Cook-Manage

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bootstrop4

1. 在index.js中引入bootsrap4的链接

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

2. header.vue基础内容

Vue | 复制代码

```vue
<template>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="py-2" href="#" aria-label="Product">
      <svg
        xmlns="http://www.w3.org/2000/svg"
        width="24"
        height="24"
        fill="none"
        stroke="currentColor"
        stroke-linecap="round"
        stroke-linejoin="round"
        stroke-width="2"
        class="d-block mx-auto"
        role="img"
        viewBox="0 0 24 24"
        focusable="false"
      >
        <title>Product</title>
        <circle cx="12" cy="12" r="10" />
        <path
          d="M14.31 8l5.74 9.94M9.69 8h11.48M7.38 12l5.74-9.94M9.69 16L3.95 6.06M14.31 16H2.83m13.79-4l-5.74 9.94"
        />
      </svg>
    </a>
    <a href="/" class="navbar-brand">Pizza点餐系统</a>
    <ul class="navbar-nav">
      <li><a href="#" class="nav-link">主页</a></li>
      <li><a href="#" class="nav-link">菜单</a></li>
      <li><a href="#" class="nav-link">管理</a></li>
      <li><a href="#" class="nav-link">关于我们</a></li>
    </ul>
    <ul class="navbar-nav ml-auto">
      <li><a href="#" class="nav-link">登录</a></li>
      <li><a href="#" class="nav-link">注册</a></li>
    </ul>
  </nav>
</template>
```

3. 将header.vue的内容展示到页面上

```vue
<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
App.vue > {} "App.vue" > template > div#app > div.container > route
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 }
```

```
<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="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>
```

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

```
<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>
```

```
<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

```vue
<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

```vue
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，

```vue
<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>
```

Vue | 复制代码

```vue
<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>

```

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

```
Home.vue  About.vue  Untitled-1  Contact.vue ×
omponents > about > Contact.vue > template
<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

```
r.vue  Home.vue  main.js  Phone.vue  PersonName.vue  Contact.vue
mponents > 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');
})
```

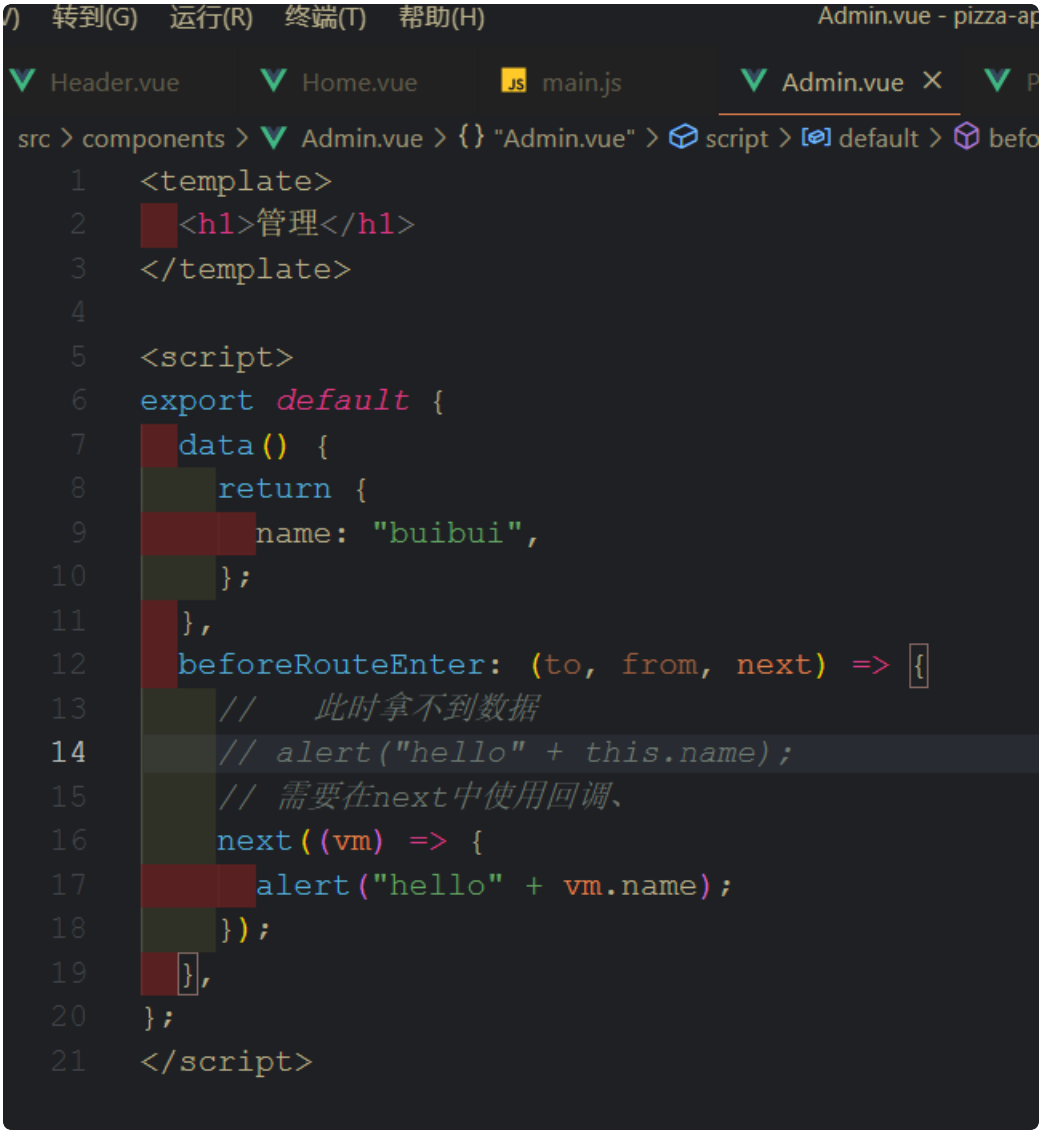
3. 路由独享守卫

与全局守卫作用范围不同

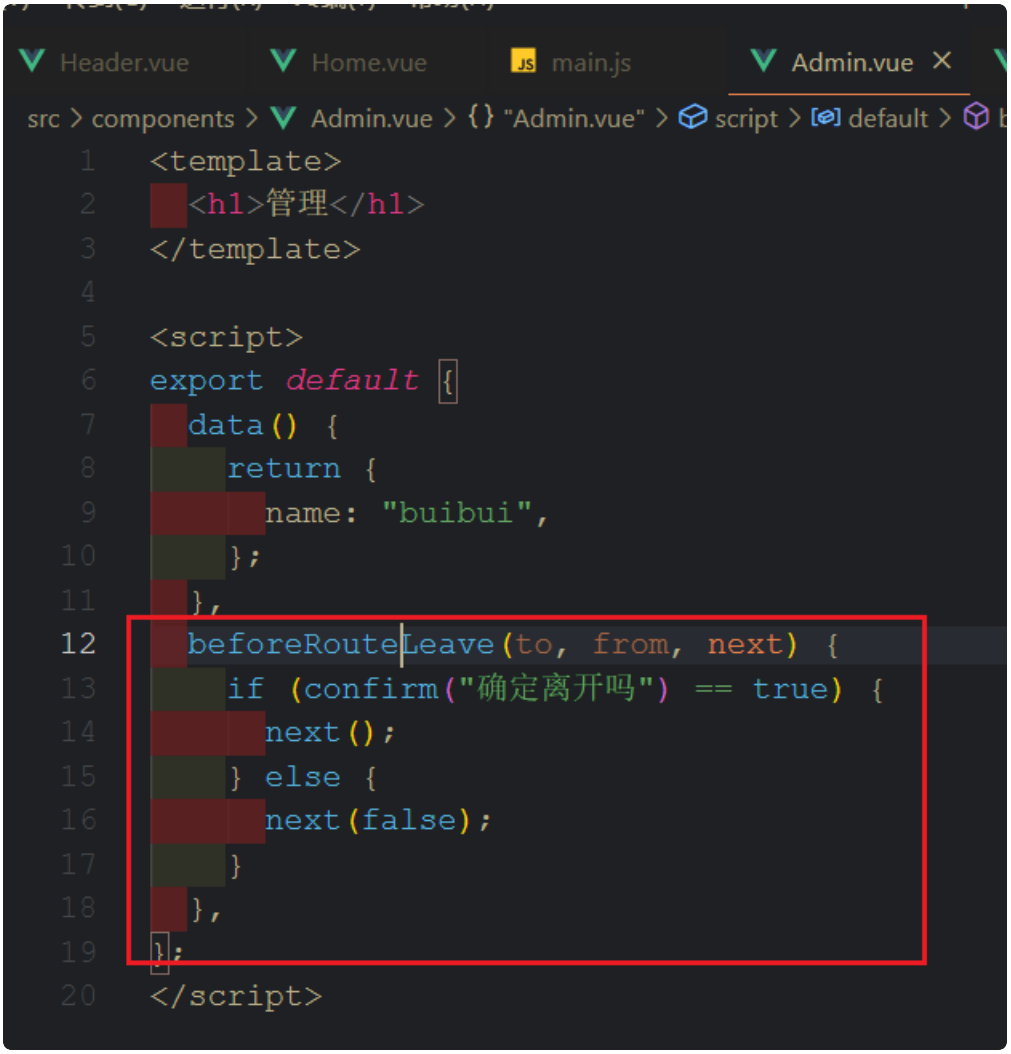
```
      ]
    },
    { path: '/about/History', name: 'historyLink', component: History },
    { path: '/about/delivery', name: 'deliveryLink', component: Delivery },
    { path: '/about/orderingGuide', name: 'orderingGuideLink', component: OrderingGuide },

  ]
},
{ 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：


转到(G) 运行(R) 终端(T) 帮助(H)
Admin.vue - pizza-ap
Header.vue
Home.vue
main.js
Admin.vue
src > components > Admin.vue > {} "Admin.vue" > script > default > befo
<template>
<h1>管理</h1>
</template>
<script>
export default {
data() {
return {
name: "buibui",
};
},
beforeRouteEnter: (to, from, next) => {
// 此时拿不到数据
// alert("hello" + this.name);
// 需要在next中使用回调、
next((vm) => {
alert("hello" + vm.name);
});
},
};
</script>

离开组件：



复用router-view

1. 抽离main.js中的路由配置

新建routes.js文件

```
der.vue   Home.vue   main.js   routes.js
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)

// 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)
})
```

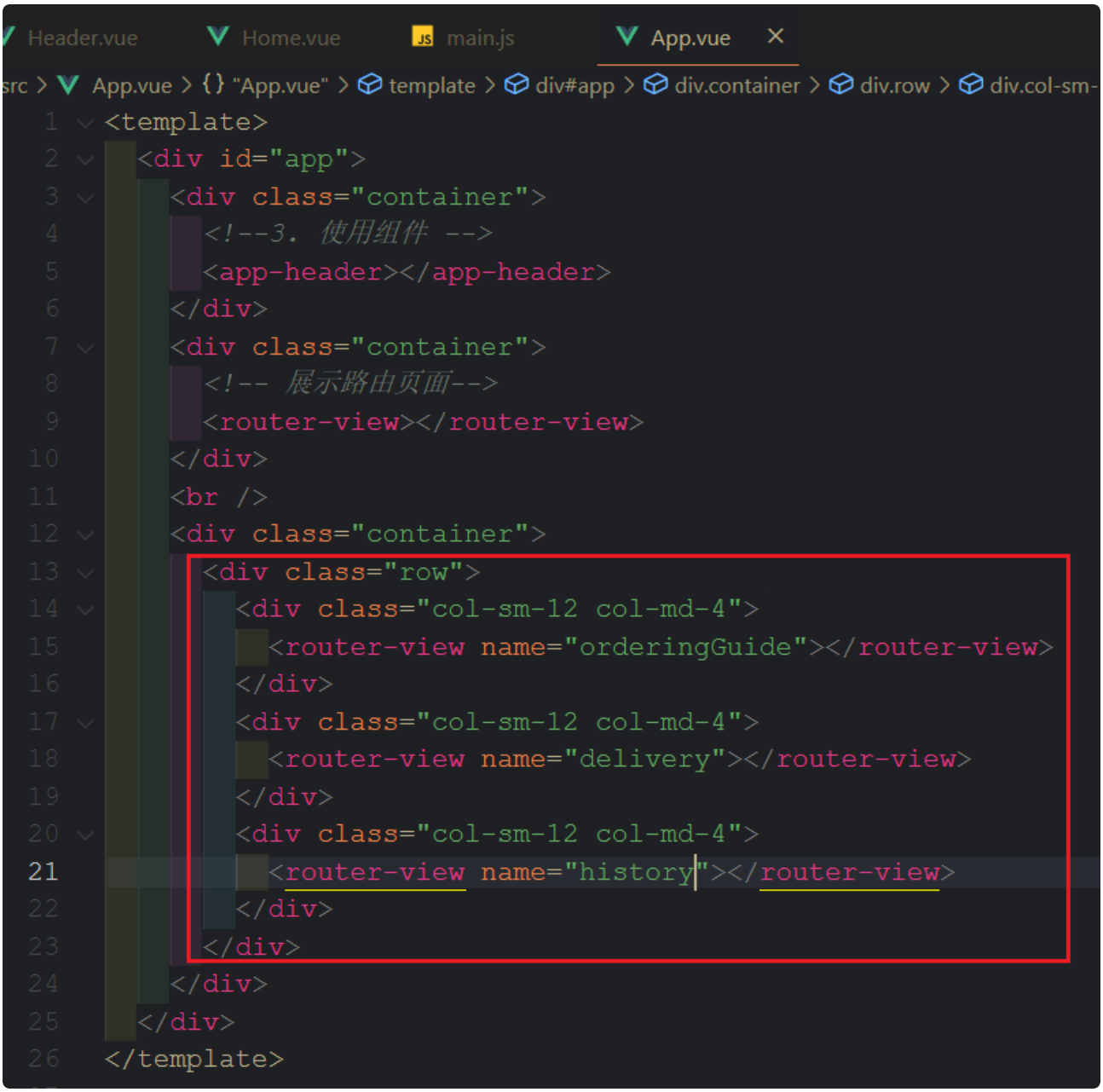
```
export const routes = [
    { path: '/', name: 'homeLink', component: Home },
    { path: '/menu', name: 'menuLink', component: Menu },
    { path: '/Login', name: 'loginLink', component: Login },
```

Vue | 复制代码

