

# Cook-Manage

---

项目准备

制作导航

配置路由跳转

路由小细节

路由name属性和跳转

二级三级路由

默认展示

导航守卫

复用router-view

控制滚动行为

登录和注册

axios实现数据存储

menu页面

设计购物车

展示购物车数据

管理页面

请求数据

删除请求

请求数据get

Vuex-mutations更改数据状态

管理页面同步到vuex

登录成功显示用户信息

对store文件的代码进行抽离1

使用Module实现模块化

vue2

1. 命令行安装

全局安装 vue-cli

```
npm install --global vue-cli
```

创建一个基于webpack 模板的新项目

```
vue init webpack myproject
```

安装依赖

```
cd my-project
```

```
npm install
```

```
npm run dev
```

## 项目准备

### 1. 创建简单模板

```
vue init webpack-simple pizza-app
```

```
cd pizza-app
```

```
npm run dev
```

### 2. 创建components组件文件夹

- 创建Header.vue组件
- 创建Home.vue组件
- 创建Menu.vue组件
- 创建Admin.vue组件
- 创建Login.vue组件
- 创建Register.vue组件
- 创建about文件夹
  - 创建About.vue组件

## 制作导航

使用bootstrap4

### 1. 在index.js中引入bootstrap4的链接

<https://v4.bootcss.com/docs/getting-started/download/>

### 2. header.vue基础内容

```
1 <template>
2   <nav class="navbar navbar-expand-lg navbar-light bg-light">
3     <a class="py-2" href="#" aria-label="Product">
4       <svg
5         xmlns="http://www.w3.org/2000/svg"
6         width="24"
7         height="24"
8         fill="none"
9         stroke="currentColor"
10        stroke-linecap="round"
11        stroke-linejoin="round"
12        stroke-width="2"
13        class="d-block mx-auto"
14        role="img"
15        viewBox="0 0 24 24"
16        focusable="false"
17      >
18       <title>Product</title>
19       <circle cx="12" cy="12" r="10" />
20       <path
21         d="M14.31 8l5.74 9.94M9.69 8h11.48M7.38 12l5.74-9.94M9.69
22         16L3.95 6.06M14.31 16H2.83m13.79-4l-5.74 9.94"
23       />
24     </svg>
25   </a>
26   <a href="/" class="navbar-brand">Pizza点餐系统</a>
27   <ul class="navbar-nav">
28     <li><a href="#" class="nav-link">主页</a></li>
29     <li><a href="#" class="nav-link">菜单</a></li>
30     <li><a href="#" class="nav-link">管理</a></li>
31     <li><a href="#" class="nav-link">关于我们</a></li>
32   <ul class="navbar-nav ml-auto">
33     <li><a href="#" class="nav-link">登录</a></li>
34     <li><a href="#" class="nav-link">注册</a></li>
35   </ul>
36 </nav>
37 </template>
```

### 3. 将header.vue的内容展示到页面上

```
<template>
  <div id="app">
    <div class="container">
      <!--3. 使用组件 -->
      <app-header></app-header>
    </div>
  </div>
</template>

<script>
// 1. 引入组件
import Header from "./components/Header.vue";
export default {
  name: "app",
  // 2. 注册组件
  components: {
    // header 和 h5 标签有点重名冲突，改名
    appHeader: Header,
  },
};
</script>

<style>
</style>
```

## 配置路由跳转

1. `npm install vue-router --save`
2. 在main.js配置路由

```

import Vue from 'vue'
import App from './App.vue'
// 1. 引入路由模块
import VueRouter from 'vue-router'
// 3. 引入各个组件
import Home from './components/Home'
import Menu from './components/Menu'

// 2. 使用路由
Vue.use(VueRouter)

const routes = [
  { path: '/', component: Home },
  { path: '/menu', component: Menu }
]

// 4. 实例化
const router = new VueRouter({
  routes,
  mode: 'history'
})

new Vue({
  el: '#app',
  // 5. 使用路由
  router,
  render: h => h(App)
})

```

### 3. app.vue中展示路由模块

- 将a标签替换为router-link href属性替换为to

使用a标签会刷新，但是router-link不会

```
</a>
<a href="/" class="navbar-brand">Pizza点餐系统</a>
<ul class="navbar-nav">
  <li><router-link to="/" class="nav-link">主页</router-link></li>
  <li><router-link to="/menu" class="nav-link">菜单</router-link></li>
  <li><router-link to="#" class="nav-link">管理</router-link></li>
  <li><router-link to="#" class="nav-link">关于我们</router-link></li>
</ul>
<ul class="navbar-nav ml-auto">
  <li><router-link to="#" class="nav-link">登录</router-link></li>
  <li><router-link to="#" class="nav-link">注册</router-link></li>
</ul>
</nav>
```

- app.vue中需要展示路由页面的地方写上router-view

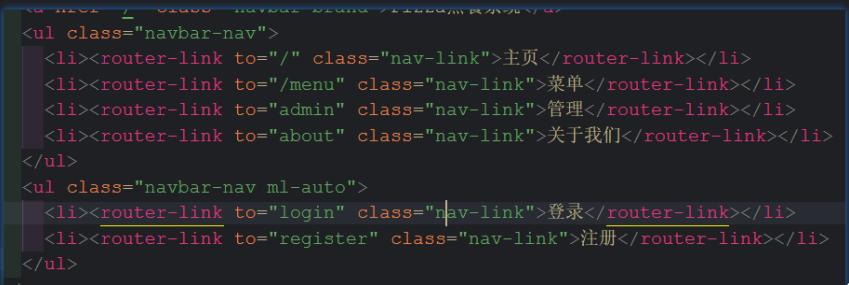
```
index.html      ▼ Header.vue      ▼ App.vue      ×      ▼ Register.vue      ▼ Home.vue
c > ▼ App.vue > {} "App.vue" > ⚡ template > ⚡ div#app > ⚡ div.container > ⚡ router-view
1   <template>
2     <div id="app">
3       <div class="container">
4         <!-- 3. 使用组件 -->
5         <app-header></app-header>
6       </div>
7       <div class="container">
8         <!-- 展示路由页面-->
9         <router-view></router-view>
10        </div>
11      </div>
12    </template>
```

- 配置好所有路由

```
import Menu from './components/Menu'
import Login from './components/Login'
import Admin from './components/Admin'
import About from './components/about/About'
import Register from './components/Register'

// 2. 使用路由
Vue.use(VueRouter)

const routes = [
  { path: '/', component: Home },
  { path: '/menu', component: Menu },
  { path: '/Login', component: Login },
  { path: '/About', component: About },
  { path: '/Register', component: Register },
  { path: '/Admin', component: Admin }
]
```



## 路由小细节

1. 可以给router-link指定它能成为什么标签（默认为a标签）

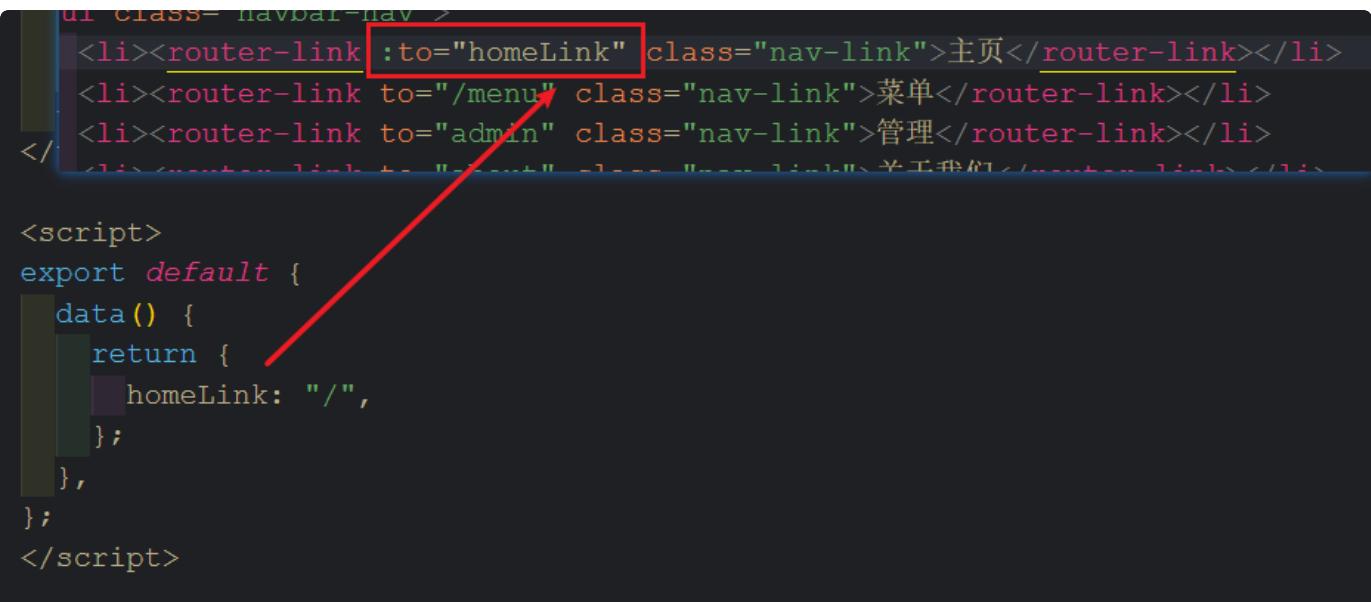
加上 tag="标签名"

2. to中可以绑定动态链接

在data中设置路径

```
<ul class="navbar-nav">
  <li><router-link :to="homeLink" class="nav-link">主页</router-link></li>
  <li><router-link to="/menu" class="nav-link">菜单</router-link></li>
  <li><router-link to="admin" class="nav-link">管理</router-link></li>
  <li><router-link to="about" class="nav-link">关于我们</router-link></li>
</ul>

<ul class="navbar-nav ml-auto">
  <li><router-link to="login" class="nav-link">登录</router-link></li>
  <li><router-link to="register" class="nav-link">注册</router-link></li>
</ul>
```



```
<script>
export default {
  data() {
    return {
      homeLink: '/',
    };
  },
};
</script>
```

3. 路由输入错误进入主页

```
// 2. 使用路由
Vue.use(VueRouter)

const routes = [
  { path: '/', component: Home },
  { path: '/menu', component: Menu },
  { path: '/Login', component: Login },
  { path: '/About', component: About },
  { path: '/Register', component: Register },
  { path: '/Admin', component: Admin },
  { path: '*', redirect: '/' }]
```

redirect可以自己设置要  
跳转的路由

## 路由name属性和跳转

```
const routes = [
  { path: '/', name:'homeLink', component: Home },
  { path: '/menu', component: Menu },
  { path: '/Login', component: Login },
  { path: '/About', component: About },
  { path: '/Register', component: Register },
  { path: '/Admin', component: Admin },
  { path: '*', redirect: '/' }

<ul class="navbar-nav">
  <li><router-link :to="{name:'homeLink'}" class="nav-link">主页</router-link></li>
  <li><router-link to="/menu" class="nav-link">菜单</router-link></li>
  <li><router-link to="admin" class="nav-link">管理</router-link></li>
  <li><router-link to="about" class="nav-link">关于我们</router-link></li>
</ul>
```

