导入 Account 组件
10 import account from './main/Account.vue'
11 import goodslist from './main/GoodsList.vue'
12
13 // 导入Account的两个子组件
14 import login from './subcom/login.vue'
15 import register from './subcom/register.vue'
16
17 // 3. 创建路由对象
18 var router = new VueRouter({
19   routes: [
20     // account goodslist
21     {
22       path: '/account',
23       component: account,
24       children: [
25         { path: 'login', component: login },
26         { path: 'register', component: register }
27       ]
28     },
29     { path: '/goodslist', component: goodslist }
30   ]
}
```

把如上的代码抽离出来，然后

当前的项目结构：



在main.js文件中

```
> JS main.js > ...
1  import Vue from 'vue'
2  // 1. 导入 vue-router 包
3  import VueRouter from 'vue-router'
4  // 2. 手动安装 VueRouter
5  Vue.use(VueRouter)
6
7  // 导入 app 组件
8  import app from './App.vue'
9
10 // 导入 自定义路由模块
11 import router from './router.js'
12
13 var vm = new Vue({
14   el: '#app',
15   render: c => c(app), // render 会挂
16   router // 4. 将路由对象挂载到 vm 上
17 })
18
19 // 注意: App 这个组件, 是通过 vm 实例
20 // Account 和 GoodsList 组件, 是通过
```

在router.js文件中

login.vue Account.vue register.vue App.vue

```
src > JS router.js > ...
1  import VueRouter from 'vue-router'
2
3  // 导入 Account 组件
4  import account from './main/Account.vue'
5  import goodslist from './main/GoodsList.vue'
6
7  // 导入Account的两个子组件
8  import login from './subcom/login.vue'
9  import register from './subcom/register.vue'
10
11 // 3. 创建路由对象
12 var router = new VueRouter({
13   routes: [
14     // account goodslist
15     {
16       path: '/account',
17       component: account,
18       children: [
19         { path: 'login', component: login },
20         { path: 'register', component: register }
21       ]
22     },
23     { path: '/goodslist', component: goodslist }
24   ]
25 })
26
27 // 把路由对象暴露出去
28 export default router
```