```vue
import Home from './components/Home'
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'

// 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'

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


export 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 },

        ]
```

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

2. 复用router-view



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

Vue | 复制代码

```vue
<template>
  <div class="row mt-3">
    <div class="col-md-12 col-lg-12">
      <div class="card">
        <div class="card-body">
          <img class="mx-auto d-block" src="../../src/assets/logo.png" alt="" />
          <form action="" @submit.prevent="onSubmit">
            <div class="form-group">
              <label for="email">邮箱</label>
              <input type="email" class="form-control" v-model="email" />
            </div>
            <div class="form-group">
              <label for="password">密码</label>
              <input type="password" class="form-control" v-model="password" />
            </div>
            <div class="form-group">
              <label for="confirm-password">确认密码</label>
              <input
                type="password"
                class="form-control"
                v-model="confirmPassword"
              />
            </div>
            <button type="submit" class="btn btn-block btn-success">
              注册
            </button>
          </form>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      email: "",
      password: "",
      confirmPassword: "",
    };
  },
  methods: {
```

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

登录:

Vue | 复制代码

```vue
<template>
  <div class="row mt-3">
    <div class="col-md-12 col-lg-12">
      <div class="card">
        <div class="card-body">
          <img class="mx-auto d-block" src="../../src/assets/logo.png" alt="" />
          <form action="" @submit.prevent="onSubmit">
            <div class="form-group">
              <label for="email">邮箱</label>
              <input type="email" class="form-control" v-model="email" />
            </div>
            <div class="form-group">
              <label for="password">密码</label>
              <input type="password" class="form-control" v-model="password" />
            </div>

            <button type="submit" class="btn btn-block btn-success">
              登录
            </button>
          </form>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      email: "",
      password: "",
    };
  },
  methods: {
    onSubmit() {},
  },
};
</script>
```

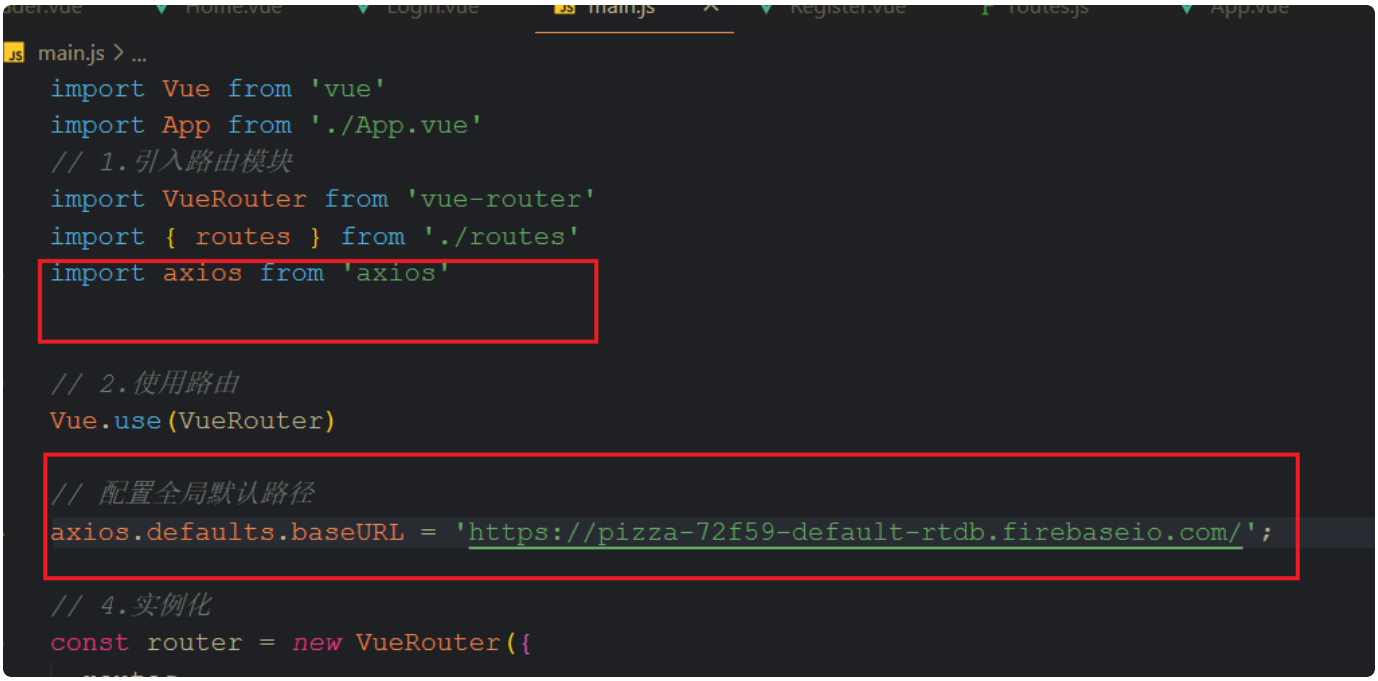
axios实现数据存储

https://console.firebase.google.com/

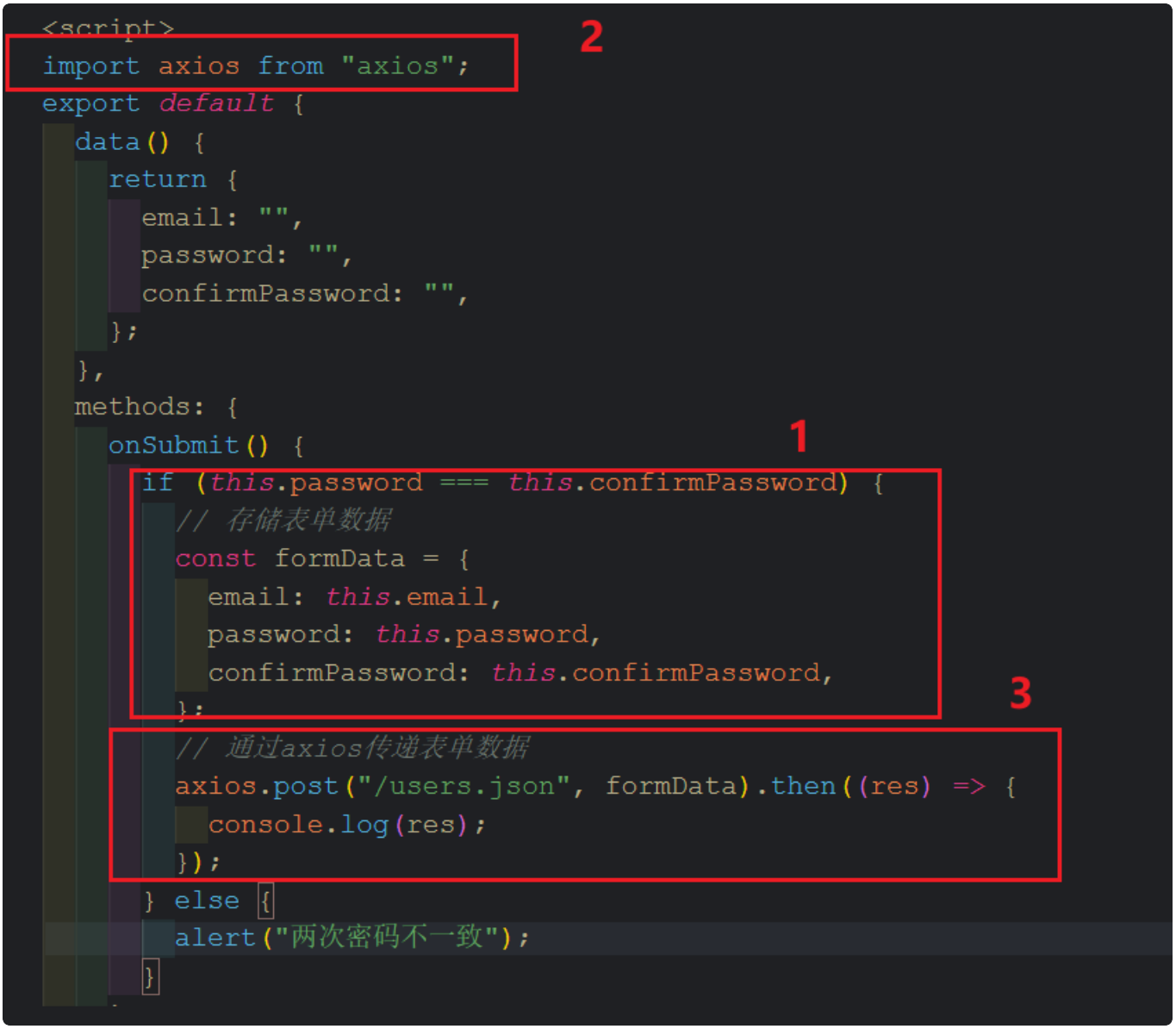
1. 下载axios

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

2. 配置axios



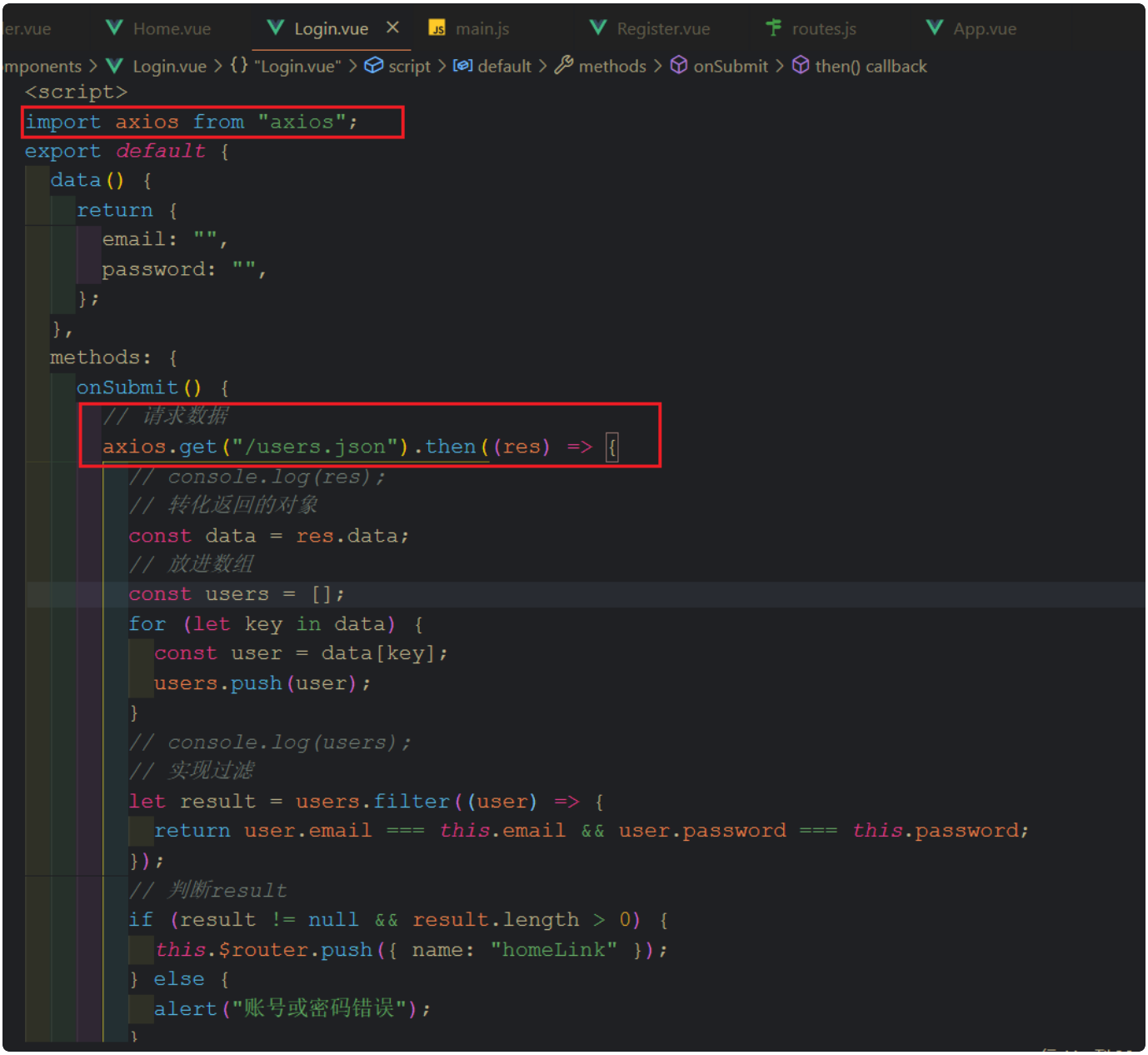
2. 注册页面



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


er.vue
Home.vue
Login.vue
main.js
Register.vue
routes.js
App.vue
mponents > Login.vue > {} "Login.vue" > script > default > methods > onSubmit > then() callback
<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

Vue | 复制代码

```vue
<template>
  <div>
    <div class="col-sm-12">
      <table class="table">
        <thead class="thead-default">
          <tr>
            <th>菜谱</th>
            <th>价格</th>
            <th>加入</th>
          </tr>
        </thead>
        <tbody v-for="item in getMenuItems" :key="item.name">
          <tr>
            <td>
              <strong>{{ item.name }}</strong>
            </td>
          </tr>
          <tr v-for="option in item.options" :key="option.size">
            <td>{{ option.size }}</td>
            <td>{{ option.price }}</td>
            <td><button class="btn btn-sm btn-outline-success">+</button>
</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</template>
<script>
export default {
  data() {
    return {
      getMenuItems: {
        1: {
          name: "爆香五花肉",
          description: "这是经典川菜",
          options: [
            {
              size: "中份",
              price: "33",
            },
            {
              size: "大份",
              price: "40",
            },
```