1. 跳转到上一个浏览的页面

```
<template>
<div>
  <h1>Home</h1>
  <button @click="goToMenu" class="btn btn-success">跳转</button>
</div>
</template>

<script>
export default {
  methods: {
    goToMenu() {
      // 跳转到上一次浏览的页面
      this.$router.go(-1);
    },
  },
};
</script>
```

## 2. 指定跳转的地址

```
export default {
  methods: {
    goToMenu() {
      // 跳转到上一次浏览的页面
      // this.$router.go(-1);
      this.$router.replace(["/menu"]);
    },
  },
};
```

## 3. 跳转到指定路由名字下

```
export default {
  methods: {
    goToMenu() {
      // 跳转到上一次浏览的页面
      // this.$router.go(-1);
      // this.$router.replace("/menu");
      this.$router.replace({ name: "menuLink" });
    },
  },
};
```

#### 4. 常用压栈

```
export default {
  methods: {
    goToMenu() [
      // 跳转到上一次浏览的页面
      // this.$router.go(-1);
      // this.$router.replace("/menu");
      // this.$router.replace({ name: "menuLink" });
      // this.$router.push("/menu");
      this.$router.push({ name: "menuLink" });
    ],
  },
};
```

## 二级三级路由

### 1. 在about文件夹中创建:

- Contact.vue
- Delivery.vue
- History.vue
- OrderingGuide.vue

### 2. about.vue中设置结构,

```
<template>
<div>
  <div class="row mb-5">
    <div class="col-4">
      <!-- 导航 -->
      <div class="list-group mb-5">
        <router-link tag="li" class="nav-link" :to="{ name: 'historyLink' }">
          <a class="list-group-item list-group-item-action">历史订单</a>
        </router-link>
        <router-link tag="li" class="nav-link" :to="{ name: 'contactLink' }">
          <a class="list-group-item list-group-item-action">联系我们</a>
        </router-link>
        <router-link
          tag="li"
          class="nav-link"
          :to="{ name: 'orderingGuideLink' }"
        >
          <a class="list-group-item list-group-item-action">点餐菜单</a>
        </router-link>
        <router-link tag="li" class="nav-link" :to="{ name: 'deliveryLink' }">
          <a class="list-group-item list-group-item-action">快递信息</a>
        </router-link>
      </div>
    </div>
    <div class="col-8">
      <!-- 导航所对应的内容 -->
      <router-view></router-view>
    </div>
  </div>
</div>
</template>
```

```
1 <template>
2   <div>
3     <div class="row mb-5">
4       <div class="col-4">
5         <!-- 导航 -->
6         <div class="list-group mb-5">
7           <router-link tag="li" class="nav-link" :to="{ name: 'historyLink' }">
8             <a class="list-group-item list-group-item-action">历史订单</a>
9           </router-link>
10          <router-link tag="li" class="nav-link" :to="{ name: 'contactLink' }">
11            <a class="list-group-item list-group-item-action">联系我们</a>
12          </router-link>
13          <router-link
14            tag="li"
15            class="nav-link"
16            :to="{ name: 'orderingGuideLink' }"
17          >
18            <a class="list-group-item list-group-item-action">点餐菜单</a>
19          </router-link>
20          <router-link tag="li" class="nav-link" :to="{ name: 'deliveryLink' }">
21            <a class="list-group-item list-group-item-action">快递信息</a>
22          </router-link>
23        </div>
24      </div>
25      <div class="col-8">
26        <!-- 导航所对应的内容 -->
27        <router-view></router-view>
28      </div>
29    </div>
30  </div>
31 </template>
32
```

### 3. 在main.js中配置二级路由

```

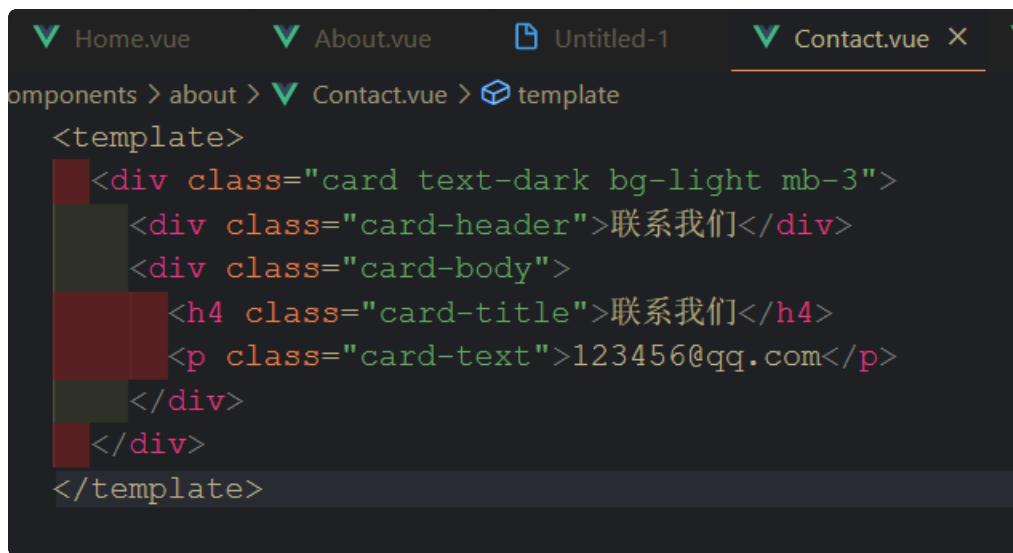
// 二级路由
import Contact from './components/about/Contact'
import Delivery from './components/about/Delivery'
import History from './components/about/History'
import OrderingGuide from './components/about/OrderingGuide'

// 2. 使用路由
Vue.use(VueRouter)

const routes = [
  { path: '/', name: 'homeLink', component: Home },
  { path: '/menu', name: 'menuLink', component: Menu },
  { path: '/Login', name: 'loginLink', component: Login },
  {
    path: '/About', name: 'aboutLink', component: About, children: [
      { path: '/about/contact', name: 'contactLink', component: Contact },
      { path: '/about/History', name: 'historyLink', component: History },
      { path: '/about/delivery', name: 'deliveryLink', component: Delivery },
      { path: '/about/orderingGuide', name: 'orderingGuideLink', component: OrderingGuide }
    ]
  },
  { path: '/Register', name: 'registerLink', component: Register },
  { path: '/Admin', name: 'adminLink', component: Admin },
  { path: '*', redirect: '/' }
]

```

#### 4. 二级路由内容



```

<template>
  <div class="card text-dark bg-light mb-3">
    <div class="card-header">联系我们</div>
    <div class="card-body">
      <h4 class="card-title">联系我们</h4>
      <p class="card-text">123456@qq.com</p>
    </div>
  </div>
</template>

```

#### 5. 在about中新建Contact文件夹

- 新建Phone.vue
- 新建PersonName.vue

三级路由页面内容

```
components > about > Contact.vue > template > div.card.text-dark.bg-light.mb-3 > div.card-body > route
<template>
  <div class="card text-dark bg-light mb-3">
    <div class="card-header">联系我们</div>
    <div class="card-body">
      <h4 class="card-title">联系我们</h4>
      <p class="card-text">123456@qq.com</p>

      <router-link :to="{ name: 'phoneNumber' }">电话</router-link>
      <router-link :to="{ name: 'personName' }">联系人</router-link>

      <router-view></router-view>
    </div>
  </div>
</template>
```

引入路由：

```
// 三级路由
import Phone from './components/about/contact/Phone'
import PersonName from './components/about/contact/PersonName'

const routes = [
  { path: '/', name: 'homeLink', component: Home },
  { path: '/menu', name: 'menuLink', component: Menu },
  { path: '/Login', name: 'loginLink', component: Login },
  {
    path: '/About', name: 'aboutLink', component: About, children: [
      {
        path: '/about/contact', name: 'contactLink', component: Contact,
        // 三级路由
        children: [
          { path: '/phone', name: "phoneNumber", component: Phone },
          { path: '/personName', name: "personName", component: PersonName },
        ]
      },
      { path: '/about/History', name: 'historyLink', component: History },
      { path: '/about/delivery', name: 'deliveryLink', component: Delivery },
      { path: '/about/orderingGuide', name: 'orderingGuideLink', component: OrderingGuide },
    ]
  }
]
```

# 默认展示

```
const routes = [
  { path: '/', name: 'homeLink', component: Home },
  { path: '/menu', name: 'menuLink', component: Menu },
  { path: '/Login', name: 'loginLink', component: Login },
  {
    path: '/About', name: 'aboutLink', redirect: '/about/contact', component: About, children: [
      {
        path: '/about/contact', name: 'contactLink', redirect: '/personName', component: Contact,
        // 三级路由
        children: [
          { path: '/phone', name: "phoneNumber", component: Phone },
          { path: '/personName', name: "personName", component: PersonName },
        ]
      ],
      { path: '/about/History', name: 'historyLink', component: History },
      { path: '/about/delivery', name: 'deliveryLink', component: Delivery },
      { path: '/about/orderingGuide', name: 'orderingGuideLink', component: OrderingGuide },
    ]
  },
  { path: '/Register', name: 'registerLink', component: Register },
  { path: '/Admin', name: 'adminLink', component: Admin },
  { path: '*', redirect: '/' }
]
```

# 导航守卫

未登录就无法点击别的导航

## 1. 全局守卫

```
// 全局守卫 to 进入 from 来自 next 是否展示
router.beforeEach((to, from, next) => {
  // 判断 store.getters.isLogin === false
  if (to.path == '/login' || to.path == '/register') {
    next();
  } else {
    alert('还没有登录，请登录');
    next('/login');
  }
})
```

## 2. 组件守卫

```
// 后置钩子
router.afterEach((to, from) => {
  alert(['after each']);
})
```

### 3. 路由独享守卫

与全局守卫作用范围不同

```
],
},
{ path: '/about/History', name: 'historyLink', component: History },
{ path: '/about/delivery', name: 'deliveryLink', component: Delivery },
{ path: '/about/orderingGuide', name: 'orderingGuideLink', component: OrderingGuide },

],
},
{ path: '/Register', name: 'registerLink', component: Register },
{
  path: '/Admin', name: 'adminLink', component: Admin, beforeEnter: (to, from, next) => [
    alert('非登录状态不能访问此页面'),
    next(false),
  ]
},
{ path: '*', redirect: '/' }
```