```
            ],
          },
          2: {
            name: "鱼香肉丝",
            description: "这是经典川菜",
            options: [
              {
                size: "中份",
                price: "33",
              },
              {
                size: "大份",
                price: "40",
              },
            ],
          },
          3: {
            name: "梅菜扣肉",
            description: "这是经典川菜",
            options: [
              {
                size: "中份",
                price: "33",
              },
              {
                size: "大份",
                price: "40",
              },
            ],
          },
        },
      };
    },
  };
  </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>
```

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

Vue | 复制代码

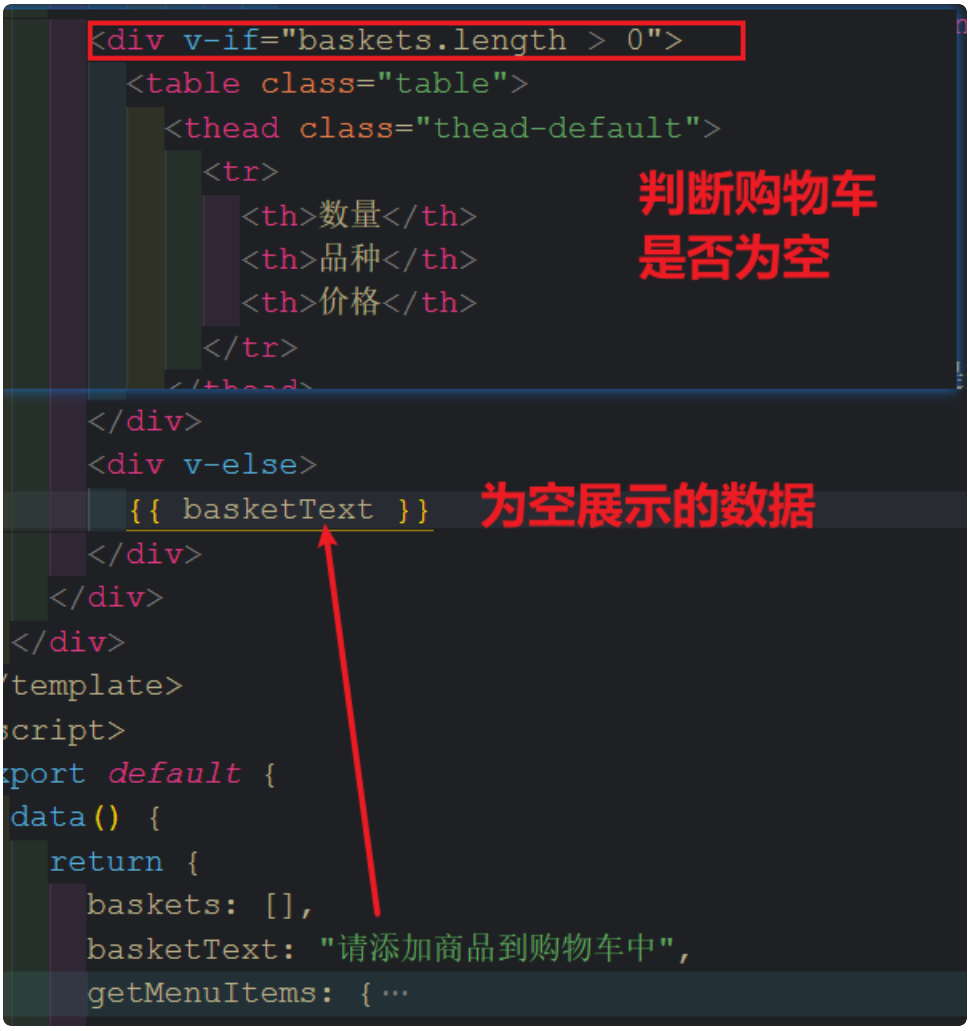
```vue
<template>
  <div class="row">
    <!-- 菜单 -->
    <div class="col-sm-12 col-md-8">
      <table class="table">
        <thead class="thead-default">
          <tr>
            <th>菜谱</th>
            <th>价格</th>
            <th>加入</th>
          </tr>
        </thead>
        <tbody v-for="item in getMenuItems" :key="item.name">
          <tr>
            <td>
              <strong>{{ item.name }}</strong>
            </td>
          </tr>
          <tr v-for="option in item.options" :key="option.size">
            <td>{{ option.size }}</td>
            <td>{{ option.price }}</td>
            <td>
              <button
                class="btn btn-sm btn-outline-success"
                @click="addToBasket(item, option)"
              >
                +
              </button>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
    <!-- 购物车-->
    <div class="col-md-4 col-sm-12">
      <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>
              <button class="btn btn-sm">-</button>
              <span>1</span>
              <button class="btn btn-sm">+</button>
            </td>
            <td>ssss</td>
            <td>ssss</td>
          </tr>
        </tbody>
      </table>
      <p>总价: </p>
      <button class="btn btn-success btn-block">提交</button>
    </div>
  </div>
</template>
<script>
export default {
  data() {
    return {
      baskets: [],
      getMenuItems: {
        1: {
          name: "爆香五花肉",
          description: "这是经典川菜",
          options: [
            {
              size: "中份",
              price: "33",
            },
            {
              size: "大份",
              price: "40",
            },
          ],
        },
        2: {
          name: "鱼香肉丝",
          description: "这是经典川菜",
          options: [
            {
              size: "中份",
              price: "33",
            },
            {
              size: "大份",
              price: "40",
            },
          ],
```

```
          },
          3: {
            name: "梅菜扣肉",
            description: "这是经典川菜",
            options: [
              {
                size: "中份",
                price: "33",
              },
              {
                size: "大份",
                price: "40",
              },
            ],
          },
        },
      };
    },
    methods: {
      addToBasket(item, option) {
        this.baskets.push({
          name: item.name,
          size: option.size,
          price: option.price,
          quantity: 1,
        });
      },
    },
  };
  </script>
```

展示购物车数据

1. 判断购物车是否为空



2. 展示购物车数据

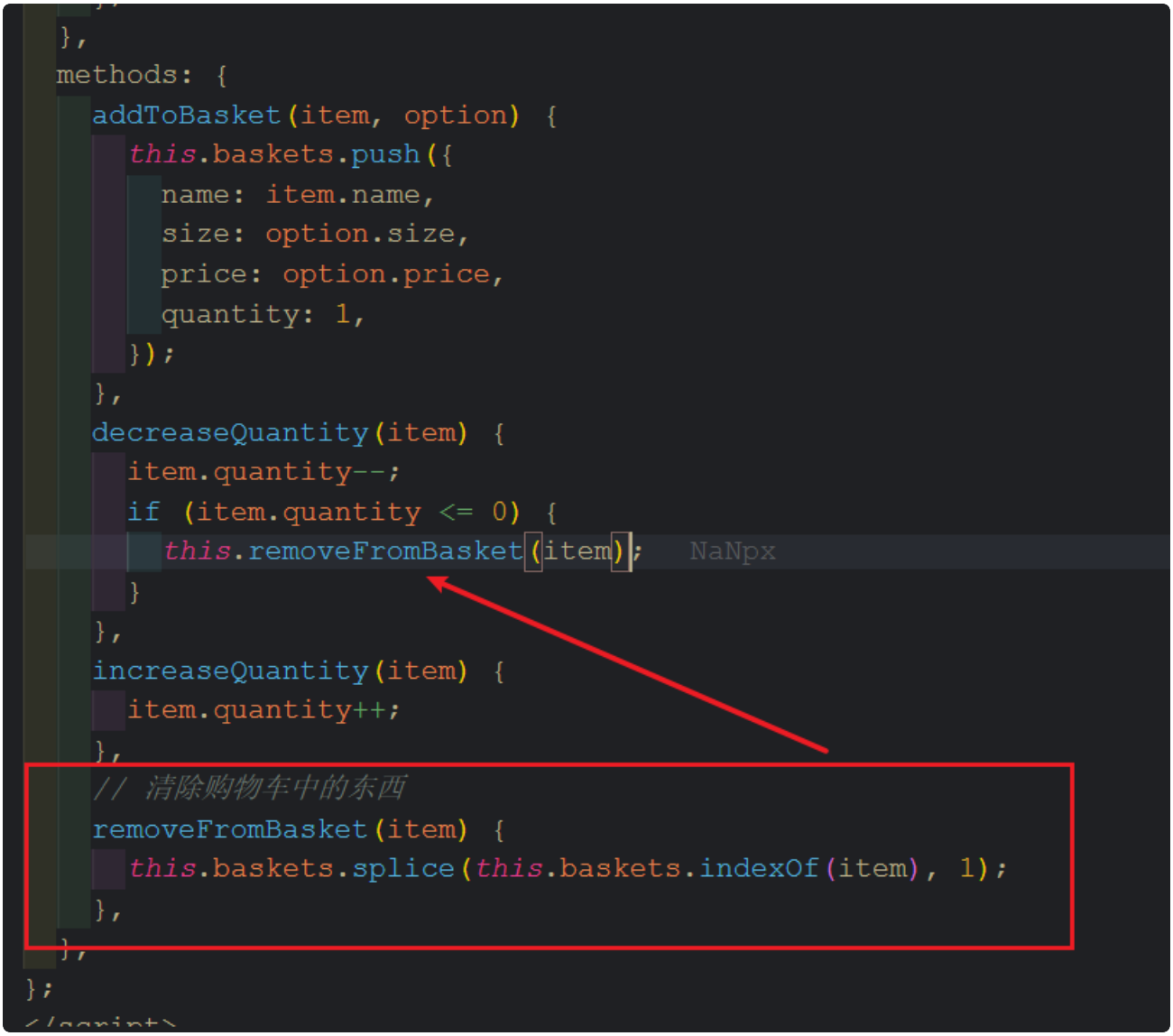
```
<!-- 购物车-->
<div class="col-md-4 col-sm-12">
  <div v-if="baskets.length > 0">
    <table class="table">
      <thead class="thead-default">
        <tr>
          <th>数量</th>
          <th>品种</th>
          <th>价格</th>
        </tr>
      </thead>
      <tbody v-for="item in baskets" :key="item.name">
        <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

```
<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);    NaNpx
      }
    },
    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

```
      <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>
xport 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

Vue | 复制代码

```vue
<template>
  <div class="row">
    <div class="col-sm-12 col-md-8">
      <!-- 新菜谱 -->

    </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>
          <td>爆香五花肉</td>
          <td>
            <button class="btn btn-outline-danger btn=sm">&times;
</button>
          </td>
        </tbody>
      </table>
    </div>
  </div>
</template>

<script>
export default {};
</script>
```

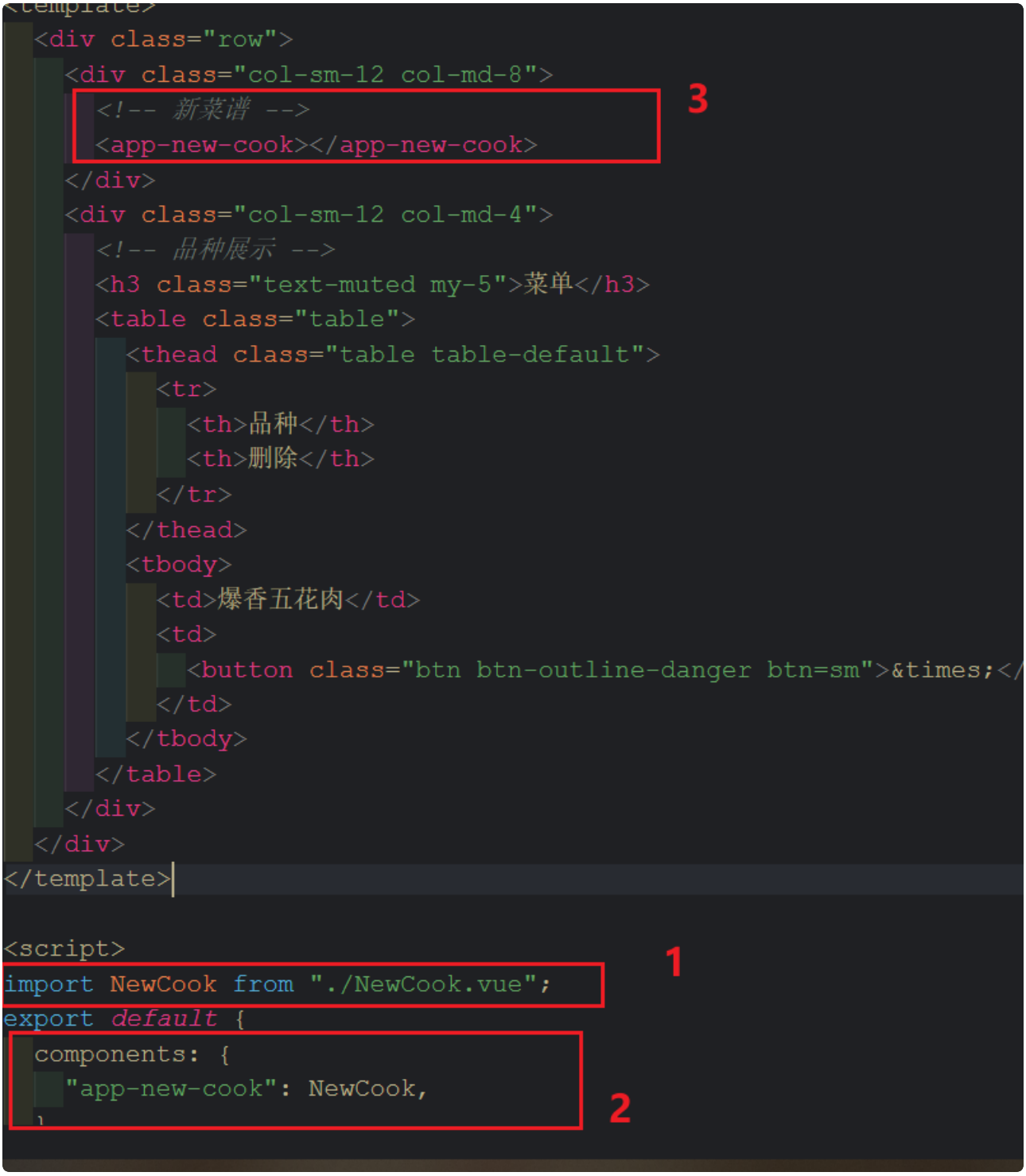
2. 新建newCook.vue组件

Vue | 复制代码

```vue
<template>
  <form>
    <h3 class="text-muted my-5">添加新的菜单</h3>
    <div class="form-group row">
      <label class="col-sm-1">品种</label>
      <div class="col-sm-11">
        <input type="text" class="form-control" v-model="newCook.name" />
      </div>
    </div>
    <div class="form-group row">
      <label class="col-sm-1">描述</label>
      <div class="col-sm-11">
        <textarea
          rows="5"
          class="form-control"
          v-model="newCook.description"
        ></textarea>
      </div>
    </div>
    <p><strong>选项1</strong></p>
    <div class="form-group row">
      <label class="col-sm-1">尺寸</label>
      <div class="col-sm-11">
        <input type="text" class="form-control" v-model="newCook.size1"
/>
      </div>
    </div>
    <div class="form-group row">
      <label class="col-sm-1">价格</label>
      <div class="col-sm-11">
        <input type="text" class="form-control" v-model="newCook.price1"
/>
      </div>
    </div>
    <p><strong>选项2</strong></p>
    <div class="form-group row">
      <label class="col-sm-1">尺寸</label>
      <div class="col-sm-11">
        <input type="text" class="form-control" v-model="newCook.size2"
/>
      </div>
    </div>
    <div class="form-group row">
      <label class="col-sm-1">价格</label>
      <div class="col-sm-11">
```

```
          <input type="text" class="form-control" v-model="newCook.price2" />
        </div>
      </div>
      <div class="form-group row">
        <button class="btn btn-success btn-block" type="button">添加</button>
      </div>
    </form>
  </template>

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

3. 在admin.vue中



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);
    });
  },
```