### 4. 组件内守卫

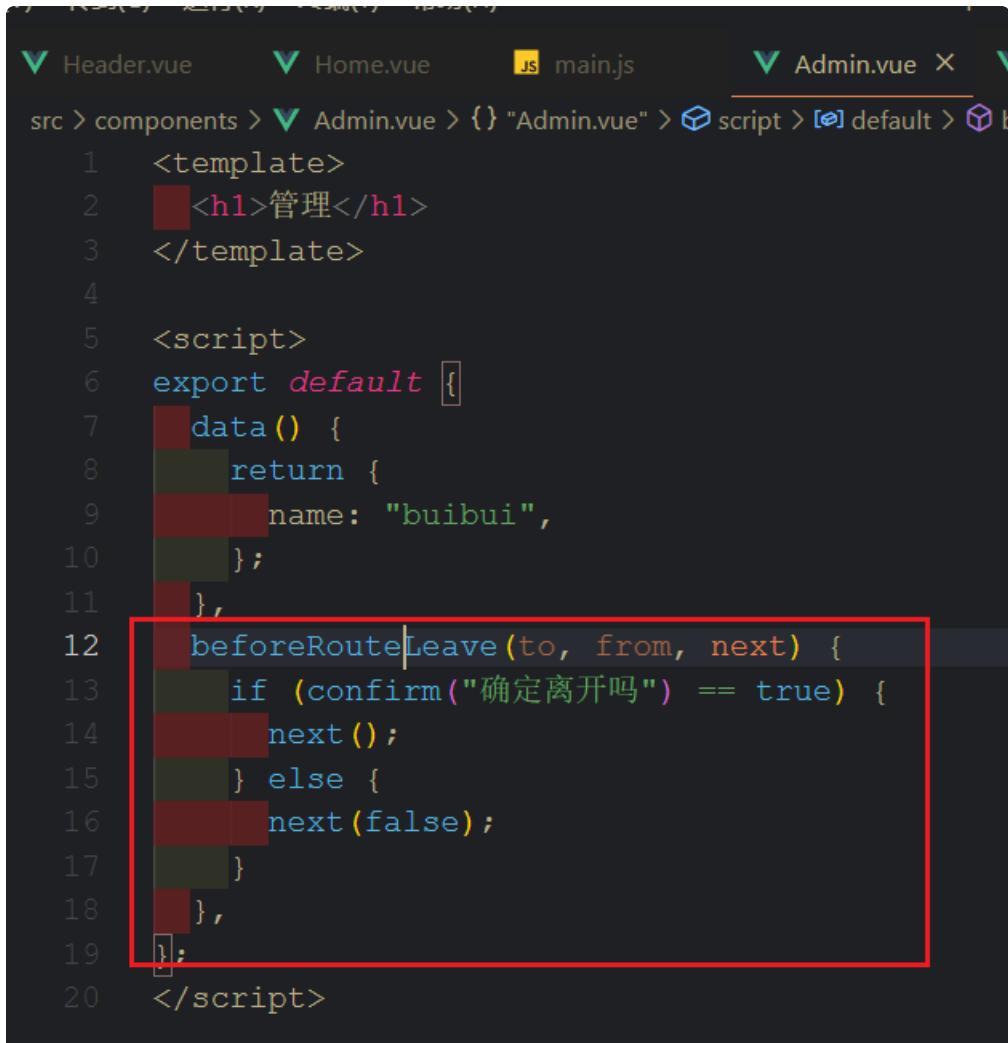
进入组件:

V Header.vue V Home.vue JS main.js V Admin.vue X V P

src > components > V Admin.vue > {} "Admin.vue" > script > default > beforeRouteEnter

```
1  <template>
2    <h1>管理</h1>
3  </template>
4
5  <script>
6  export default {
7    data () {
8      return {
9        name: "buibui",
10       };
11     },
12     beforeRouteEnter: (to, from, next) => [
13       // 此时拿不到数据
14       // alert("hello" + this.name);
15       // 需要在next中使用回调、
16       next ( (vm) => {
17         alert ("hello" + vm.name);
18       });
19     ],
20   };
21 </script>
```

离开组件:



```
src > components > Admin.vue > {} "Admin.vue" > script > default > buibui
1  <template>
2    <h1>管理</h1>
3  </template>
4
5  <script>
6    export default {
7      data() {
8        return {
9          name: "buibui",
10         };
11       },
12      beforeRouteLeave(to, from, next) {
13        if (confirm("确定离开吗") == true) {
14          next();
15        } else {
16          next(false);
17        }
18      },
19    };
20  </script>
```

## 复用router-view

1. 抽离main.js中的路由配置

新建routes.js文件

```
der.vue      V Home.vue      JS main.js      X routes.js
s main.js > ...
import Vue from 'vue'
import App from './App.vue'
// 1. 引入路由模块
import VueRouter from 'vue-router'
import { routes } from './routes'

// 2. 使用路由
Vue.use(VueRouter)

// 3. 实例化
const router = new VueRouter({
  routes,
  mode: 'history'
})

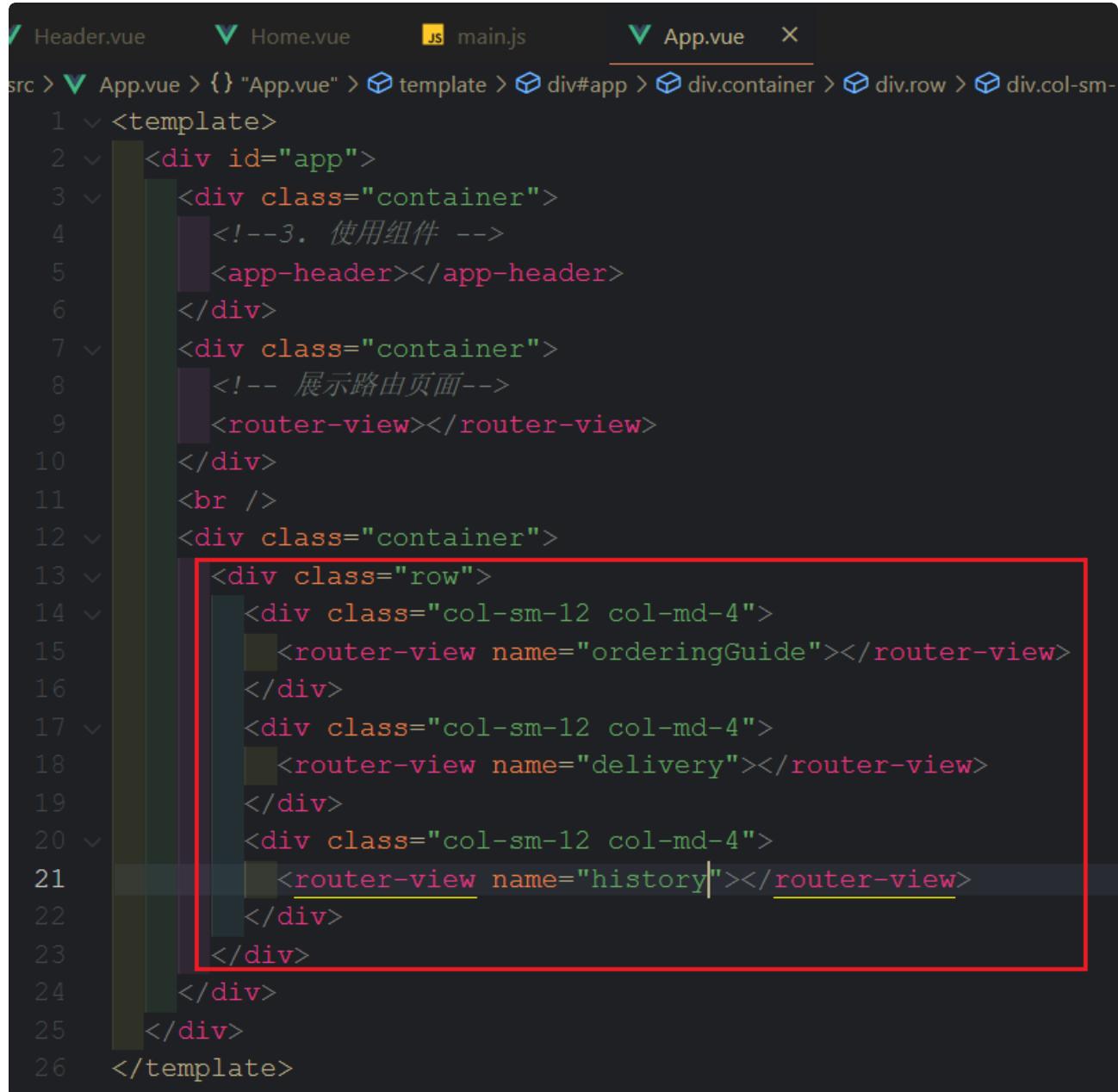
new Vue({
  el: '#app',
  // 4. 使用路由
  router,
  render: h => h(App)
})
```

```
9 export const routes = [
10   { path: '/', name: 'homeLink', component: Home },
11   { path: '/menu', name: 'menuLink', component: Menu },
12   { path: '/Login', name: 'loginLink', component: Login },
```

```
1 import Home from './components/Home'
2 import Menu from './components/Menu'
3 import Login from './components/Login'
4 import Admin from './components/Admin'
5 import About from './components/about/About'
6 import Register from './components/Register'
7
8 // 二级路由
9 import Contact from './components/about/Contact'
10 import Delivery from './components/about/Delivery'
11 import History from './components/about/History'
12 import OrderingGuide from './components/about/OrderingGuide'
13
14 // 三级路由
15 import Phone from './components/about/contact/Phone'
16 import PersonName from './components/about/contact/PersonName'
17
18
19 export const routes = [
20     { path: '/', name: 'homeLink', component: Home },
21     { path: '/menu', name: 'menuLink', component: Menu },
22     { path: '/Login', name: 'loginLink', component: Login },
23     {
24         path: '/About', name: 'aboutLink', redirect: '/about/contact',
25         component: About, children: [
26             {
27                 path: '/about/contact', name: 'contactLink', redirect:
28                     '/personName', component: Contact,
29                     // 三级路由
30                     children: [
31                         { path: '/phone', name: "phoneNumber", component:
32                             Phone },
33                         { path: '/personName', name: "personName", component:
34                             PersonName },
35                     ]
36             },
37             { path: '/about/History', name: 'historyLink', component:
38                 History },
39             { path: '/about/delivery', name: 'deliveryLink', component:
40                 Delivery },
41             { path: '/about/orderingGuide', name: 'orderingGuideLink',
42                 component: OrderingGuide },
43         ]
44     }
45 ]
```

```
39     },
40     { path: '/Register', name: 'registerLink', component: Register },
41     { path: '/Admin', name: 'adminLink', component: Admin },
42     { path: '*', redirect: '/' }
43 ]
```