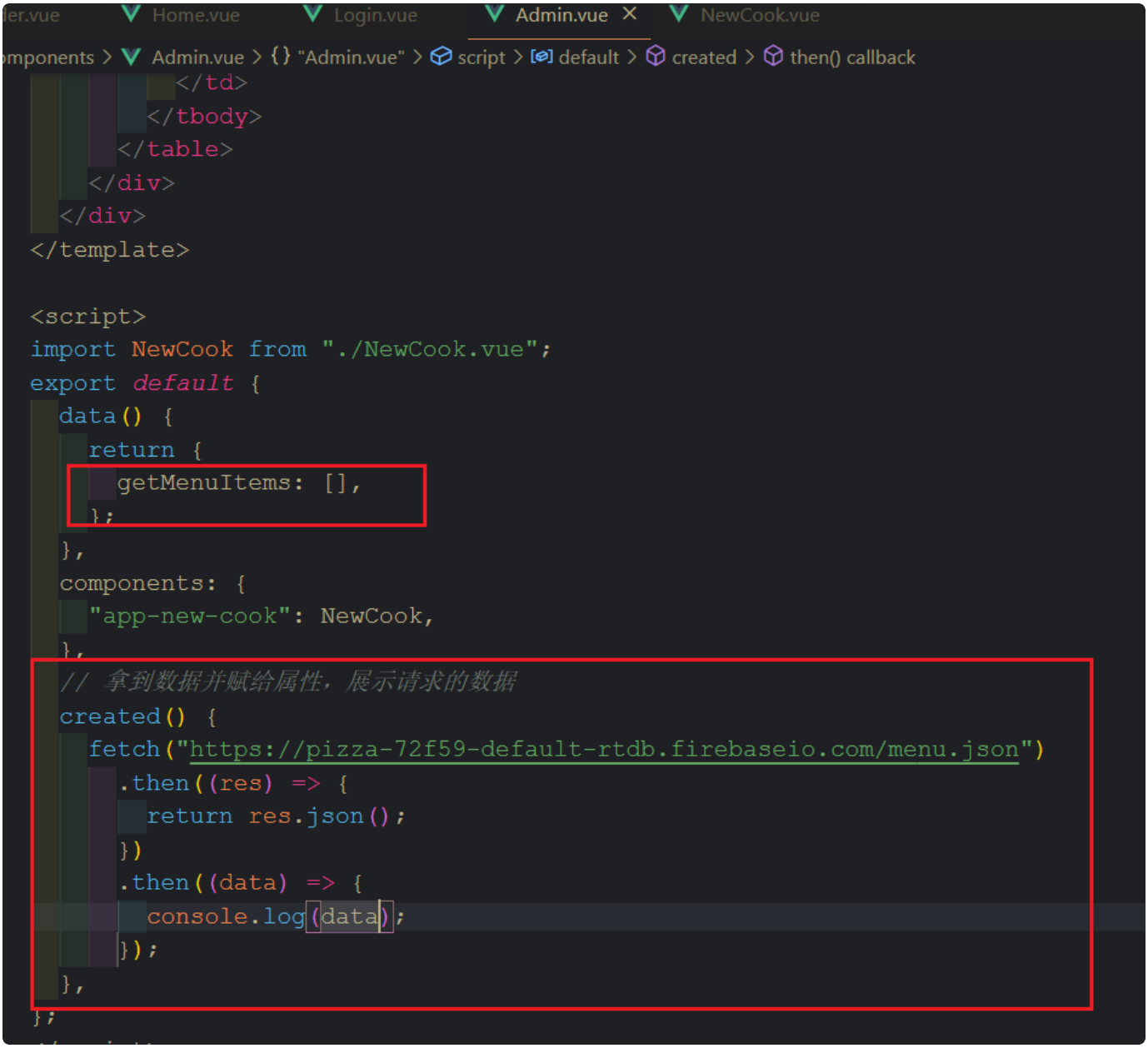
3. 处理数据

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-rtdb.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


Home.vue
Login.vue
Admin.vue
NewCook.vue
components > Admin.vue > {} "Admin.vue" > script > default > created > then() callback
</td>
</tbody>
</table>

</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-rtdb.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-rtdb.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

```
            <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>
```

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-rtdb.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-rtdb.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-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;
    });
  },
```

main.js中配置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/';

// 配置Vue原型(在任何组件中都可以正常使用axios)
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

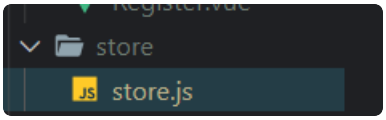
将数据存在vuex中，方便任何组件使用

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

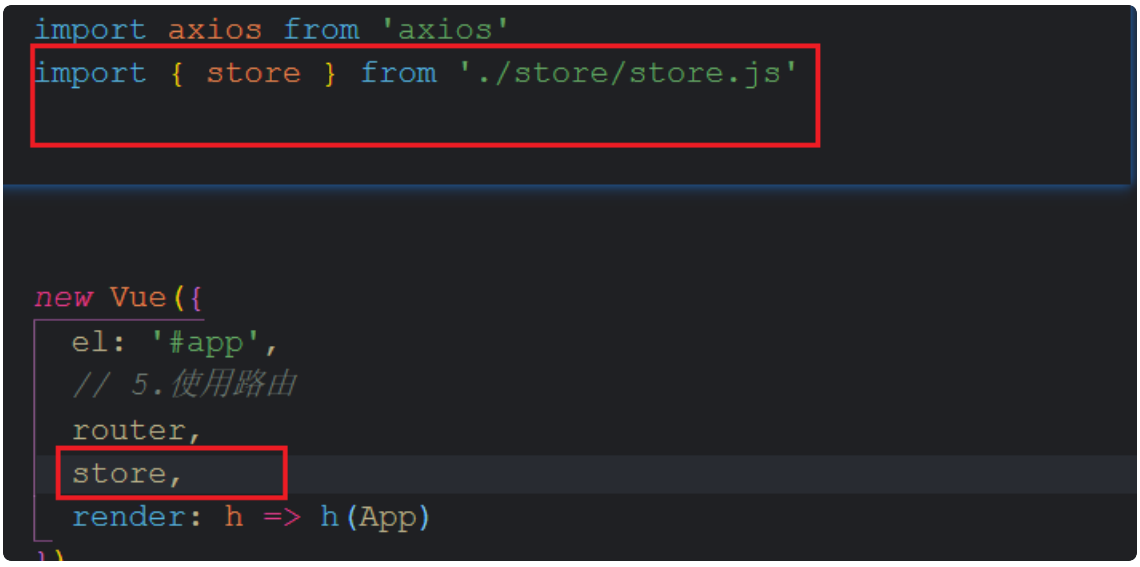


store.js内容：

Vue 复制代码

```vue
import Vue from "vue";
import Vuex from 'vuex';

Vue.use(Vuex);

export const store = new Vuex.Store({
    state: {
        // 存储数据
    },
    getters: {
        // 获取数据的方法,获取属性的状态 state
    },
    mutations: {
        // 改变属性的状态
    },
    actions: {
        // 应用mutations
    }
})

```

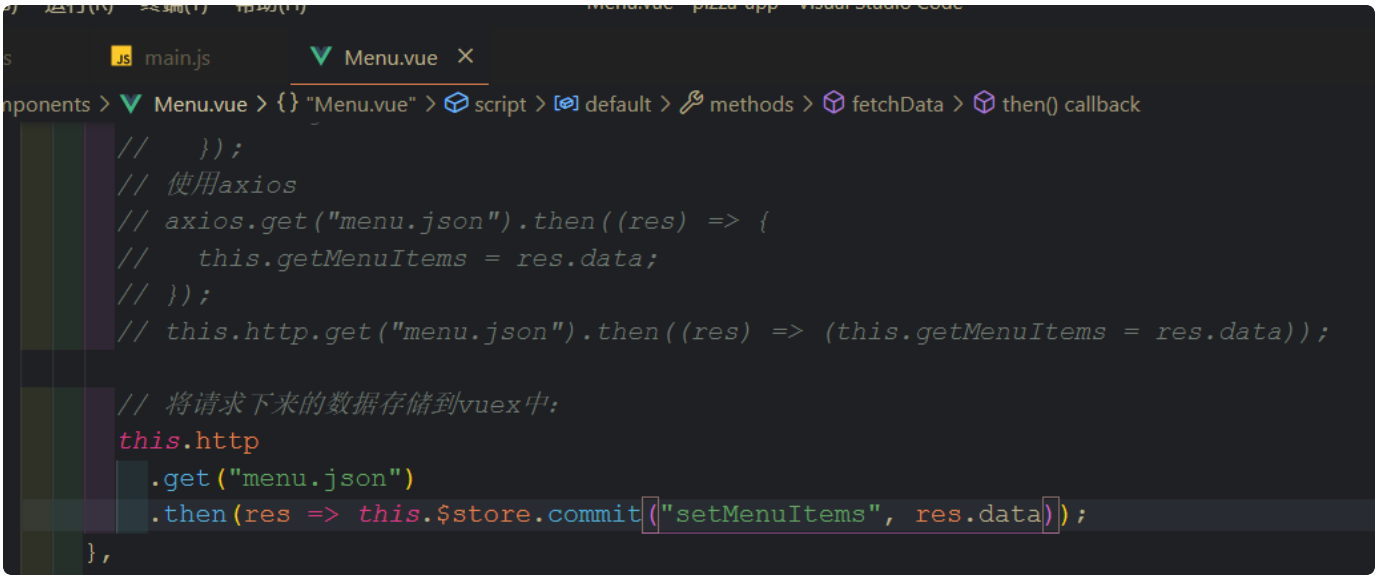
3. 在main.js中配置store



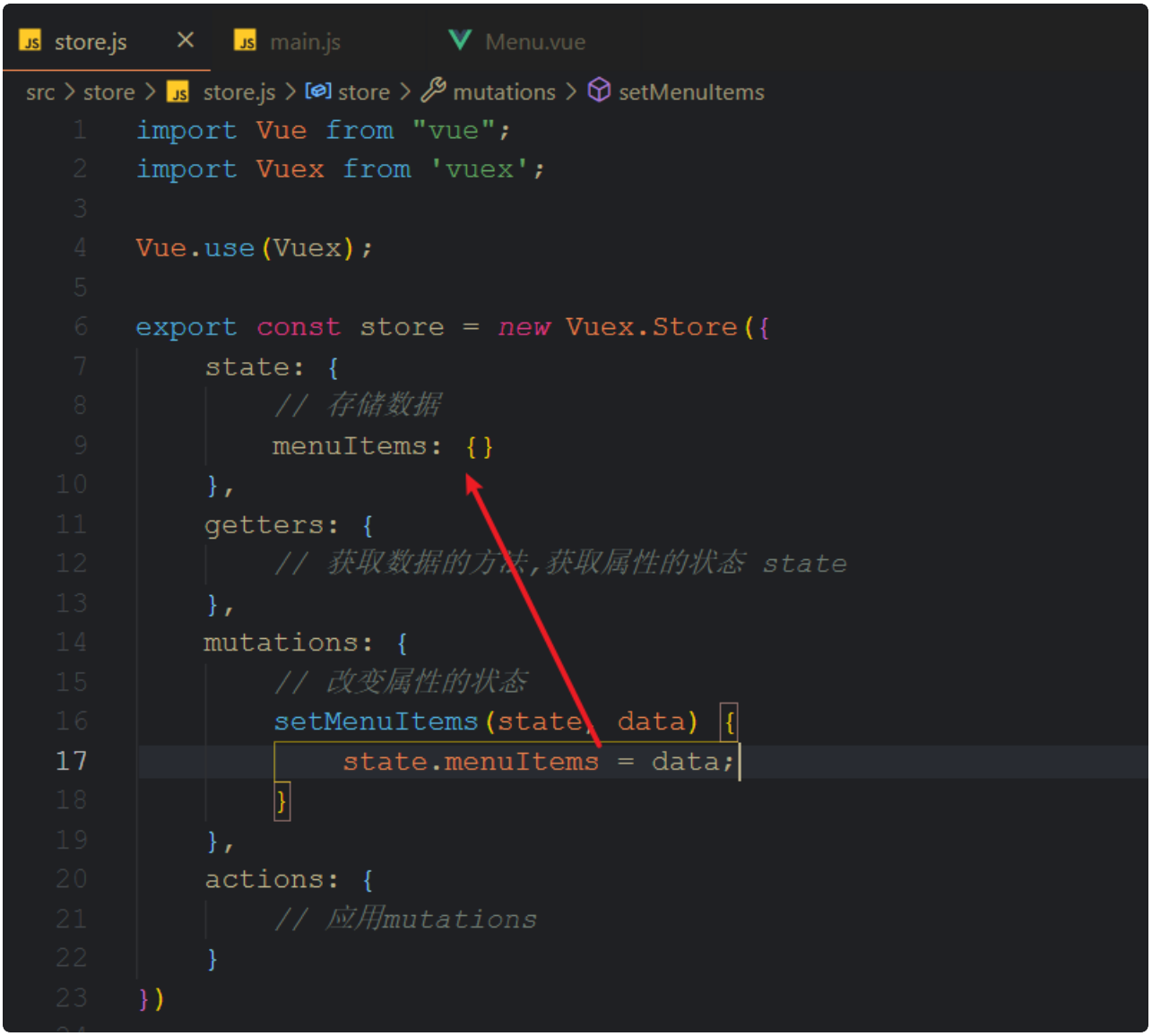
4. 在store.js中预设属性

```
xport 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



6. 处理接收到的数据并存在menuItems中：



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

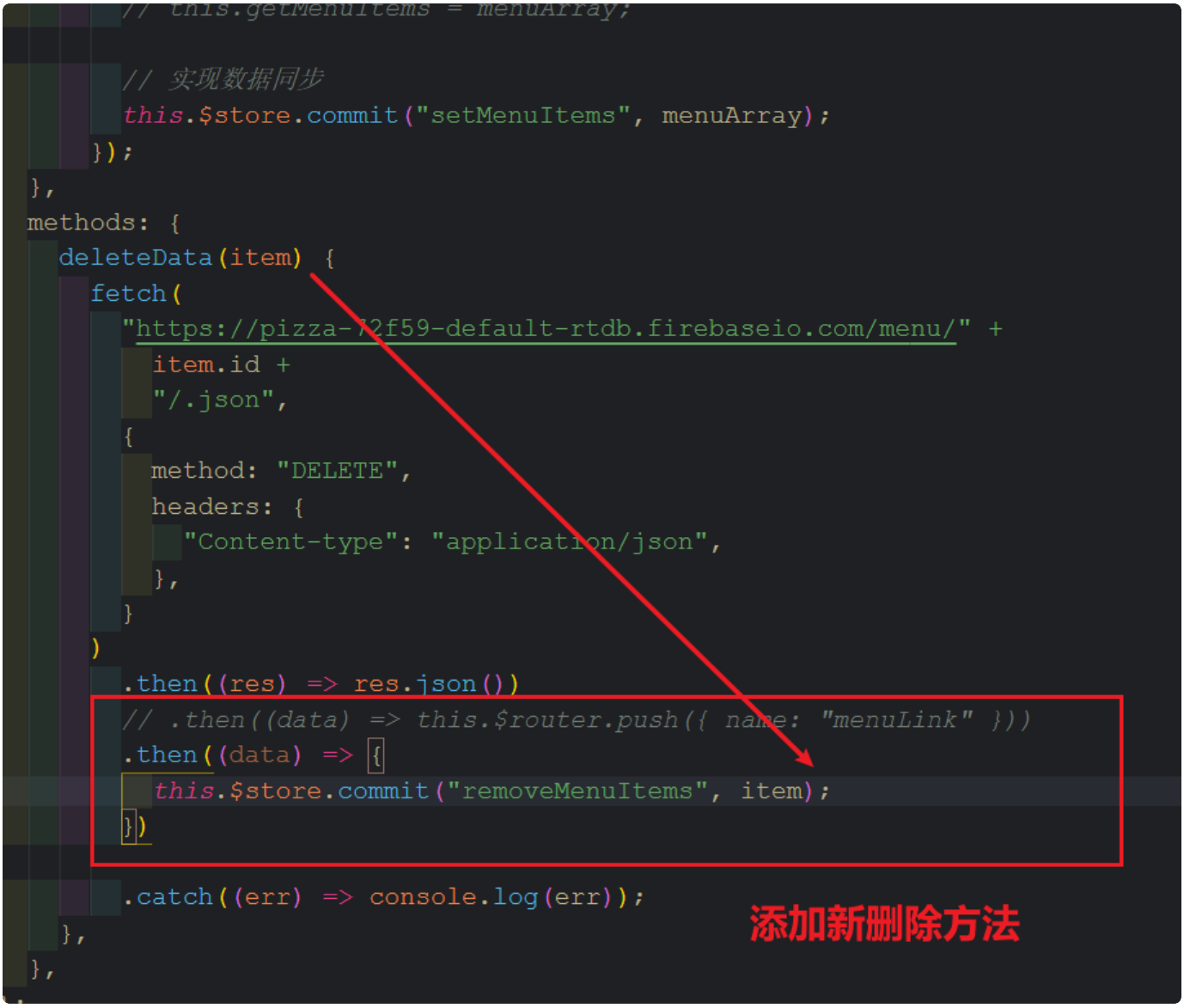
1. 修改admin页面

```
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-rtdb.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.$store.commit("setMenuItems", menuArray);
});
},
methods: {
deleteData(item) {
fetch(
"https://pizza-f2f59-default-rtdb.firebaseio.com/menu/" +
item.id +
"/.json",
{
method: "DELETE",
headers: {
"Content-type": "application/json",
},
}
)
.then((res) => res.json())
// .then((data) => this.$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
        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
        pushToMenuItems(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

```
> 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

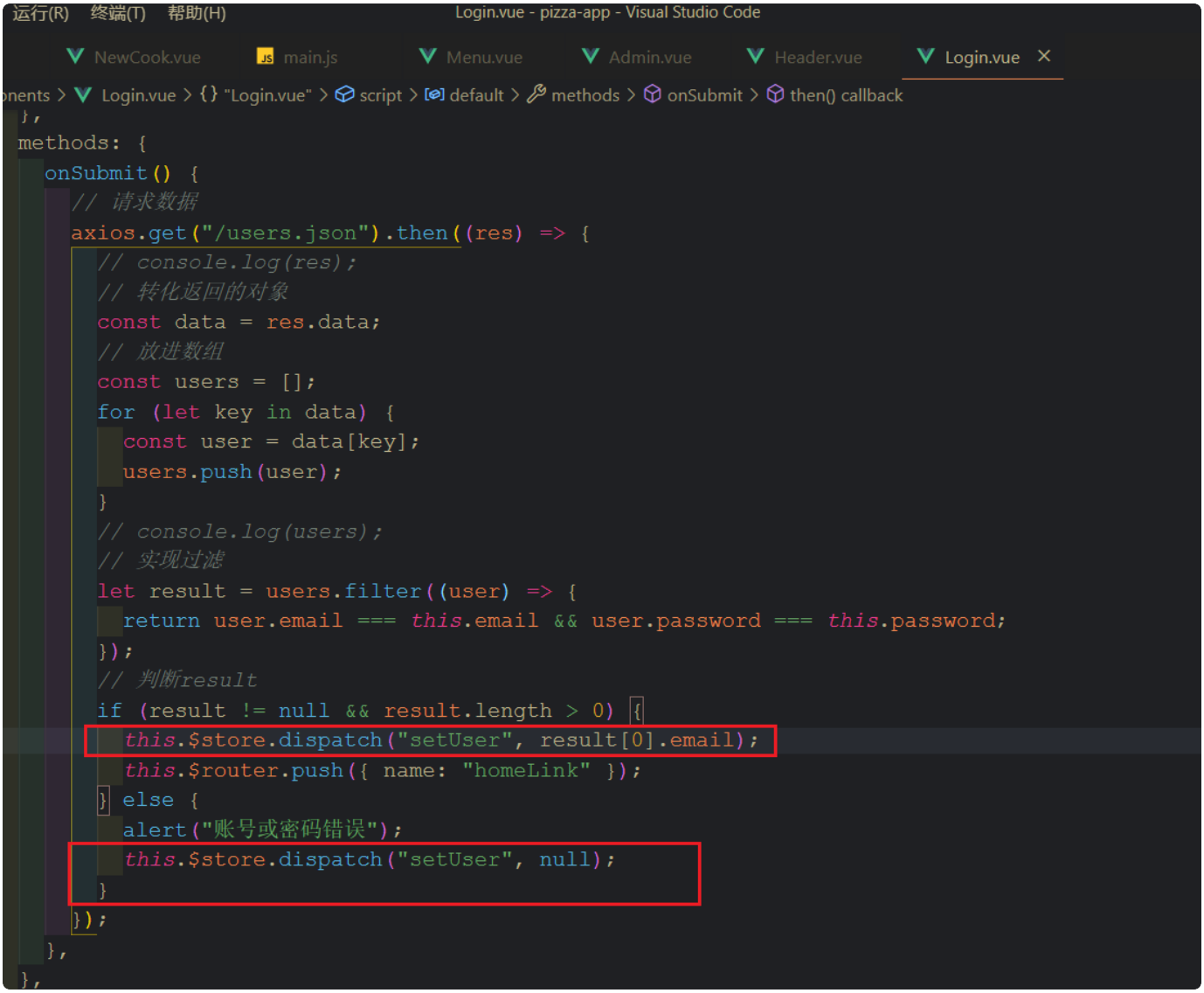
```
Vue.use(Vuex);

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

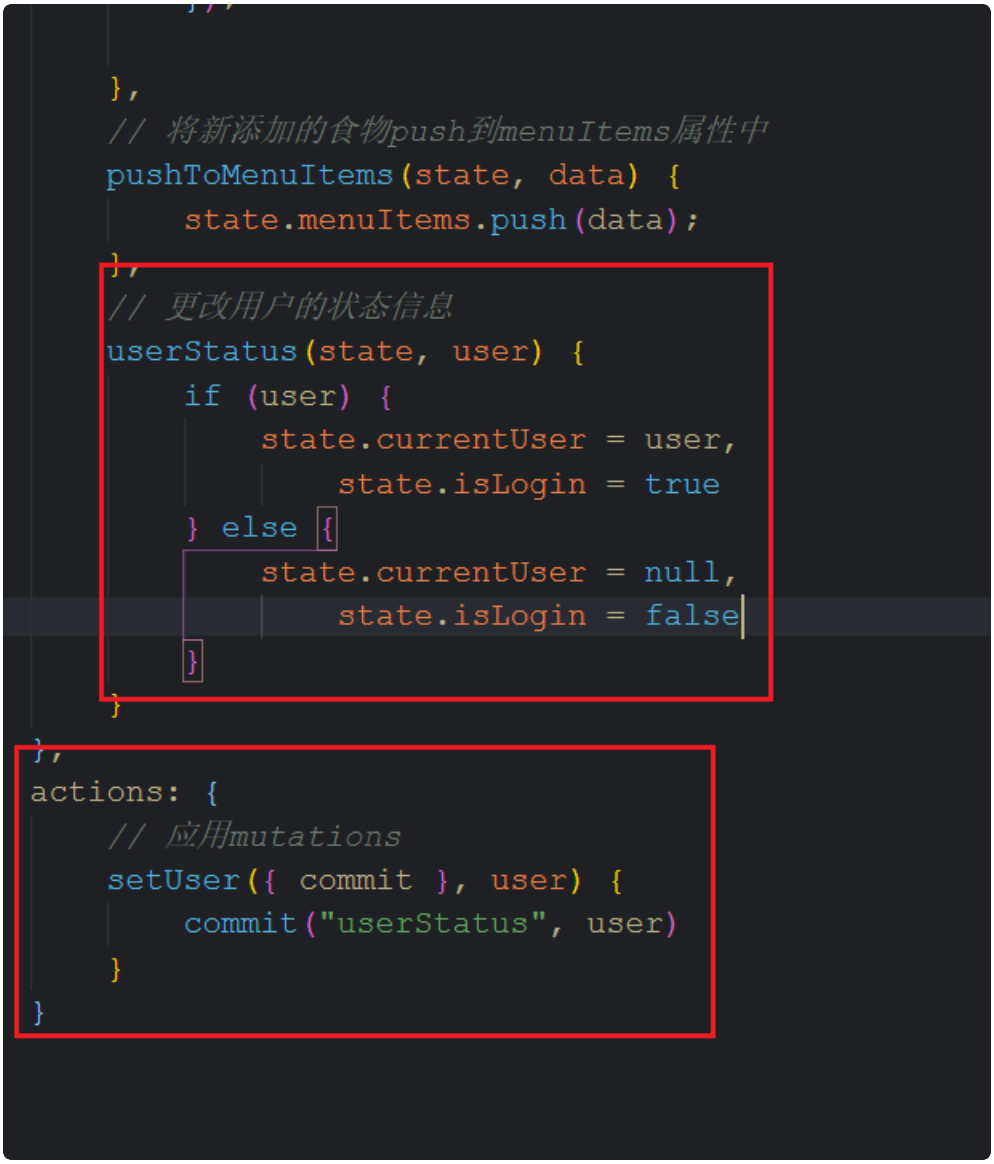
5. 设置登录

```
    </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>
xport 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
NewCook.vue
main.js
Menu.vue
Admin.vue
Header.vue
Login.vue
onents > 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