## 2. 复用router-view



```
src > App.vue > {} "App.vue" > template > div#app > div.container > div.row > div.col-sm-12 col-md-4
1 <template>
2 <div id="app">
3   <div class="container">
4     <!--3. 使用组件-->
5     <app-header></app-header>
6   </div>
7   <div class="container">
8     <!-- 展示路由页面-->
9     <router-view></router-view>
10  </div>
11  <br />
12  <div class="container">
13    <div class="row">
14      <div class="col-sm-12 col-md-4">
15        <router-view name="orderingGuide"></router-view>
16      </div>
17      <div class="col-sm-12 col-md-4">
18        <router-view name="delivery"></router-view>
19      </div>
20      <div class="col-sm-12 col-md-4">
21        <router-view name="history"></router-view>
22      </div>
23    </div>
24  </div>
25 </div>
26 </template>
```

routes.js中：

```
export const routes = [
  {
    path: '/', name: 'homeLink', components: [
      default: Home,
      'orderingGuide': OrderingGuide,
      'delivery': Delivery,
      'history': History
    ]
  },
  { path: '/menu', name: 'menuLink', component: Menu },
  { path: '/Login', name: 'loginLink', component: Login },
]
```

## 控制滚动行为

```
// 4.实例化
const router = new VueRouter({
  routes,
  mode: 'history',
  scrollBehavior(to, from, savePosition) {
    return [ { x: 0, y: 100 } ]
  }
})
```

```
// 4.实例化
const router = new VueRouter({
  routes,
  mode: 'history',
  scrollBehavior(to, from, savePosition) {
    // return { x: 0, y: 100 }
    return [ { selector: '.btn' } ]
  }
})
```

```
// 4.实例化
const router = new VueRouter({
  routes,
  mode: 'history',
  scrollBehavior(to, from, savePosition) {
    // return { x: 0, y: 100 }
    // return { selector: '.btn' }
    if (savePosition) {
      return savePosition
    } else {
      return { x: 0, y: 0 }
    }
  }
})
```

## 登录和注册

注册

```
1 <template>
2   <div class="row mt-3">
3     <div class="col-md-12 col-lg-12">
4       <div class="card">
5         <div class="card-body">
6           
8           <form action="" @submit.prevent="onSubmit">
9             <div class="form-group">
10               <label for="email">邮箱</label>
11               <input type="email" class="form-control" v-model="email" />
12             </div>
13             <div class="form-group">
14               <label for="password">密码</label>
15               <input type="password" class="form-control" v-
16                 model="password" />
17             </div>
18             <div class="form-group">
19               <label for="confirm-password">确认密码</label>
20               <input
21                 type="password"
22                 class="form-control"
23                 v-model="confirmPassword"
24                 />
25             </div>
26             <button type="submit" class="btn btn-block btn-success">
27               注册
28             </button>
29           </form>
30         </div>
31       </div>
32     </template>
33
34 <script>
35 export default {
36   data() {
37     return {
38       email: '',
39       password: '',
40       confirmPassword: '',
41     };
42   },
43   methods: {
```

```
44     onSubmit() {},  
45   },  
46 };  
47 </script>
```

登录：

```
1 <template>
2   <div class="row mt-3">
3     <div class="col-md-12 col-lg-12">
4       <div class="card">
5         <div class="card-body">
6           
8           <form action="" @submit.prevent="onSubmit">
9             <div class="form-group">
10               <label for="email">邮箱</label>
11               <input type="email" class="form-control" v-model="email" />
12             </div>
13             <div class="form-group">
14               <label for="password">密码</label>
15               <input type="password" class="form-control" v-
model="password" />
16             </div>
17             <button type="submit" class="btn btn-block btn-success">
18               登录
19             </button>
20           </form>
21         </div>
22       </div>
23     </div>
24   </div>
25 </template>
26
27 <script>
28 export default {
29   data() {
30     return {
31       email: '',
32       password: '',
33     };
34   },
35   methods: {
36     onSubmit() {},
37   },
38 };
39 </script>
```

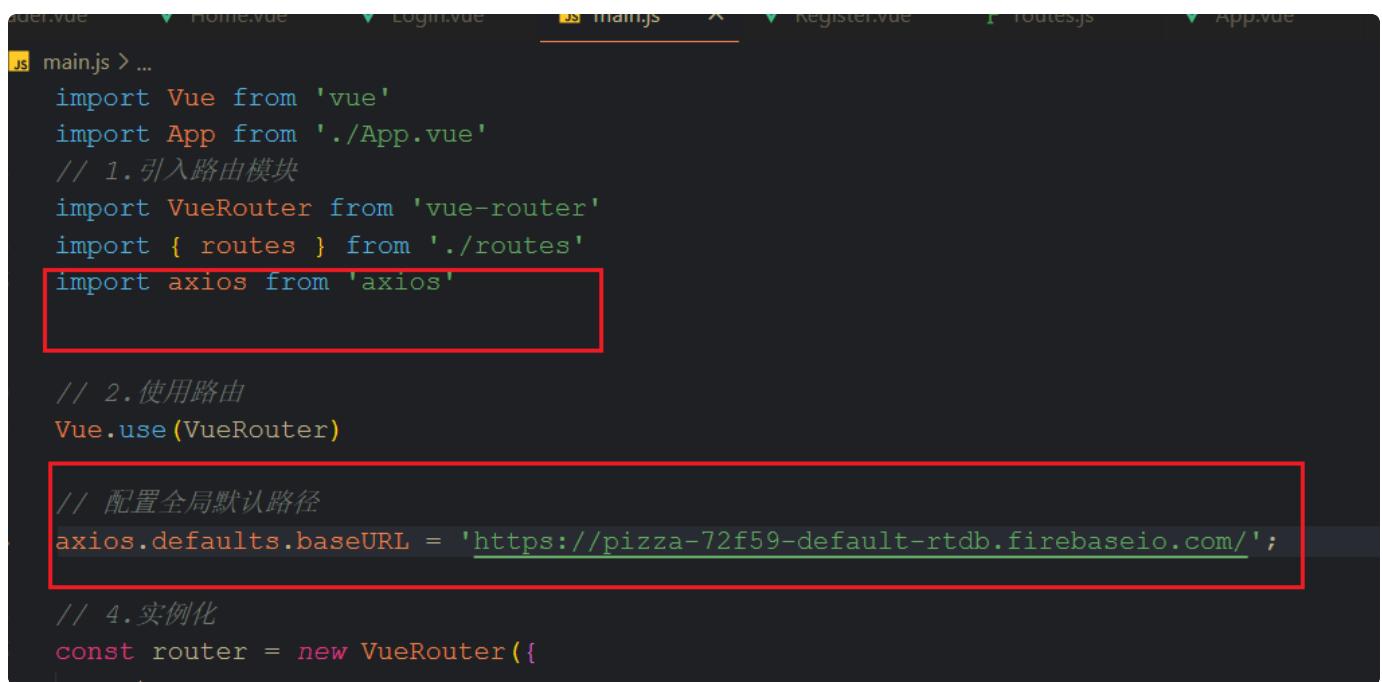
# axios实现数据存储

<https://console.firebaseio.google.com/>

## 1. 下载axios

```
npm install axios --save
```

## 2. 配置axios



The screenshot shows a code editor with several files listed in the sidebar: user.vue, Home.vue, Login.vue, main.js (selected), Register.vue, routes.js, and App.vue. The main.js file contains the following code:

```
main.js > ...
import Vue from 'vue'
import App from './App.vue'
// 1. 引入路由模块
import VueRouter from 'vue-router'
import { routes } from './routes'
import axios from 'axios'

// 2. 使用路由
Vue.use(VueRouter)

// 配置全局默认路径
axios.defaults.baseURL = 'https://pizza-72f59-default-rtdb.firebaseio.com/';

// 4. 实例化
const router = new VueRouter({
```

A red rectangular box highlights the import statement for axios and the baseURL configuration in the axios.defaults object.

## 2. 注册页面

```
<script>
  import axios from "axios";
  export default {
    data() {
      return {
        email: "",
        password: "",
        confirmPassword: "",
      };
    },
    methods: {
      onSubmit() {
        if (this.password === this.confirmPassword) {
          // 存储表单数据
          const formData = {
            email: this.email,
            password: this.password,
            confirmPassword: this.confirmPassword,
          };
          // 通过axios传递表单数据
          axios.post("/users.json", formData).then((res) => {
            console.log(res);
          });
        } else {
          alert("两次密码不一致");
        }
      }
    }
  }

```

2. 注册成功后跳转到登录

```
import axios from "axios";
export default {
  data() {
    return {
      email: "",
      password: "",
      confirmPassword: ""
    };
  },
  methods: {
    onSubmit() {
      if (this.password === this.confirmPassword) {
        // 存储表单数据
        const formData = {
          email: this.email,
          password: this.password,
          confirmPassword: this.confirmPassword,
        };
        // 通过axios传递表单数据
        axios
          .post("/users.json", formData)
          .then((res) => this.$router.push({ name: "loginLink" }));
      } else {
        alert("两次密码不一致");
      }
    },
  },
};
```

### 3. 实现登录

```
<script>
import axios from "axios";
export default {
  data() {
    return {
      email: "",
      password: ""
    };
  },
  methods: {
    onSubmit() {
      // 请求数据
      axios.get("/users.json").then((res) => [
        // console.log(res);
        // 转化返回的对象
        const data = res.data;
        // 放进数组
        const users = [];
        for (let key in data) {
          const user = data[key];
          users.push(user);
        }
        // console.log(users);
        // 实现过滤
        let result = users.filter((user) => {
          return user.email === this.email && user.password === this.password;
        });
        // 判断result
        if (result != null && result.length > 0) {
          this.$router.push({ name: "homeLink" });
        } else {
          alert("账号或密码错误");
        }
      ]);
    }
  }
};
```

## menu页面

```
1 <template>
2   <div>
3     <div class="col-sm-12">
4       <table class="table">
5         <thead class="thead-default">
6           <tr>
7             <th>菜谱</th>
8             <th>价格</th>
9             <th>加入</th>
10            </tr>
11          </thead>
12        <tbody v-for="item in getMenuItems" :key="item.name">
13          <tr>
14            <td>
15              <strong>{{ item.name }}</strong>
16            </td>
17          </tr>
18          <tr v-for="option in item.options" :key="option.size">
19            <td>{{ option.size }}</td>
20            <td>{{ option.price }}</td>
21            <td><button class="btn btn-sm btn-outline-success">+</button>
22            </td>
23          </tr>
24        </tbody>
25      </table>
26    </div>
27  </template>
28 <script>
29 export default {
30   data() {
31     return {
32       getMenuItems: [
33         {
34           name: "爆香五花肉",
35           description: "这是经典川菜",
36           options: [
37             {
38               size: "中份",
39               price: "33",
40             },
41             {
42               size: "大份",
43               price: "40",
44             },
45           ],
46         }
47       ]
48     }
49   }
50 }
51 
```

```
45          ],
46      },
47  ▼    2: {
48      name: "鱼香肉丝",
49      description: "这是经典川菜",
50  ▼    options: [
51  ▼      {
52          size: "中份",
53          price: "33",
54      },
55  ▼      {
56          size: "大份",
57          price: "40",
58      },
59      ],
60  },
61  ▼    3: {
62      name: "梅菜扣肉",
63      description: "这是经典川菜",
64  ▼    options: [
65  ▼      {
66          size: "中份",
67          price: "33",
68      },
69  ▼      {
70          size: "大份",
71          price: "40",
72      },
73      ],
74  },
75  },
76  };
77  },
78  };
79  </script>
```

## 设计购物车

1. 给按钮绑定方法，并传值

```
<td>
  <button
    class="btn btn-sm btn-outline-success"
    @click="addToBasket(item, option)"
  >
    +
  </button>
</td>
```

## 2. 存储值

```
export default {
  data() {
    return {
      baskets: [],
      getMenuItems: { ... },
    },
  },
  methods: {
    addToBasket(item, option) {
      this.baskets.push({
        name: item.name,
        size: option.size,
        price: option.price,
        quantity: 1,
      });
    },
  },
}
</script>
```

## 3. 购物车结构

```
1 <template>
2   <div class="row">
3     <!-- 菜单 -->
4     <div class="col-sm-12 col-md-8">
5       <table class="table">
6         <thead class="thead-default">
7           <tr>
8             <th>菜谱</th>
9             <th>价格</th>
10            <th>加入</th>
11          </tr>
12        </thead>
13        <tbody v-for="item in getMenuItems" :key="item.name">
14          <tr>
15            <td>
16              <strong>{{ item.name }}</strong>
17            </td>
18          </tr>
19          <tr v-for="option in item.options" :key="option.size">
20            <td>{{ option.size }}</td>
21            <td>{{ option.price }}</td>
22            <td>
23              <button
24                class="btn btn-sm btn-outline-success"
25                @click="addToBasket(item, option)"
26              >
27                +
28              </button>
29            </td>
30          </tr>
31        </tbody>
32      </table>
33    </div>
34    <!-- 购物车-->
35    <div class="col-md-4 col-sm-12">
36      <table class="table">
37        <thead class="thead-default">
38          <tr>
39            <th>数量</th>
40            <th>品种</th>
41            <th>价格</th>
42          </tr>
43        </thead>
44        <tbody>
45          <tr>
```

```
46 ▼          <td>
47 ▼          <button class="btn btn-sm">-</button>
48 ▼          <span>1</span>
49 ▼          <button class="btn btn-sm">+</button>
50          </td>
51 ▼          <td>ssss</td>
52 ▼          <td>ssss</td>
53          </tr>
54          </tbody>
55          </table>
56 ▼          <p>总价: </p>
57 ▼          <button class="btn btn-success btn-block">提交</button>
58          </div>
59          </div>
60      </template>
61 ▼      <script>
62 ▼      export default {
63 ▼      data() {
64 ▼      return {
65          baskets: [],
66          getMenuItems: {
67 ▼          1: {
68              name: "爆香五花肉",
69              description: "这是经典川菜",
70              options: [
71 ▼          {
72                  size: "中份",
73                  price: "33",
74              },
75 ▼          {
76                  size: "大份",
77                  price: "40",
78              },
79          ],
80      },
81 ▼          2: {
82              name: "鱼香肉丝",
83              description: "这是经典川菜",
84              options: [
85 ▼          {
86                  size: "中份",
87                  price: "33",
88              },
89 ▼          {
89 ▼          size: "大份",
90                  price: "40",
91              },
92          ],
93      }
94      }
95      }
96      </script>
```

```
94      },
95  ▼    3: {
96      name: "梅菜扣肉",
97      description: "这是经典川菜",
98  ▼    options: [
99  ▼      {
100         size: "中份",
101         price: "33",
102     },
103  ▼     {
104         size: "大份",
105         price: "40",
106     },
107     ],
108   },
109   },
110   };
111 },
112 ▼  methods: {
113   ▼  addToBasket(item, option) {
114     this.baskets.push({
115       name: item.name,
116       size: option.size,
117       price: option.price,
118       quantity: 1,
119     });
120   },
121   },
122   };
123 </script>
```

## 展示购物车数据

1. 判断购物车是否为空

```
<div v-if="baskets.length > 0">
  <table class="table">
    <thead class="thead-default">
      <tr>
        <th>数量</th>
        <th>品种</th>
        <th>价格</th>
      </tr>
    </thead>
    <tbody>
      <tr><td>1</td><td>苹果</td><td>10</td></tr>
    </tbody>
  </table>
</div>
<div v-else>
  {{ basketText }}  

</div>
</div>
</div>
</template>
<script>
import default {
  data() {
    return {
      baskets: [],
      basketText: "请添加商品到购物车中",
      getMenuItems: { ... }
    }
  }
}
</script>
```

## 判断购物车 是否为空

### 为空展示的数据

## 2. 展示购物车数据

```
<!-- 购物车-->
<div class="col-md-4 col-sm-12">
  <div v-if="baskets.length > 0">
    <table class="table">
      <thead class="thead-default">
        <tr>
          <th>数量</th>
          <th>品种</th>
          <th>价格</th>
        </tr>
      </thead>
      <tbody v-for="item in baskets" :key="item.name">
        <tr>
          <td>
            <button class="btn btn-sm">-</button>
            <span>{{ item.quantity }}</span>
            <button class="btn btn-sm">+</button>
          </td>
          <td>{{ item.name }} {{ item.size }}</td>
          <td>{{ item.price * item.quantity }}</td>
        </tr>
      </tbody>
    </table>
  <p>总价: </p>
```

渲染数据

### 3. 给增加和减少按钮绑定方法

```
</thead>
<tbody v-for="item in baskets" :key="item.name">
  <tr>
    <td>
      <button class="btn btn-sm" @click="decreaseQuantity(item)">
        -
      </button>
      <span>{{ item.quantity }}</span>
      <button class="btn btn-sm" @click="increaseQuantity(item)">
        +
      </button>
    </td>
    <td>{{ item.name }} {{ item.size }}</td>
    <td>{{ item.price * item.quantity }}</td>
  </tr>
</tbody>
</table>
```

#### 4. 实现增加和减少

```
        },
      methods: {
        addToBasket(item, option) {
          this.baskets.push({
            name: item.name,
            size: option.size,
            price: option.price,
            quantity: 1,
          });
        },
        decreaseQuantity(item) {
          item.quantity--;
          if (item.quantity <= 0) {
            this.removeFromBasket(item);
          }
        },
        increaseQuantity(item) {
          item.quantity++;
        },
        // 清除购物车中的东西
        removeFromBasket(item) {
          this.baskets.splice(this.baskets.indexOf(item), 1);
        },
      };
    
```

## 5. 展示总价格

```
<template>
    <p>总价: {{ total + "RMB" }}</p>
    <button class="btn btn-success btn-block">提交</button>
</div>
<div v-else>
    {{ basketText }}
</div>
</div>
</div>
</template>
<script>
export default {
  data() {
    return {
      baskets: [],
      basketText: "请添加商品到购物车中",
      getMenuItems: { ... },
    },
  },
  // 动态展示的数据
  computed: {
    total() {
      // 遍历整个购物车
      let totalCost = 0;
      for (let index in this.baskets) {
        let individualItem = this.baskets[index];
        totalCost += individualItem.quantity * individualItem.price;
      }
      return totalCost;
    },
  },
}
```

## 6. 完善增加的方法

```
methods: {
  addToBasket(item, option) {
    let basket = {
      name: item.name,
      size: option.size,
      price: option.price,
      quantity: 1,
    };
    if (this.baskets.length > 0) {
      // 过滤
      let result = this.baskets.filter((basket) => {
        return basket.name === item.name && basket.price === option.price;
      });
      if (result != null && result.length > 0) {
        result[0].quantity++;
      } else {
        this.baskets.push(basket);
      }
    } else {
      this.baskets.push(basket);
    }
  },
  decreaseQuantity(item) {
    item.quantity--;
    if (item.quantity <= 0) {
      this.removeFromBasket(item);
    }
  }
},
```

## 管理页面

### 1. 结构

```
1 ▼ <template>
2 ▼   <div class="row">
3 ▼     <div class="col-sm-12 col-md-8">
4       <!-- 新菜谱 -->
5
6     </div>
7 ▼   <div class="col-sm-12 col-md-4">
8     <!-- 品种展示 -->
9 ▼       <h3 class="text-muted my-5">菜单</h3>
10 ▼      <table class="table">
11 ▼        <thead class="table table-default">
12 ▼          <tr>
13 ▼            <th>品种</th>
14 ▼            <th>删除</th>
15          </tr>
16        </thead>
17 ▼      <tbody>
18        <td>爆香五花肉</td>
19        <td>
20          <button class="btn btn-outline-danger btn-sm">&times;;
21        </td>
22      </tbody>
23    </table>
24  </div>
25 </div>
26 </template>
27
28 ▼ <script>
29   export default {};
30 </script>
```

## 2. 新建newCook.vue组件

```
1 <template>
2   <form>
3     <h3 class="text-muted my-5">添加新的菜单</h3>
4     <div class="form-group row">
5       <label class="col-sm-1">品种</label>
6       <div class="col-sm-11">
7         <input type="text" class="form-control" v-model="newCook.name" />
8       </div>
9     </div>
10    <div class="form-group row">
11      <label class="col-sm-1">描述</label>
12      <div class="col-sm-11">
13        <textarea
14          rows="5"
15          class="form-control"
16          v-model="newCook.description"
17        ></textarea>
18      </div>
19    </div>
20    <p><strong>选项1</strong></p>
21    <div class="form-group row">
22      <label class="col-sm-1">尺寸</label>
23      <div class="col-sm-11">
24        <input type="text" class="form-control" v-model="newCook.size1" />
25      </div>
26    </div>
27    <div class="form-group row">
28      <label class="col-sm-1">价格</label>
29      <div class="col-sm-11">
30        <input type="text" class="form-control" v-model="newCook.price1" />
31      </div>
32    </div>
33    <p><strong>选项2</strong></p>
34    <div class="form-group row">
35      <label class="col-sm-1">尺寸</label>
36      <div class="col-sm-11">
37        <input type="text" class="form-control" v-model="newCook.size2" />
38      </div>
39    </div>
40    <div class="form-group row">
41      <label class="col-sm-1">价格</label>
42      <div class="col-sm-11">
```

```
43             <input type="text" class="form-control" v-model="newCook.price2"
44         />
45     </div>
46     </div>
47     <div class="form-group row">
48         <button class="btn btn-success btn-block" type="button">添加
49     </button>
50     </div>
51 </template>
52 <script>
53 export default {
54     data() {
55         return {
56             newCook: {},
57         };
58     },
59 }
60 </script>
```