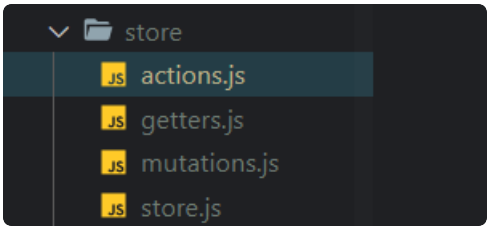
8. 组件内守卫

一旦点击退出就回到登录页面，并清空登录信息

```
NewCook.vue  main.js  Menu.vue  Admin.vue  Header.vue  Login.vue
components > Login.vue > {} "Login.vue" > script > default > beforeRouteEnter > 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

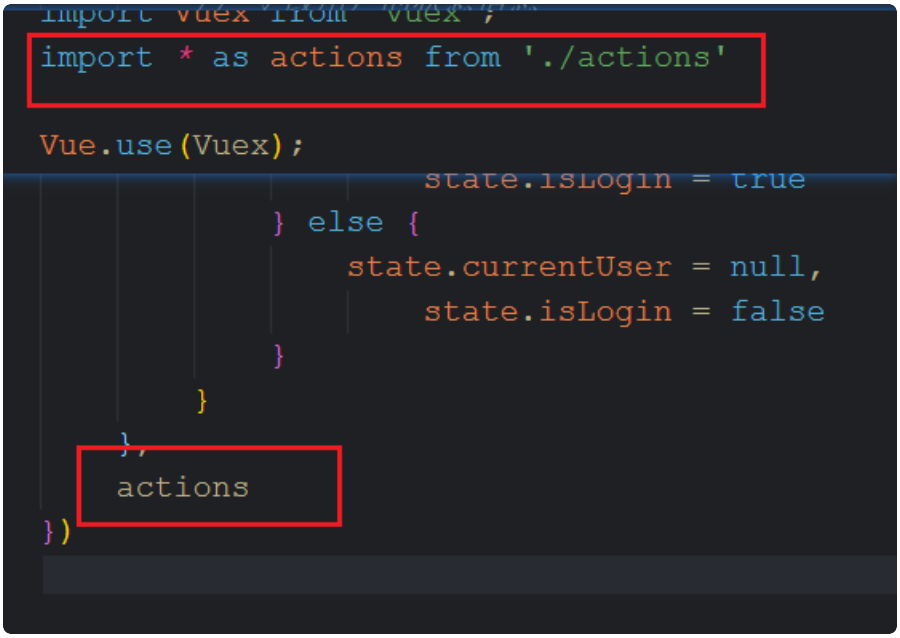
1. 新建三个文件



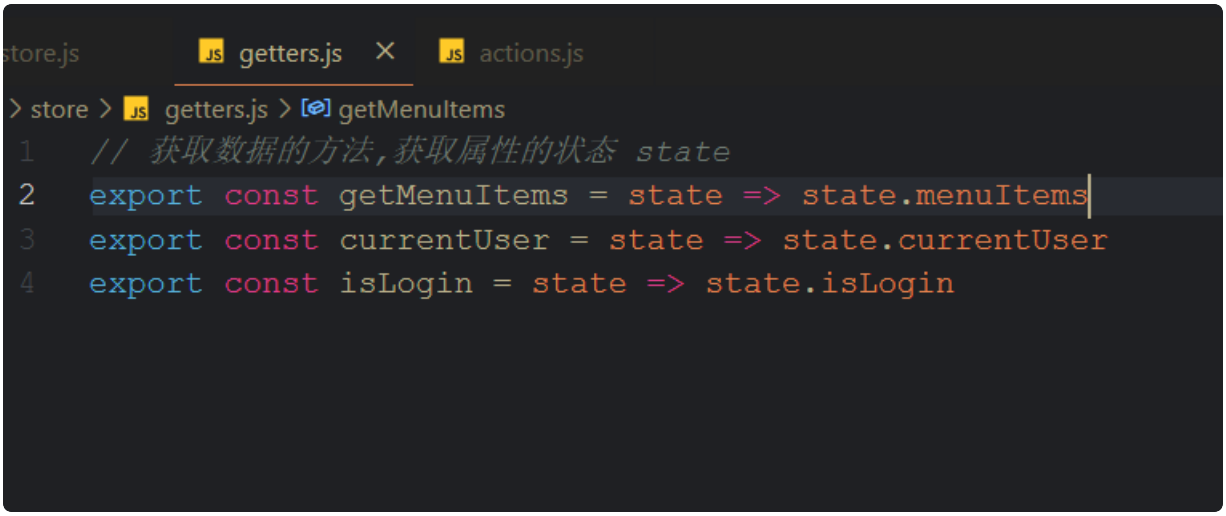
2. 移动actions内容

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

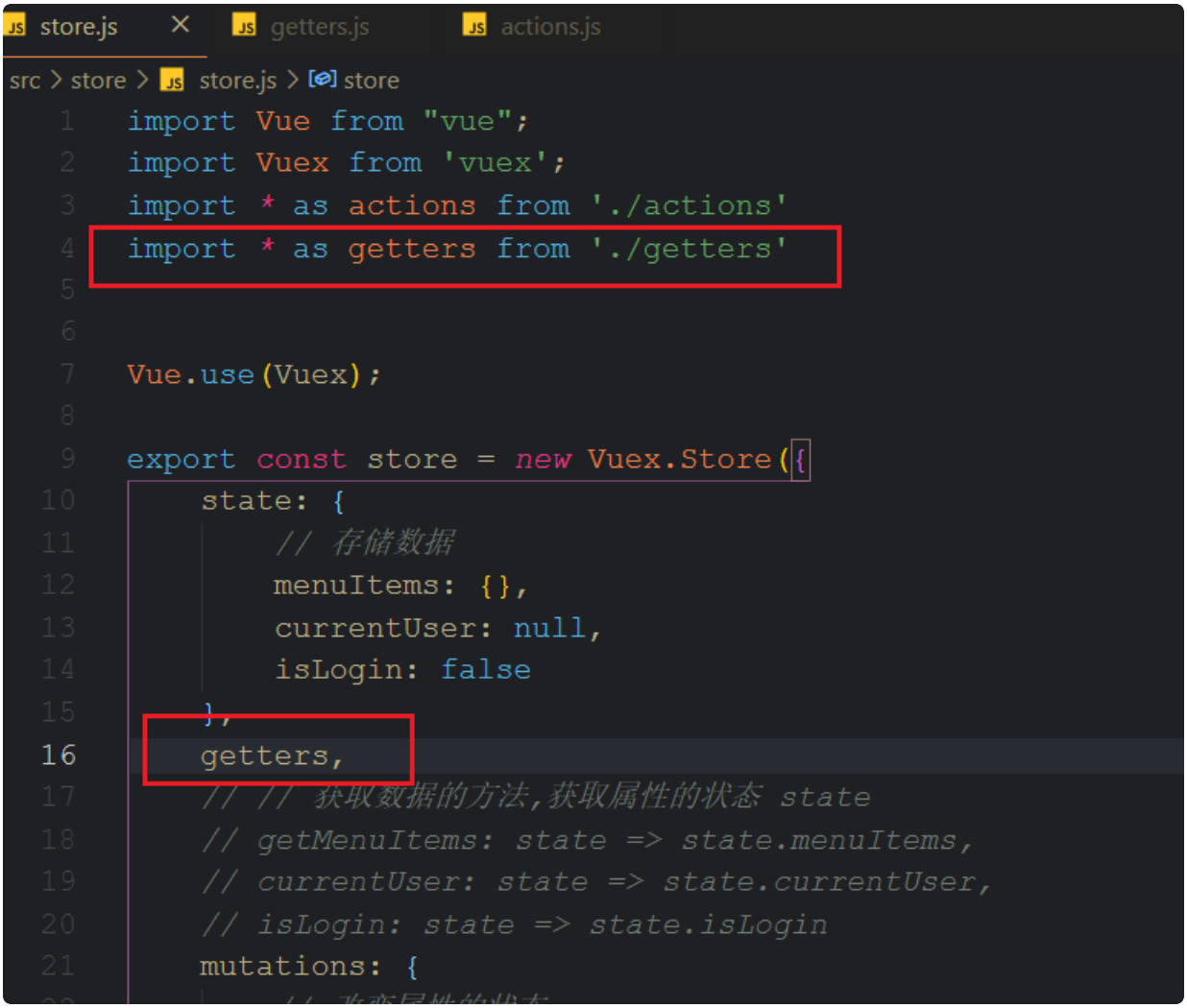
3. 引入store.js



4. getters.js



5. 引入store.js


store.js
getters.js
actions.js
src > store > store.js > store
import Vue from "vue";
import Vuex from 'vuex';
import * as actions from './actions'
import * as getters from './getters'
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: {

6. mutations:


store.js
mutations.js
getters.js
actions.js
src > store > mutations.js > pushToMenuItems
// 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
}
}

7. store

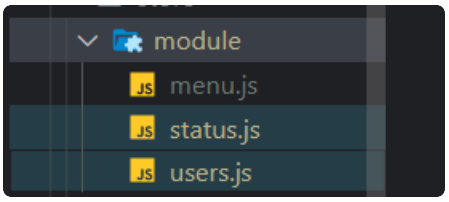
store.js × mutations.js getters.js actions.js

src > store > store.js > ...

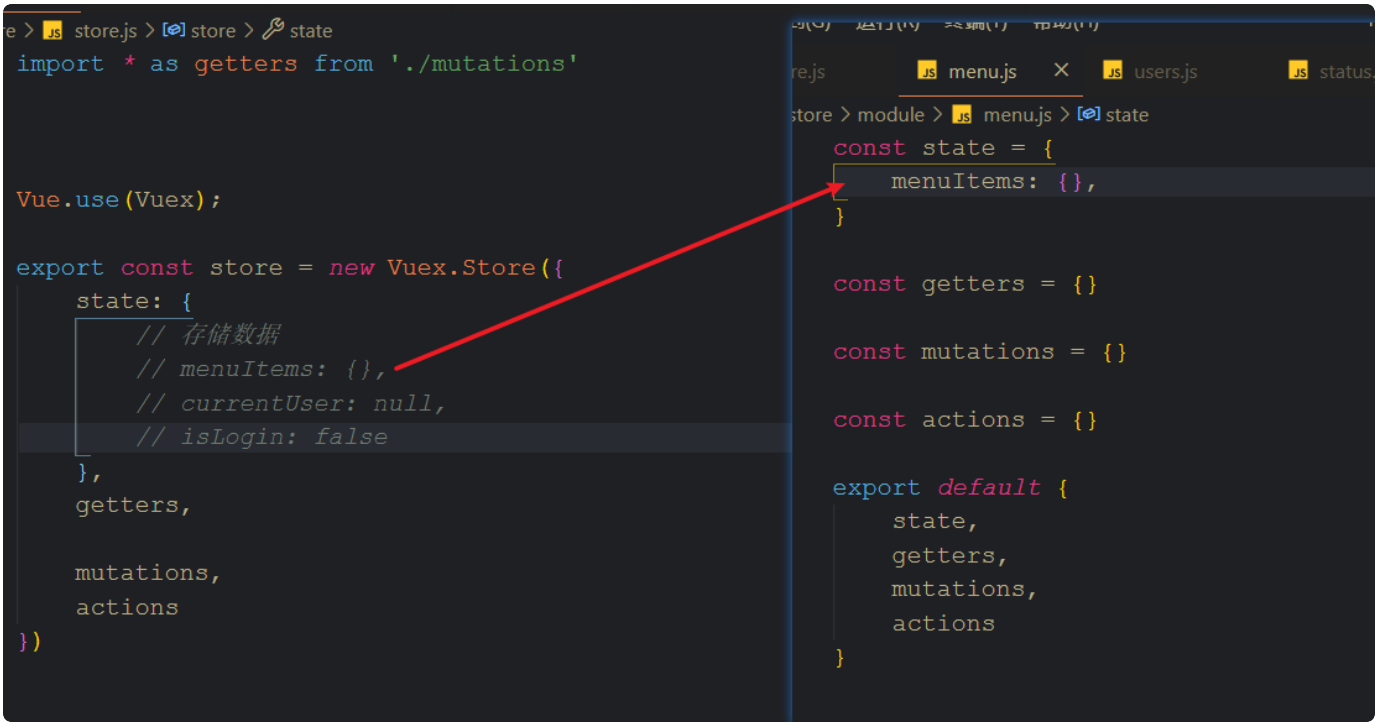
```
import Vue from "vue";
import Vuex from 'vuex';
import * as actions from './actions'
import * as getters from './getters'
import * as mutations from './mutations'



Vue.use(Vuex);