### 3. 在admin.vue中

```
<template>
  <div class="row">
    <div class="col-sm-12 col-md-8">
      <!-- 新菜谱 -->
      <app-new-cook></app-new-cook> 3
    </div>
    <div class="col-sm-12 col-md-4">
      <!-- 品种展示 -->
      <h3 class="text-muted my-5">菜单</h3>
      <table class="table">
        <thead class="table table-default">
          <tr>
            <th>品种</th>
            <th>删除</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>爆香五花肉</td>
            <td>
              <button class="btn btn-outline-danger btn-sm">&times;</button>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</template>

<script>
import NewCook from './NewCook.vue'; 1
export default {
  components: {
    "app-new-cook": NewCook, 2
  }
}
```

## 请求数据

1. 给按钮绑定事件

```
</div>
<div class="form-group row">
  <button
    @click="addMenuItem"
    class="btn btn-success btn-block"
    type="button">
    添加
  </button>
</div>
</form>
```

## 2. 使用fetch方法提交数据

```
methods: {
  addMenuItem() {
    // console.log(this.newCook);
    let data = {
      name: this.newCook.name,
      description: this.newCook.description,
      options: [
        { size: this.newCook.size1, price: this.newCook.price1 },
        { size: this.newCook.size2, price: this.newCook.price2 },
      ],
    };
    // es6 fetch传递数据
    fetch("https://pizza-72f59-default-rtdb.firebaseio.com/menu.json", {
      method: "POST",
      headers: { "Content-type": "application/json" },
      body: JSON.stringify(data),
    }).then((res) => {
      console.log(res);
    });
  },
},
```

## 3. 处理数据

添加成功跳转到相应的页面

```
},
methods: {
  addMenuItem() {
    // console.log(this.newCook);
    let data = {
      name: this.newCook.name,
      description: this.newCook.description,
      options: [
        { size: this.newCook.size1, price: this.newCook.price1 },
        { size: this.newCook.size2, price: this.newCook.price2 },
      ],
    };
    // es6 fetch传递数据
    fetch("https://pizza-72f59-default.firebaseio.com/menu.json", {
      method: "POST",
      headers: { "Content-type": "application/json" },
      body: JSON.stringify(data),
    })
      .then((res) => res.json())
      .then((data) => this.$router.push({ name: "menuLink" }));
      .catch((err) => console.log(err));
    },
  },
},
</script>
```

## 删除请求

### 1. 请求数据

```
ler.vue      V Home.vue      V Login.vue      V Admin.vue X V NewCook.vue
components > V Admin.vue > {} "Admin.vue" > script > default > created > then() callback
    </td>
    </tbody>
    </table>
</div>
</div>
</template>

<script>
import NewCook from "./NewCook.vue";
export default {
  data() {
    return {
      getMenuItems: [],
    };
  },
  components: {
    "app-new-cook": NewCook,
  },
  // 拿到数据并赋给属性，展示请求的数据
  created() {
    fetch("https://pizza-72f59-default-firebase.firebaseio.com/menu.json")
      .then((res) => {
        return res.json();
      })
      .then((data) => {
        console.log([data]);
      });
  },
}

```

2. 转换key，赋予id

```
components: {
  "app-new-cook": NewCook,
},
// 拿到数据并赋给属性，展示请求的数据
created() {
  fetch("https://pizza-72f59-default.firebaseio.com/menu.json")
    .then((res) => {
      return res.json();
    })
    .then((data) => {
      // console.log(data);
      // 遍历data，将key值转为唯一id
      let menuArray = [];
      for (let key in data) {
        // console.log(key);
        data[key].id = key;
        menuArray.push(data[key]);
      }
      this.getMenuItems = menuArray;
    });
}
```

### 3. 展示数据

```
<thead>
  <th>品种</th>
  <th>删除</th>
</tr>
</thead>
<tbody v-for="item in getMenuItems" :key="item.name">
  <td>{{ item.name }}</td>
  <td>
    <button class="btn btn-outline-danger btn-sm">&times;</button>
  </td>
</tbody>
```

### 4. 删除按钮绑定方法

```
<button
  class="btn btn-outline-danger btn-sm"
  @click="deleteData(item)">
  &times;
</button>
```

### 5. 实现删除方法

```
methods: {
  deleteData(item) {
    fetch(
      "https://pizza-72f59-default-firebase.firebaseio.com/menu/" +
      item.id +
      ".json",
      {
        method: "DELETE",
        headers: {
          "Content-type": "application/json",
        },
      }
    )
      .then((res) => res.json())
      .then((data) => this.$router.push({ name: "menuLink" }))
      .catch((err) => console.log(err));
  },
};
```

## 请求数据get

同步菜单和管理页面的数据

1. 删除menu.vue的假数据

```
basketText: "请添加商品到购物车中",
getMenuItems: {} ,
};

},
```

2. 请求数据

```
},
// 及时请求数据
created() {
  this.fetchData();
},
methods: {
  fetchData() {
    fetch("https://pizza-72f59-default-firebase.firebaseio.com/menu.json")
      .then((res) => {
        return res.json();
      })
      .then((data) => [
        this.getMenuItems = data;
      ]);
  },
},
```

使用axios请求数据：

```
methods: {
  fetchData() {
    // fetch("https://pizza-72f59-default-firebase.firebaseio.com/menu.json")
    //   .then((res) => {
    //     return res.json();
    //   })
    //   .then((data) => {
    //     this.getMenuItems = data;
    //   });
    // 使用axios
    axios.get("menu.json").then((res) => {
      this.getMenuItems = res.data;
    });
  },
},
```

main.js中配置axios

```
1 import axios from "axios"
2
3 // 2. 使用路由
4 Vue.use(VueRouter)
5
6 // 配置全局默认路径
7 axios.defaults.baseURL = 'https://pizza-72f59-default-firebase.firebaseio.com/';
8
9 // 配置Vue原型(在任何组件中都可以正常使用axios)
10 Vue.prototype.$http = axios;
```

get:

```
methods: {
  fetchData() {
    // fetch("https://pizza-72f59-default-rtdb.firebaseio.com/menu.json")
    //   .then((res) => {
    //     return res.json();
    //   })
    //   .then((data) => {
    //     this.getMenuItems = data;
    //   });
    // 使用axios
    // axios.get("menu.json").then((res) => {
    //   this.getMenuItems = res.data;
    // });
    this.http.get("menu.json").then((res) => (this.getMenuItems = res.data));
  },
},
```

post:

```
this.http
  .post("menu.json", data)
  .then((res) => this.$router.push({ name: "menuLink" }));
```

## Vuex-mutations更改数据状态

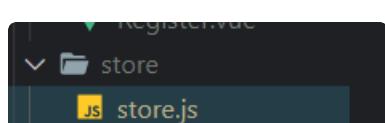
将数据存在vuex中，方便任何组件使用

1. `npm view vuex versions --json`

`npm install vuex@版本 --save`

`npm install vuex --save`

2. 新建store文件夹和文件



store.js内容：

```
1 import Vue from "vue";
2 import Vuex from 'vuex';
3
4 Vue.use(Vuex);
5
6 export const store = new Vuex.Store({
7     state: {
8         // 存储数据
9     },
10    getters: {
11        // 获取数据的方法,获取属性的状态 state
12    },
13    mutations: {
14        // 改变属性的状态
15    },
16    actions: {
17        // 应用mutations
18    }
19 })
20
```

### 3. 在main.js中配置store

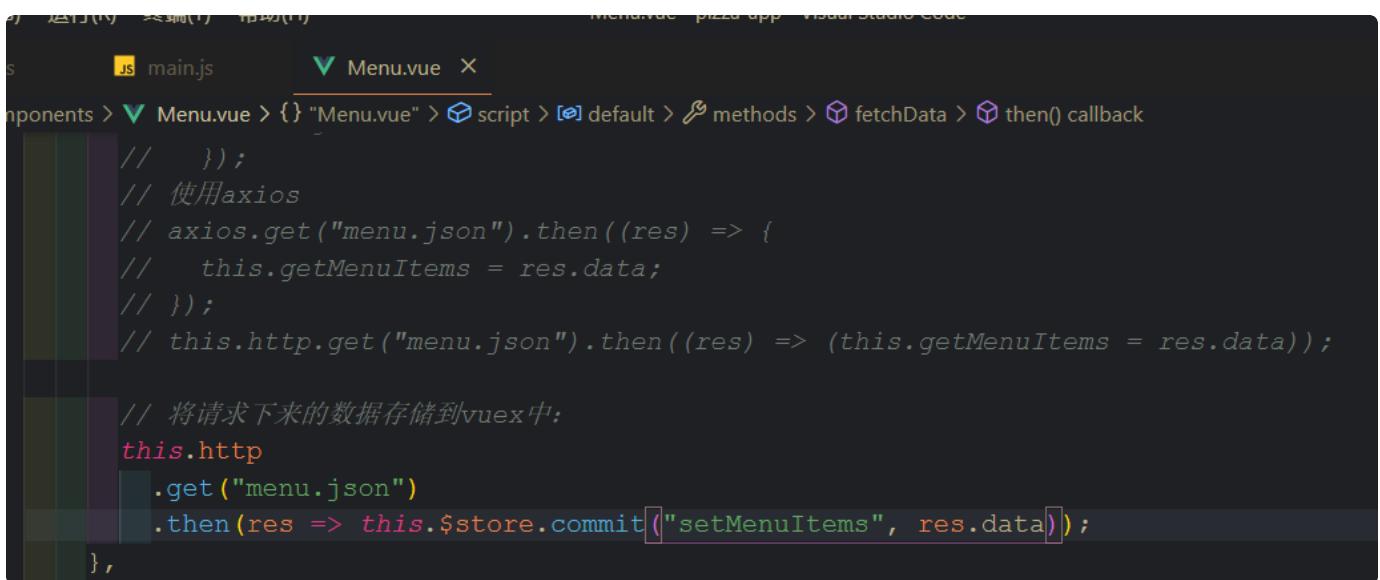
```
import axios from 'axios'
import { store } from './store/store.js'
```

```
new Vue({
  el: '#app',
  // 5. 使用路由
  router,
  store,
  render: h => h(App)
})
```

### 4. 在store.js中预设属性

```
export const store = new Vuex.Store({
  state: {
    // 存储数据
    menuItems: []
  },
})
```

## 5. 在Menu.vue中获取数据并传到state.js中



A screenshot of the Visual Studio Code interface. The left sidebar shows components like main.js and Menu.vue. The right pane displays the code for Menu.vue:

```
components > ▼ Menu.vue > { } "Menu.vue" > script > default > methods > fetchData > then() callback
  //   );
  // 使用axios
  // axios.get("menu.json").then((res) => {
  //   this.getMenuItems = res.data;
  // });
  // this.http.get("menu.json").then((res) => (this.getMenuItems = res.data));

  // 将请求下来的数据存储到vuex中:
  this.http
    .get("menu.json")
    .then(res => this.$store.commit(["setMenuItems", res.data]));
},
```

## 6. 处理接收到的数据并存在menuItems中：

store.js main.js Menu.vue

```
src > store > store.js > mutations > setMenuItems
1 import Vue from "vue";
2 import Vuex from 'vuex';
3
4 Vue.use(Vuex);
5
6 export const store = new Vuex.Store({
7     state: {
8         // 存储数据
9         menuItems: {}
10    },
11    getters: {
12        // 获取数据的方法, 获取属性的状态 state
13    },
14    mutations: {
15        // 改变属性的状态
16        setMenuItems(state, data) {
17            state.menuItems = data;
18        }
19    },
20    actions: {
21        // 应用mutations
22    }
23})
```

7. 修改menu.vue,拿取数据

```
export default {
  data() {
    return {
      baskets: [],
      basketText: "请添加商品到购物车中",
      // getMenuItems: {},
    };
  },
  // 动态展示的数据
  computed: {
    getMenuItems() {
      // 在vuex中获取数据
      return this.$store.state.menuItems;
    },
  },
}
```

## 管理页面同步到vuex

### 1. 修改admin页面

```
import NewCook from './NewCook.vue',
export default {
  data() {
    return {
      // getMenuItems: [],
    };
  },
  components: {
    "app-new-cook": NewCook,
  },
  computed: {
    getMenuItems: {
      // 在vuex中获取数据
      get() {
        return this.$store.state.menuItems;
      },
      set() {}
    },
  },
}
```

### 2. 实现数据同步

```
    },
    // 拿到数据并赋给属性，展示请求的数据
  created() {
    fetch("https://pizza-72f59-default-firebase.firebaseio.com/menu.json")
      .then((res) => {
        return res.json();
      })
      .then((data) => {
        // console.log(data);
        // 遍历data，将key值转为唯一id
        let menuArray = [];
        for (let key in data) {
          // console.log(key);
          data[key].id = key;
          menuArray.push(data[key]);
        }
        // this.getMenuItems = menuArray;
        // 实现数据同步
        this.$store.commit("setMenuItems", menuArray);
      });
  },
  methods: { ... }
```

### 3. 设置新的删除

```
// this.getMenuItems = menuArray;

// 实现数据同步
this.$store.commit("setMenuItems", menuArray);
});

},
methods: {
deleteData(item) {
fetch(
  "https://pizza-72f59-default-firebase.firebaseio.com/menu/" +
    item.id +
    ".json",
  {
    method: "DELETE",
    headers: {
      "Content-type": "application/json",
    },
  }
)
.then((res) => res.json())
// .then((data) => this.$router.push({ name: "menuLink" }))
.then((data) => [
  this.$store.commit("removeMenuItems", item),
])
}

.catch((err) => console.log(err));
},
}
}
```

添加新删除方法

#### 4. 删除方法

```

mutations: {
    // 改变属性的状态
    setMenuItems(state, data) {
        state.menuItems = data;
    },
    // 将匹配到的对象，在menuItems数组中删除
    removeMenuItems(state, data) {
        state.menuItems.forEach((item, index) => {
            if (item == data) {
                state.menuItems.splice(index, 1);
            }
        });
    }
},
actions: {
    // ...
}

```

## 5. newcook.vue中新添加push方法

```

//      .catch((err) => console.log(err));
// this.http
//      .post("menu.json", data)
//      .then((res) => this.$router.push({ name: "menuLink" }));
// 数据同步到vuex
this.http
    .post("menu.json", data)
    .then([res] => this.$store.commit("pushToMenuItems", data));
},
},

```

**添加新的push方法**

## 6. 实现push方法

```
        mutations: {
            // 改变属性的状态
            setMenuItems(state, data) {
                state.menuItems = data;
            },
            // 将匹配到的对象，在menuItems数组中删除
            removeMenuItem(state, data) {
                state.menuItems.forEach((item, index) => {
                    if (item == data) {
                        state.menuItems.splice(index, 1);
                    }
                });
            },
            // 将新添加的食物push到menuItems属性中
            pushToMenuItem(state, data) {
                state.menuItems.push(data);
            }
        },
        actions: {
            // 应用mutations
        }
    })
}
```

## 登录成功显示用户信息

1. 通过getter获取数据

```
> JS store.js > store > getters > getMenuItems
import Vue from "vue";
import Vuex from 'vuex';

Vue.use(Vuex);

export const store = new Vuex.Store({
  state: {
    // 存储数据
    menuItems: {}
  },
  getters: {
    // 获取数据的方法, 获取属性的状态 state
    getMenuItems: state => state.menuItems
  }
});
```

## 2. 更改menu.vue

```
computed: {
  getMenuItems () {
    // 在vuex中获取数据
    // return this.$store.state.menuItems;
    // 通过getters获取数据
    return this.$store.getters.getMenuItems;
  },
  total () {
```

## 3. 更改admin.vue

```
"app-new-cook": NewCook,
},
computed: {
  getMenuItems: {
    // 在vuex中获取数据
    get () {
      return this.$store.getters.getMenuItems;
    },
    set () {},
  },
},
// 拿到数据并赋给属性, 展示请求的数据
created () {
```

#### 4. 设置登录状态

```
Vue.use(Vuex);

export const store = new Vuex.Store({
  state: {
    // 存储数据
    menuItems: {},
    currentUser: null,
    isLogin: false
  },
  getters: [
    // 获取数据的方法, 获取属性的状态 state
    getMenuItems: state => state.menuItems,
    currentUser: state => state.currentUser,
    isLogin: state => state.isLogin
  ],
  mutations: {
    // 改变属性的状态
    setMenuItems(state, data) {
```

#### 设置登录状态

#### 5. 设置登录

```
</ul>
<ul class="navbar-nav ml-auto">
  <li><router-link :to="{name:'loginLink'}" v-show="!isLogin" class="nav-link">登录</router-link></li>
  <li class="nav-link">{{currentUser}}</li>
  <li><router-link :to="{name:'loginLink'}" v-show="isLogin" class="nav-link">
    <a href="" class="nav-link">[退出]</a></router-link></li>
  <li><router-link :to="{name:'registerLink'}" v-show="!isLogin" class="nav-link">注册</router-link></li>
</ul>
</nav>
</template>

<script>
export default {
  data() {
    return {};
  },
  computed: {
    currentUser() {
      return this.$store.getters.currentUser
    },
    isLogin() {
      return this.$store.getters.isLogin
    }
  }
}
```

#### 6. 更改登录状态

运行(R) 终端(T) 帮助(H) Login.vue - pizza-app - Visual Studio Code

Components > Login.vue > {} "Login.vue" > script > default > methods > onSubmit > then() callback

```
    },
    methods: {
        onSubmit() {
            // 请求数据
            axios.get("/users.json").then((res) => {
                // console.log(res);
                // 转化返回的对象
                const data = res.data;
                // 放进数组
                const users = [];
                for (let key in data) {
                    const user = data[key];
                    users.push(user);
                }
                // console.log(users);
                // 实现过滤
                let result = users.filter((user) => {
                    return user.email === this.email && user.password === this.password;
                });
                // 判断result
                if (result != null && result.length > 0) {
                    this.$store.dispatch("setUser", result[0].email);
                    this.$router.push({ name: "homeLink" });
                } else {
                    alert("账号或密码错误");
                    this.$store.dispatch("setUser", null);
                }
            });
        },
    },
},
```

## 7. 设置登录

```
        },
        // 将新添加的食物push到menuItems属性中
        pushToMenuItems(state, data) {
            state.menuItems.push(data);
        },
        // 更改用户的状态信息
        userStatus(state, user) {
            if (user) {
                state.currentUser = user,
                state.isLogin = true
            } else {
                state.currentUser = null,
                state.isLogin = false
            }
        }
    },
    actions: {
        // 应用mutations
        setUser({ commit }, user) {
            commit("userStatus", user)
        }
    }
}
```

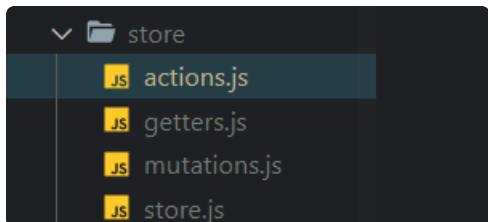
## 8. 组件内守卫

一旦点击退出就回到登录页面，并清空登录信息

```
js      NewCook.vue      main.js      Menu.vue      Admin.vue      Header.vue      Login.vue X
components > Login.vue > {} "Login.vue" > script > [o] default > [o] beforeRouteEnter > [o] next() callback
  data() {
    return {
      email: '',
      password: '',
    };
  },
  // 导航守卫
  beforeRouteEnter: (to, from, next) => {
    next((vm) => vm.$store.dispatch("setUser", null));
  },
  methods: {
    onSubmit() {
      // 请求数据
      axios.get("/users.json").then((res) => {
        // console.log(res);
        // 转化返回的对象
      });
    }
  }
}
```

## 对store文件的代码进行抽离1

1. 新建三个文件



2. 移动actions内容

```
src > store > actions.js > ...
  1  export const setUser = ({ commit }, user) => {
  2    |   commit("userStatus", user)
  3  }
  4  |
```

3. 引入store.js

```
import vuex from 'vuex';
import * as actions from './actions'

Vue.use(Vuex);

const store = new Vuex.Store({
  state: {
    menuItems: [
      { id: 1, name: 'Home' },
      { id: 2, name: 'About' },
      { id: 3, name: 'Contact' }
    ],
    currentUser: null,
    isLogin: false
  },
  mutations: {
    setMenuItems(state, menuItems) {
      state.menuItems = menuItems;
    },
    setCurrentUser(state, user) {
      state.currentUser = user;
    },
    setIsLogin(state, login) {
      state.isLogin = login;
    }
  },
  actions
})
```