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

    mutations,
    actions
})
```

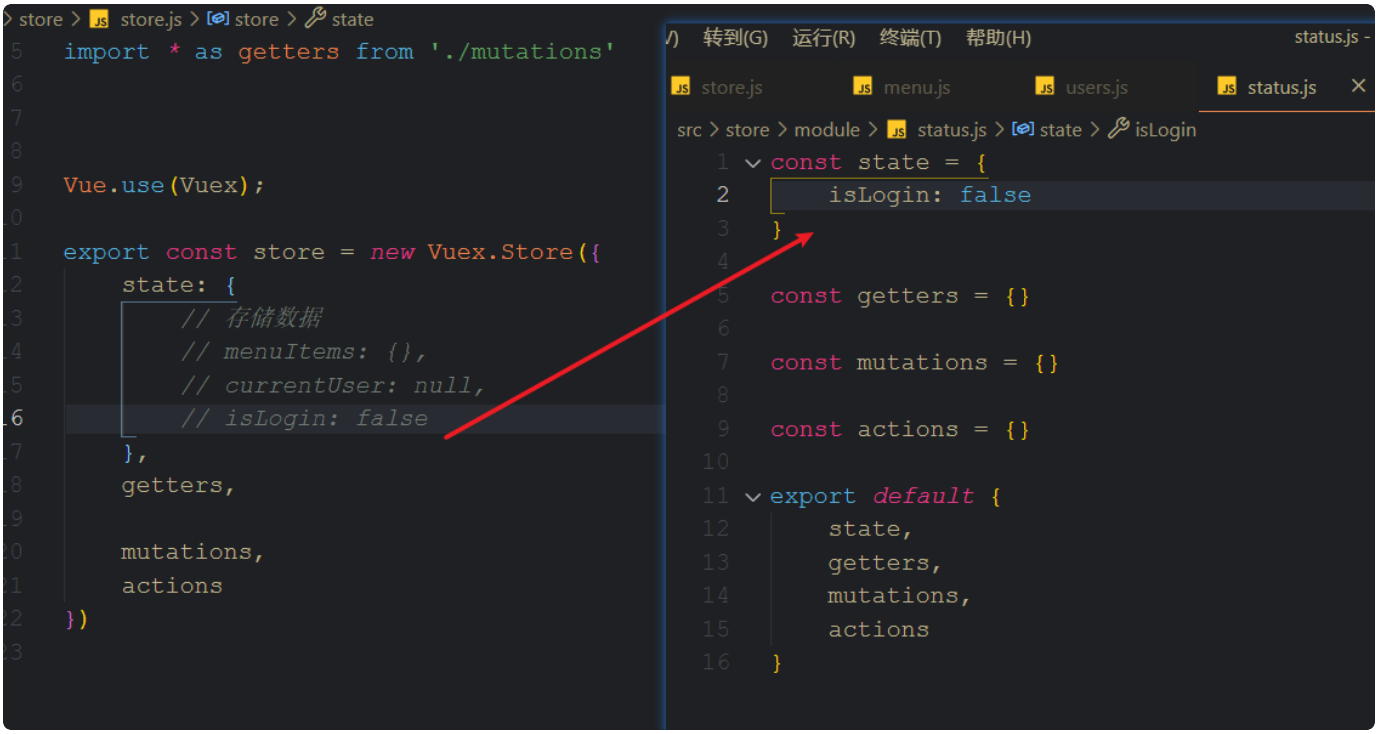
使用Module实现模块化

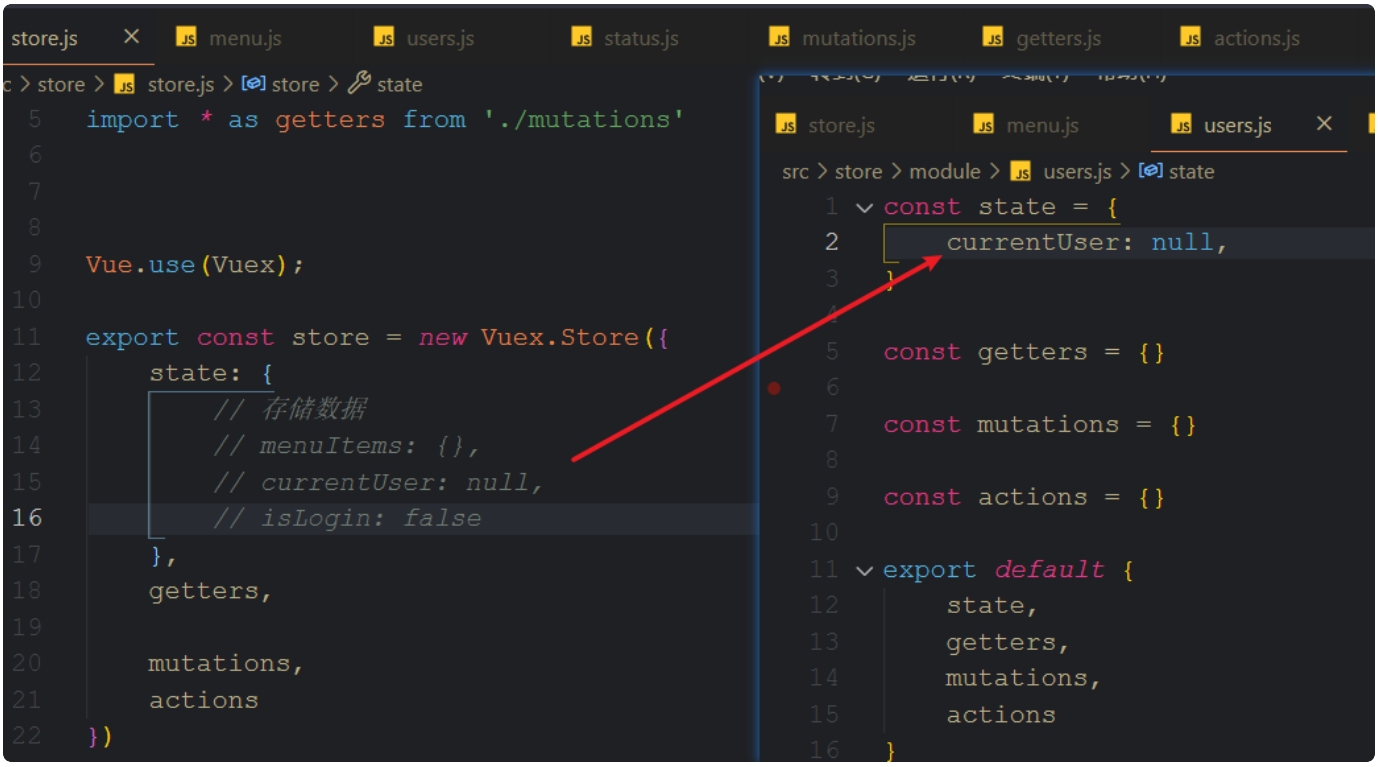
1. 新建文件



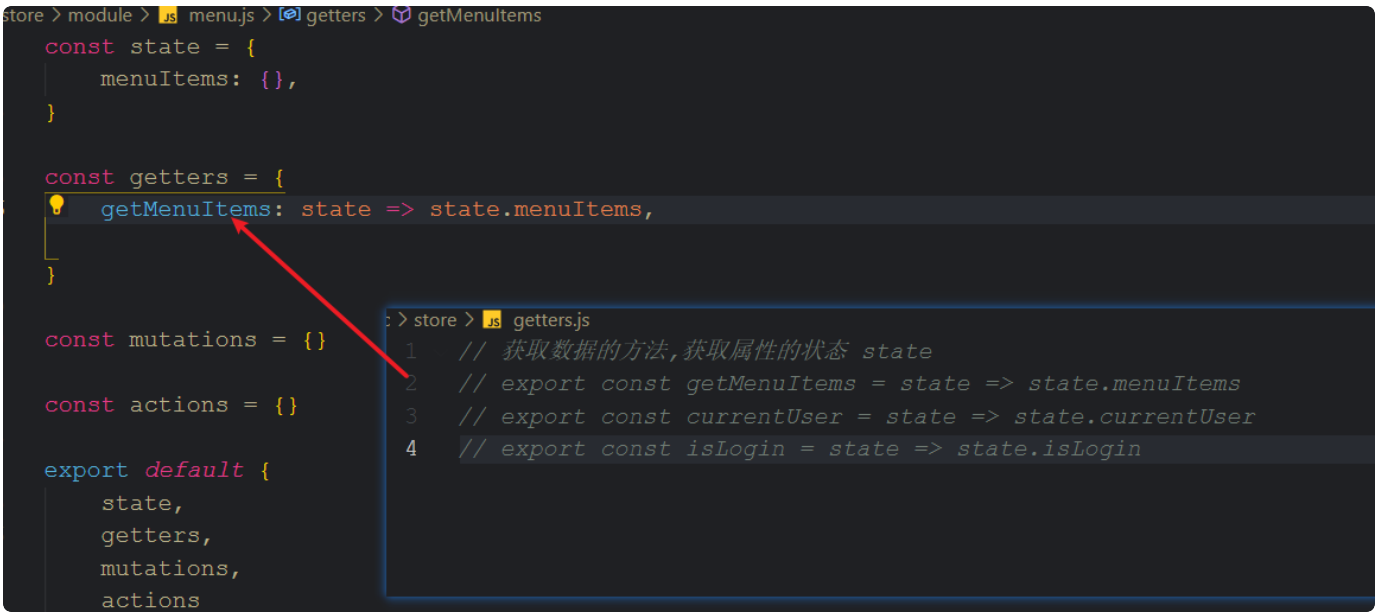
2. 抽离

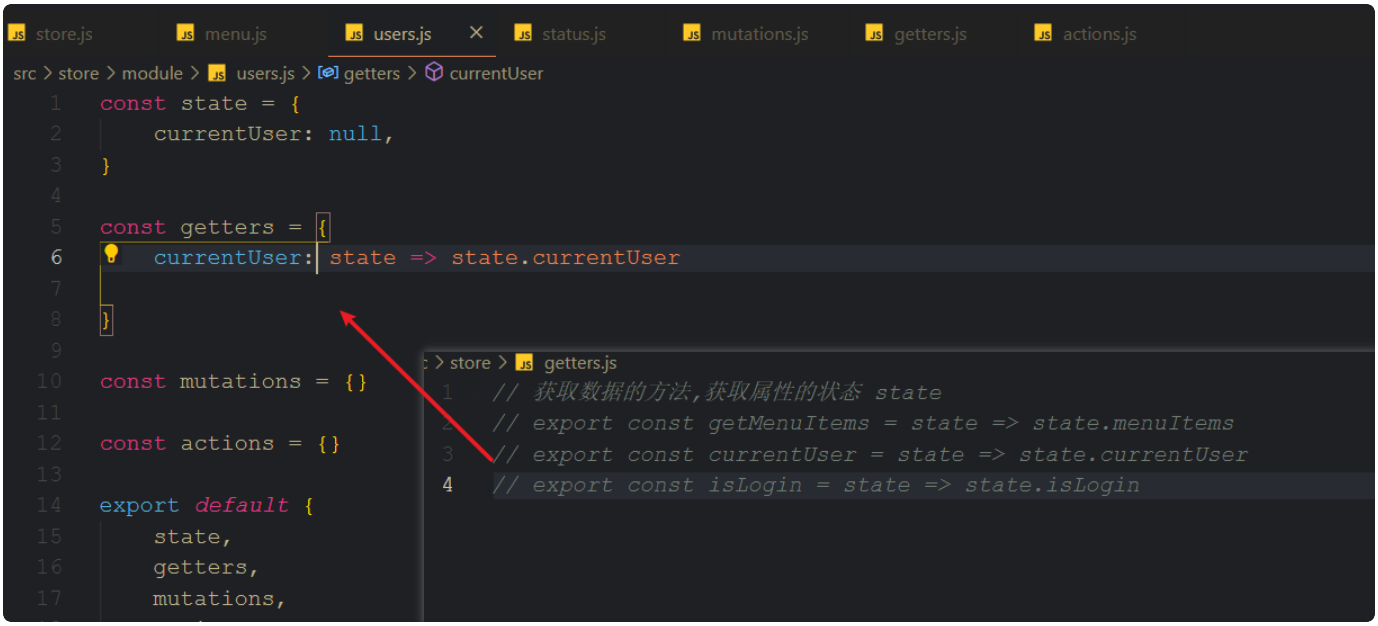

store.js > store > state
import * as getters from './mutations'
Vue.use(Vuex);
export const store = new Vuex.Store({
state: {
// 存储数据
// menuItems: {},
// currentUser: null,
// isLogin: false
},
getters,
mutations,
actions
})
menu.js
users.js
store > module > menu.js > state
const state = {
menuItems: {},
}
const getters = {}
const mutations = {}
const actions = {}
export default {
state,
getters,
mutations,
actions
}

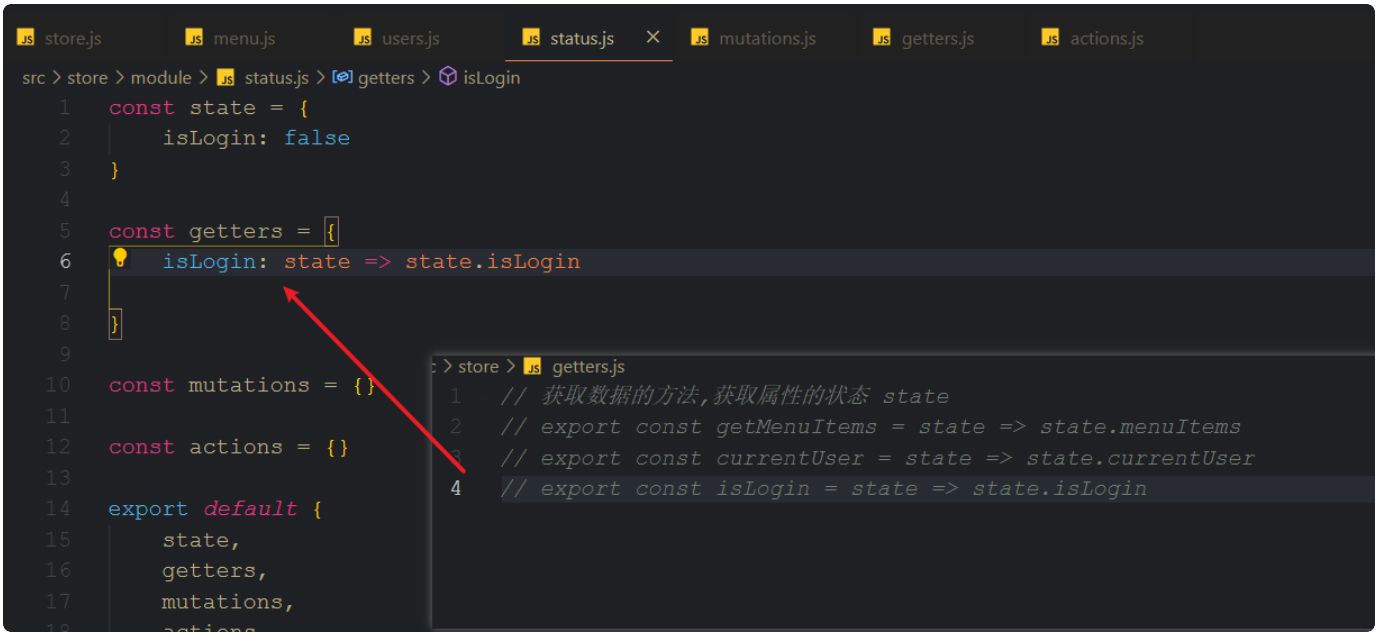

store > store.js > store > state
import * as getters from './mutations'
Vue.use(Vuex);
export const store = new Vuex.Store({
state: {
// 存储数据
// menuItems: {},
// currentUser: null,
// isLogin: false
},
getters,
mutations,
actions
})
转到(G) 运行(R) 终端(T) 帮助(H)
status.js -
store.js
menu.js
users.js
status.js
src > store > module > status.js > state > isLogin
const state = {
isLogin: false
}
const getters = {}
const mutations = {}
const actions = {}
export default {
state,
getters,
mutations,
actions
}


store.js
menu.js
users.js
status.js
mutations.js
getters.js
actions.js
store > store.js > store > state
import * as getters from './mutations'
Vue.use(Vuex);
export const store = new Vuex.Store({
state: {
// 存储数据
// menuItems: {},
// currentUser: null,
// isLogin: false
},
getters,
mutations,
actions
})
store.js
menu.js
users.js
src > store > module > users.js > state
const state = {
currentUser: null,
}
const getters = {}
const mutations = {}
const actions = {}
export default {
state,
getters,
mutations,
actions
}

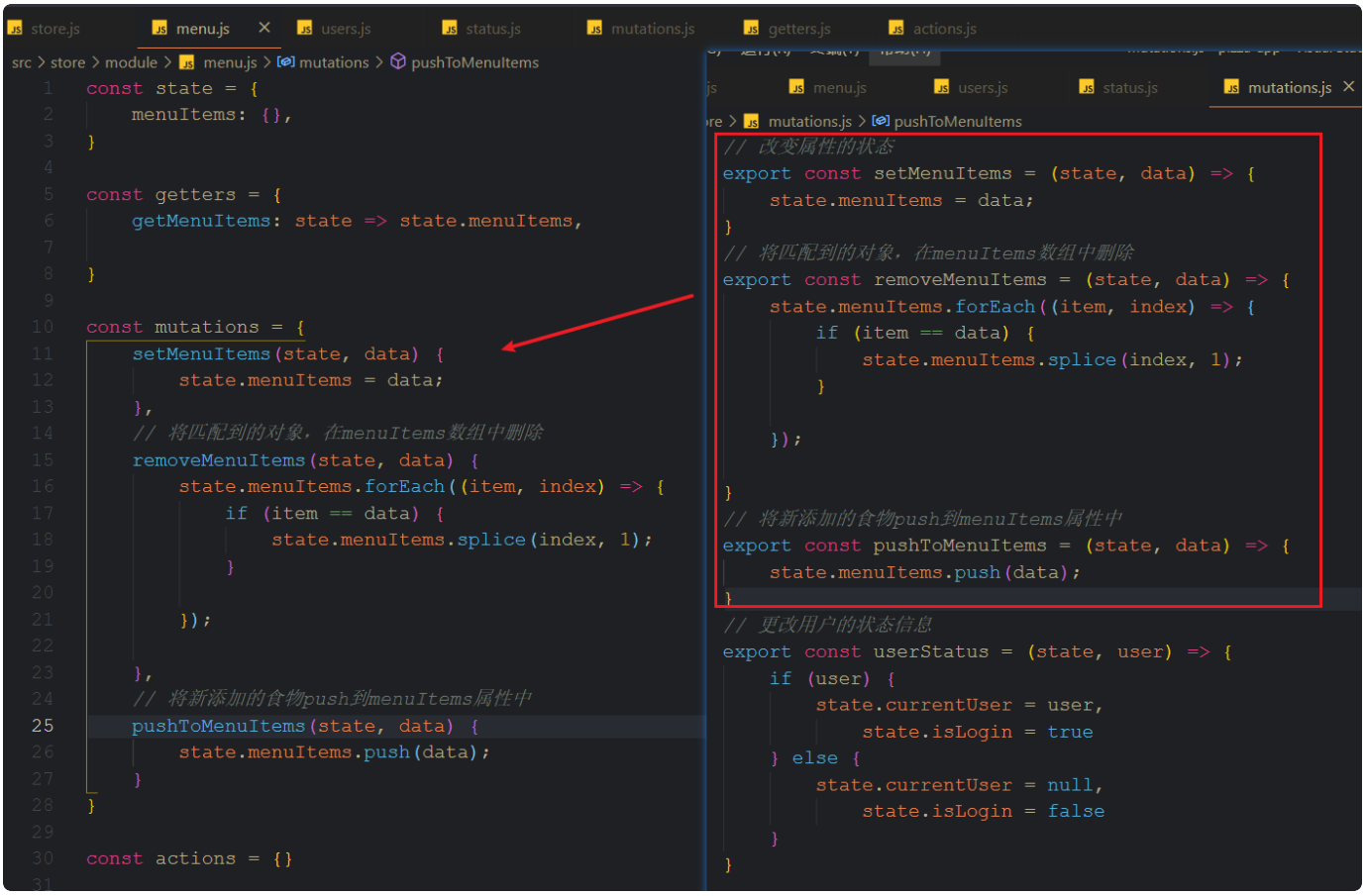
3. 抽离getter

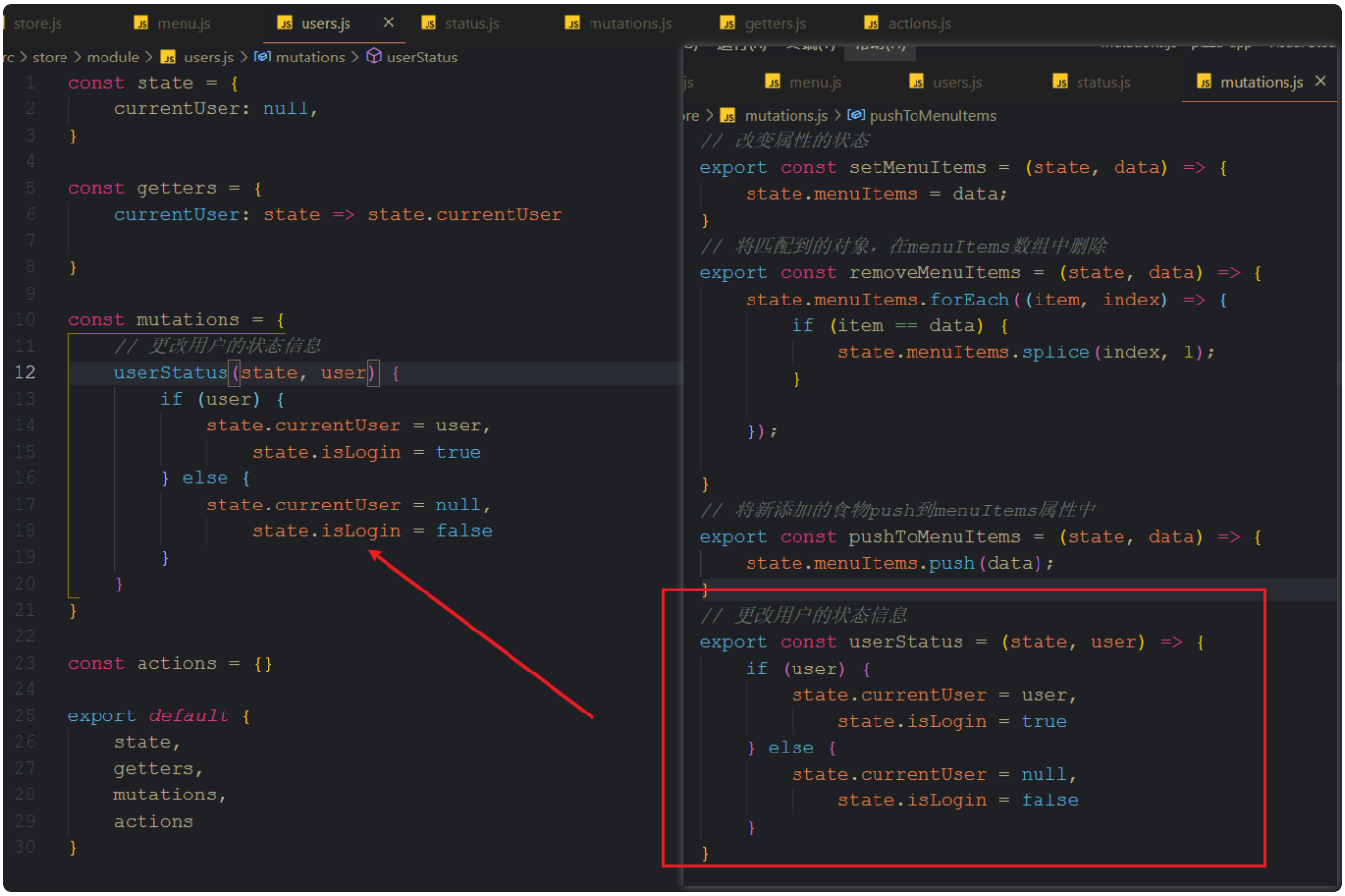

store > module > menu.js > getters > getMenuItems
const state = {
menuItems: {},
}
const getters = {
getMenuItems: state => state.menuItems,
}
const mutations = {}
const actions = {}
export default {
state,
getters,
mutations,
actions
store > getters.js
// 获取数据的方法,获取属性的状态 state