#### 4. getters.js

```
store.js      getters.js  X  actions.js

> store > getters.js > getMenuItemList
1 // 获取数据的方法,获取属性的状态 state
2 export const getMenuItems = state => state.menuItems
3 export const currentUser = state => state.currentUser
4 export const isLogin = state => state.isLogin
```

#### 5. 引入store.js

```
store.js      X   getters.js      actions.js

src > store > store.js > [e] store
1  import Vue from "vue";
2  import Vuex from 'vuex';
3  import * as actions from './actions'
4  import * as getters from './getters' [red box]
5
6
7  Vue.use(Vuex);
8
9  export const store = new Vuex.Store([
10    state: {
11      // 存储数据
12      menuItems: {},
13      currentUser: null,
14      isLoggedIn: false
15    },
16    getters, [red box]
17    // // 获取数据的方法,获取属性的状态 state
18    // getMenuItems: state => state.menuItems,
19    // currentUser: state => state.currentUser,
20    // isLoggedIn: state => state.isLoggedIn
21    mutations: {
22      // ...
23    }
24  ])

```

## 6. mutations:

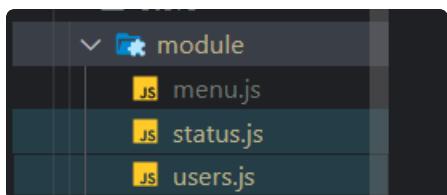
```
JS store.js      JS mutations.js X  JS getters.js      JS actions.js
src > store > JS mutations.js > [?] pushToMenuItems
1 // 改变属性的状态
2 export const setMenuItems = (state, data) => {
3     state.menuItems = data;
4 }
5 // 将匹配到的对象，在menuItems数组中删除
6 export const removeMenuItems = (state, data) => {
7     state.menuItems.forEach((item, index) => {
8         if (item == data) {
9             state.menuItems.splice(index, 1);
10        }
11    });
12 }
13
14 }
15 // 将新添加的食物push到menuItems属性中
16 export const pushToMenuItems = (state, data) => [
17     state.menuItems.push(data),
18 ]
19 // 更改用户的状态信息
20 export const userStatus = (state, user) => {
21     if (user) {
22         state.currentUser = user,
23         state.isLogin = true
24     } else {
25         state.currentUser = null,
26         state.isLogin = false
27     }
28 }
```

## 7. store

```
store.js      X  mutations.js      getters.js      actions.js
src > store > store.js > ...
1 import Vue from "vue";
2 import Vuex from 'vuex';
3 import * as actions from './actions'
4 import * as getters from './getters'
5 import * as mutations from './mutations'
6
7
8
9 Vue.use(Vuex);
10
11 export const store = new Vuex.Store({
12     state: {
13         // 存储数据
14         menuItems: {},
15         currentUser: null,
16         isLoggedIn: false
17     },
18     getters,
19
20     mutations,
21     actions
22 })
23
```

## 使用Module实现模块化

### 1. 新建文件



### 2. 抽离

```
src > JS store.js > [e] store > ↗ state
  import * as getters from './mutations'
  Vue.use(Vuex);

  export const store = new Vuex.Store({
    state: {
      // 存储数据
      // menuItems: {},
      // currentUser: null,
      // isLoggedIn: false
    },
    getters,
    mutations,
    actions
  })
}

re.js      JS menu.js  X  JS users.js      JS status.js
src > store > module > JS menu.js > [e] state
  const state = {
    menuItems: {}
  }

  const getters = {}

  const mutations = {}

  const actions = {}

  export default {
    state,
    getters,
    mutations,
    actions
  }
```

```
src > JS store.js > [e] store > ↗ state
  import * as getters from './mutations'
  Vue.use(Vuex);

  export const store = new Vuex.Store({
    state: {
      // 存储数据
      // menuItems: {},
      // currentUser: null,
      // isLoggedIn: false
    },
    getters,
    mutations,
    actions
  })
}

status.js ->
  转到(G) 运行(R) 终端(T) 帮助(H)
  JS store.js  JS menu.js  JS users.js  JS status.js  X
src > store > module > JS status.js > [e] state > ↗ isLoggedIn
  1 ✓ const state = {
  2   [
  3     isLoggedIn: false
  4   ]
  5   const getters = {}
  6
  7   const mutations = {}
  8
  9   const actions = {}
 10
 11 ✓ export default {
 12   state,
 13   getters,
 14   mutations,
 15   actions
 16 }
```

```

store.js
import * as getters from './mutations'
Vue.use(Vuex);

export const store = new Vuex.Store({
  state: {
    // 存储数据
    // menuItems: {},
    // currentUser: null,
    // isLoggedIn: false
  },
  getters,
  mutations,
  actions
})

```

```

src > store > module > users.js > state
const state = {
  currentUser: null,
}
const getters = {}
const mutations = {}
const actions = {}

export default {
  state,
  getters,
  mutations,
  actions
}

```

### 3. 抽离getter

```

store > module > menu.js > getters > getMenuItems
const state = {
  menuItems: {}
}

const getters = {
  getMenuItems: state => state.menuItems,
}

const mutations = {}

const actions = {}

export default {
  state,
  getters,
  mutations,
  actions
}

```

```

: > store > getters.js
1 // 获取数据的方法,获取属性的状态 state
2 // export const getMenuItems = state => state.menuItems
3 // export const currentUser = state => state.currentUser
4 // export const isLoggedIn = state => state.isLoggedIn

```

```
src > store > module > users.js > getters > currentUser
1 const state = {
2   |   currentUser: null,
3 }
4
5 const getters = [
6   |   currentUser: state => state.currentUser
7 ]
8
9
10 const mutations = {}
11
12 const actions = {}
13
14 export default {
15   |   state,
16   |   getters,
17   |   mutations,
```

> store > getters.js

```
1 // 获取数据的方法,获取属性的状态 state
2 // export const getMenuItems = state => state.menuItems
3 // export const currentUser = state => state.currentUser
4 // export const isLogin = state => state.isLogin
```

```
src > store > module > status.js > getters > isLogin
1 const state = {
2   |   isLogin: false
3 }
4
5 const getters = [
6   |   isLogin: state => state.isLogin
7 ]
8
9
10 const mutations = {}
11
12 const actions = {}
13
14 export default {
15   |   state,
16   |   getters,
17   |   mutations,
```

> store > getters.js

```
1 // 获取数据的方法,获取属性的状态 state
2 // export const getMenuItems = state => state.menuItems
3 // export const currentUser = state => state.currentUser
4 // export const isLogin = state => state.isLogin
```

#### 4. 抽离mutations:

```
store.js          mutations.js
src > store > module > menu.js > mutations > pushToMenuItems
1  const state = {
2    menuItems: []
3  }
4
5  const getters = {
6    getMenuItems: state => state.menuItems,
7  }
8
9
10 const mutations = {
11   setMenuItems(state, data) {
12     state.menuItems = data;
13   },
14   // 将匹配到的对象，在menuItems数组中删除
15   removeMenuItems(state, data) {
16     state.menuItems.forEach((item, index) => {
17       if (item == data) {
18         state.menuItems.splice(index, 1);
19       }
20     });
21   },
22   // 将新添加的食物push到menuItems属性中
23   pushToMenuItems(state, data) {
24     state.menuItems.push(data);
25   }
26 }
27
28 const actions = {}
```

```
// 改变属性的状态
export const setMenuItems = (state, data) => {
  state.menuItems = data;
}

// 将匹配到的对象，在menuItems数组中删除
export const removeMenuItems = (state, data) => {
  state.menuItems.forEach((item, index) => {
    if (item == data) {
      state.menuItems.splice(index, 1);
    }
  });
}

// 将新添加的食物push到menuItems属性中
export const pushToMenuItems = (state, data) => {
  state.menuItems.push(data);
}

// 更改用户的状态信息
export const userStatus = (state, user) => {
  if (user) {
    state.currentUser = user,
    state.isLogin = true
  } else {
    state.currentUser = null,
    state.isLogin = false
  }
}
```

```
store.js          menu.js      users.js  status.js    mutations.js   getters.js  actions.js
rc > store > module > users.js > [x] mutations > userStatus
1  const state = {
2    currentUser: null,
3  }
4
5  const getters = {
6    currentUser: state => state.currentUser
7
8  }
9
10 const mutations = {
11   // 更改用户的状态信息
12   userStatus(state, user) {
13     if (user) {
14       state.currentUser = user,
15       state.isLogin = true
16     } else {
17       state.currentUser = null,
18       state.isLogin = false
19     }
20   }
21 }
22
23 const actions = {}
24
25 export default {
26   state,
27   getters,
28   mutations,
29   actions
30 }
```

```
mutations.js
// 改变属性的状态
export const setMenuItems = (state, data) => {
  state.menuItems = data;
}

// 将匹配到的对象，在menuItems数组中删除
export const removeMenuItem = (state, data) => {
  state.menuItems.forEach((item, index) => {
    if (item == data) {
      state.menuItems.splice(index, 1);
    }
  });
}

// 将新添加的食物push到menuItems属性中
export const pushToMenuItems = (state, data) => {
  state.menuItems.push(data);
}

// 更改用户的状态信息
export const userStatus = (state, user) => {
  if (user) {
    state.currentUser = user,
    state.isLogin = true
  } else {
    state.currentUser = null,
    state.isLogin = false
  }
}
```

## 5. 抽离action

```
src > module > users.js > actions > setUser
const state = {
  currentUser: null,
}

const getters = {
  currentUser: state => state.currentUser
}

const mutations = {
  // 更改用户的 (parameter) state: any
  userStatus(state, user) {
    if (user) {
      state.currentUser = user,
      state.isLogin = true
    } else {
      state.currentUser = null,
      state.isLogin = false
    }
  }
}

const actions = {
  setUser({ commit }, user) {
    commit("userStatus", user)
  }
}
```

```
src > store > actions.js > ...
1  export const setUser = ({ commit }, user) => {
2    |   commit("userStatus", user)
3    |
4  }
```

## 6. 引入mudoles

store.js X menu.js users.js status.js mutations.js

```
src > store > store.js > [e] store > modules > status
  1 import Vue from "vue";
  2 import Vuex from 'vuex';
  3 // import * as actions from './actions'
  4 // import * as getters from './getters'
  5 // import * as mutations from './mutations'
  6 import menu from './module/menu';
  7 import users from './module/users';
  8 import status from './module/status';
  9
 10
 11
 12
 13 Vue.use(Vuex);
 14
 15 export const store = new Vuex.Store({
 16   // state: {
 17   //   // 存储数据
 18   //   // menuItems: {},
 19   //   // currentUser: null,
 20   //   // isLoggedIn: false
 21   // },
 22   // getters,
 23
 24   // mutations,
 25   // actions
 26   modules: [
 27     users,
 28     menu,
 29     status
 30   ]
 31 })
 32
```