// export const getMenuItems = state => state.menuItems
// export const currentUser = state => state.currentUser
// export const isLogin = state => state.isLogin



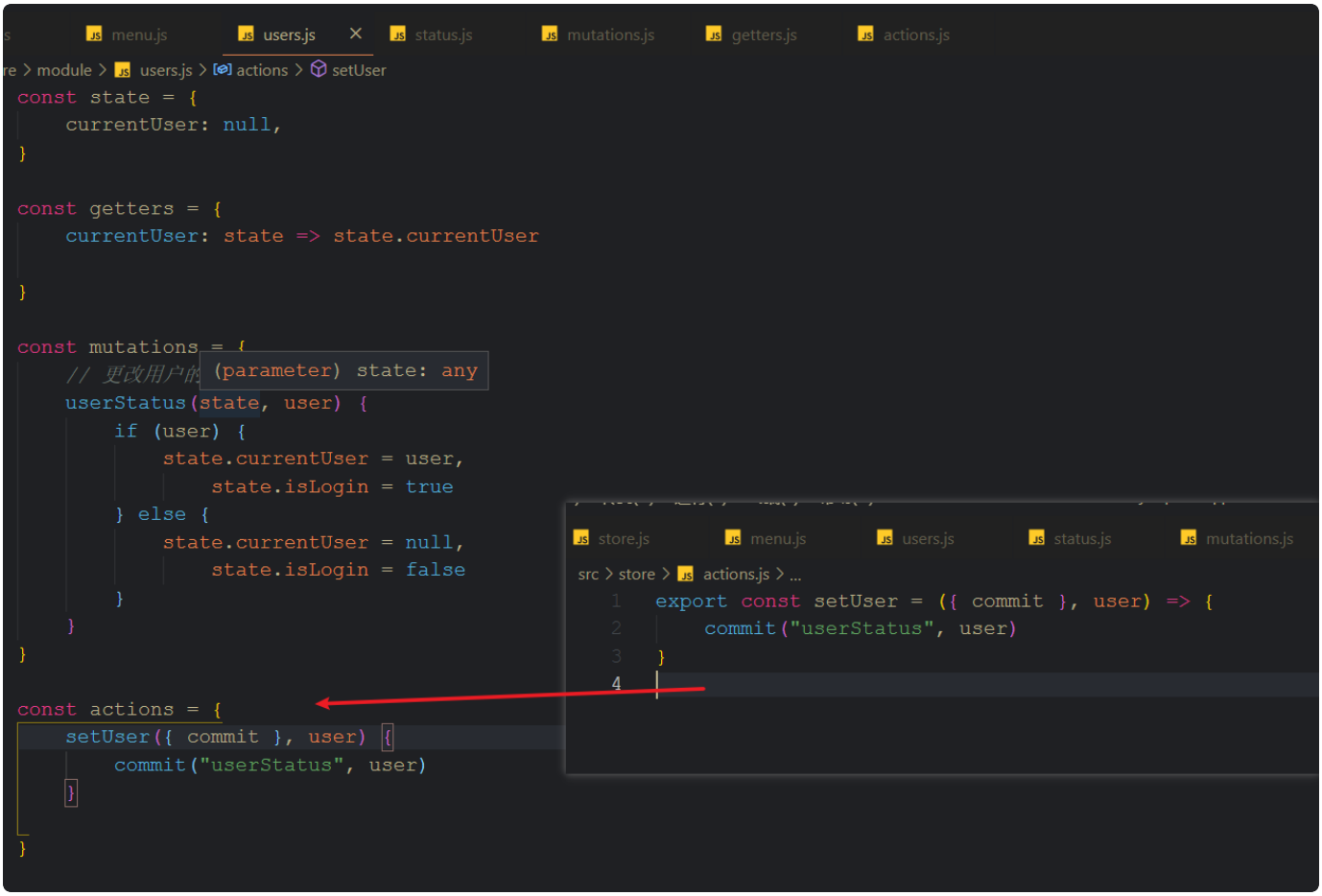


4. 抽离mutations:


store.js
menu.js
users.js
status.js
mutations.js
getters.js
actions.js
src > store > module > menu.js > mutations > pushToMenuItems
const state = {
menuItems: {},
}
const getters = {
getMenuItems: state => state.menuItems,
}
const mutations = {
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);
}
});
},
// 将新添加的食物push到menuItems属性中
pushToMenuItems(state, data) {
state.menuItems.push(data);
}
}
const actions = {}
menu.js
users.js
status.js
mutations.js
mutations.js > pushToMenuItems
// 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
src > store > module > users.js > mutations > userStatus
const state = {
currentUser: null,
}
const getters = {
currentUser: state => state.currentUser
}
const mutations = {
// 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
}
}
}
const actions = {}
export default {
state,
getters,
mutations,
actions
}
mutations.js > pushToMenuItems
// 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
}
}

5. 抽离action


menu.js
users.js
status.js
mutations.js
getters.js
actions.js
re > 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)
}
}
store.js
menu.js
users.js
status.js
mutations.js
src > store > actions.js > ...
1 export const setUser = ({ commit }, user) => {
2 commit("userStatus", user)
3 }
4

6. 引入mudoles


(V) 转到(G) 运行(R) 终端(T) 帮助(H)
store.js - pizza-app - Visual Stu
store.js
menu.js
users.js
status.js
mutations.js
src > store > store.js > store > modules > status
import Vue from "vue";
import Vuex from 'vuex';
// import * as actions from './actions'
// import * as getters from './getters'
// import * as getters from './mutations'
import menu from './module/menu';
import users from './module/users';
import status from './module/status';
Vue.use(Vuex);
export const store = new Vuex.Store({
// state: {
// // 存储数据
// // menuItems: {},
// // currentUser: null,
// // isLogin: false
// },
// getters,
// mutations,
// actions
modules: {
users,
menu,
status
}
})