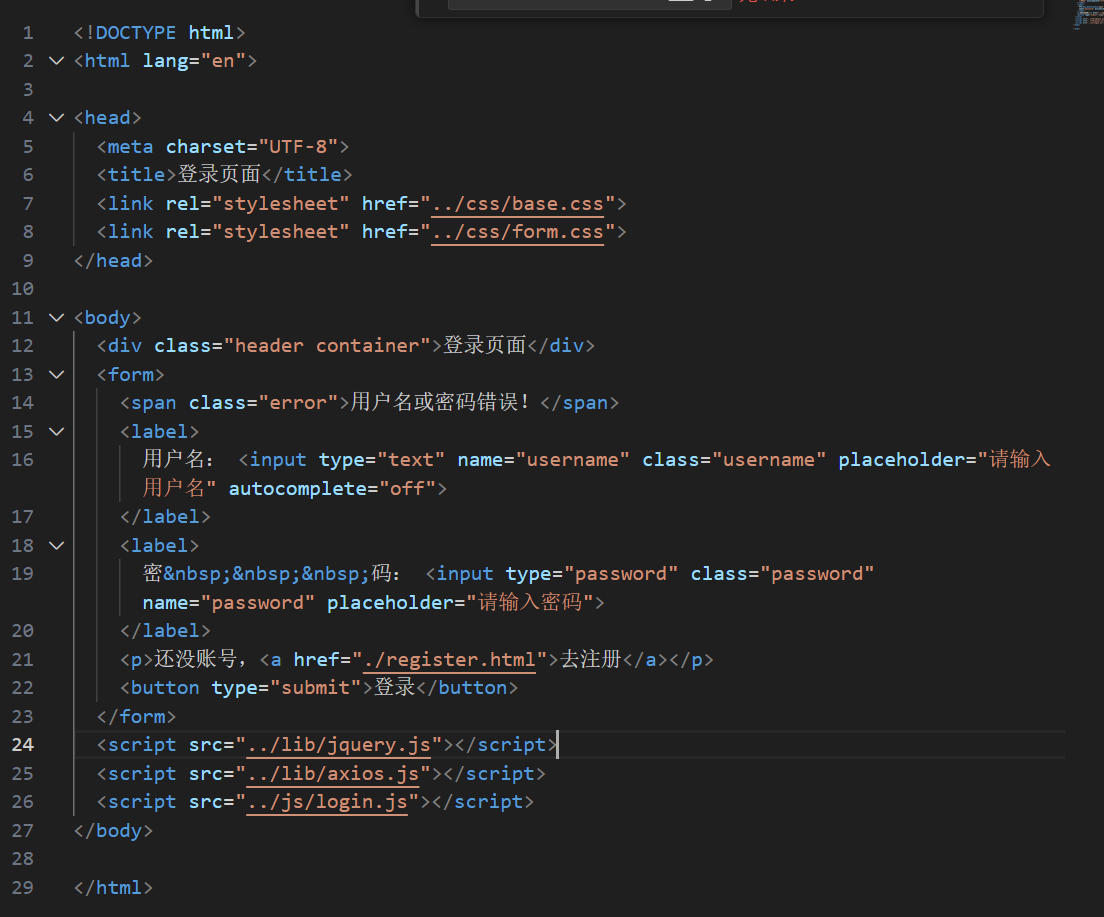
# 以太坊前端开发实验报告

前端开发的代码在原有的静态页面加了与服务器的交互功能，引用了server的api文档进行开发。

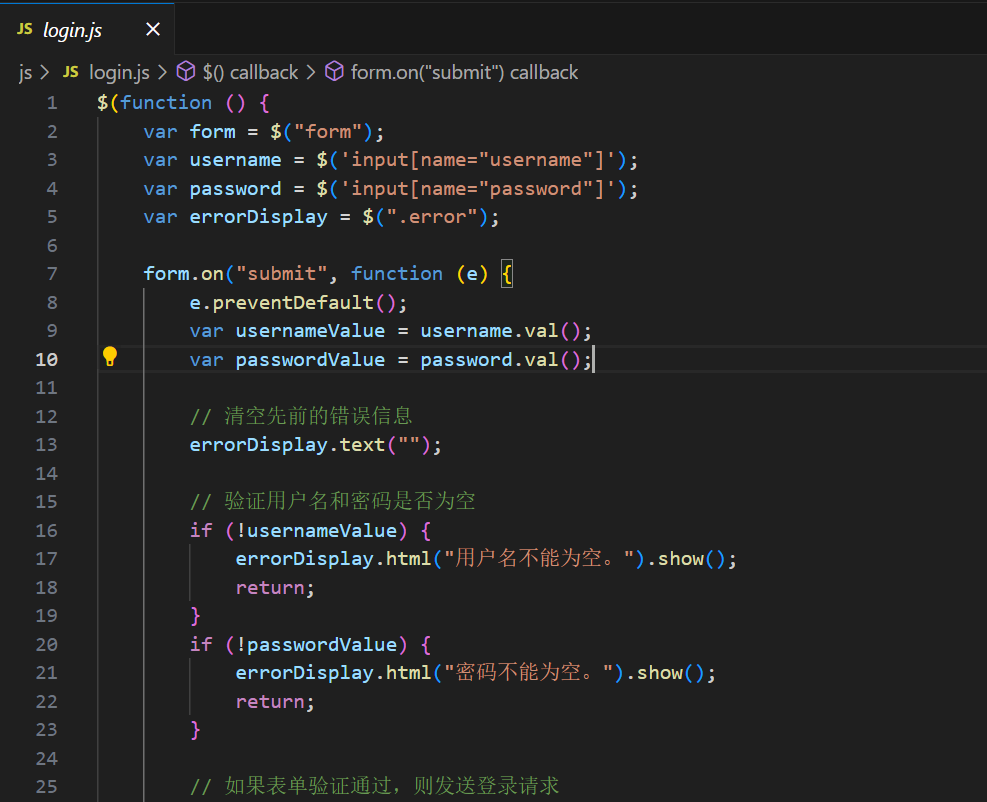
**静态页面展示：**

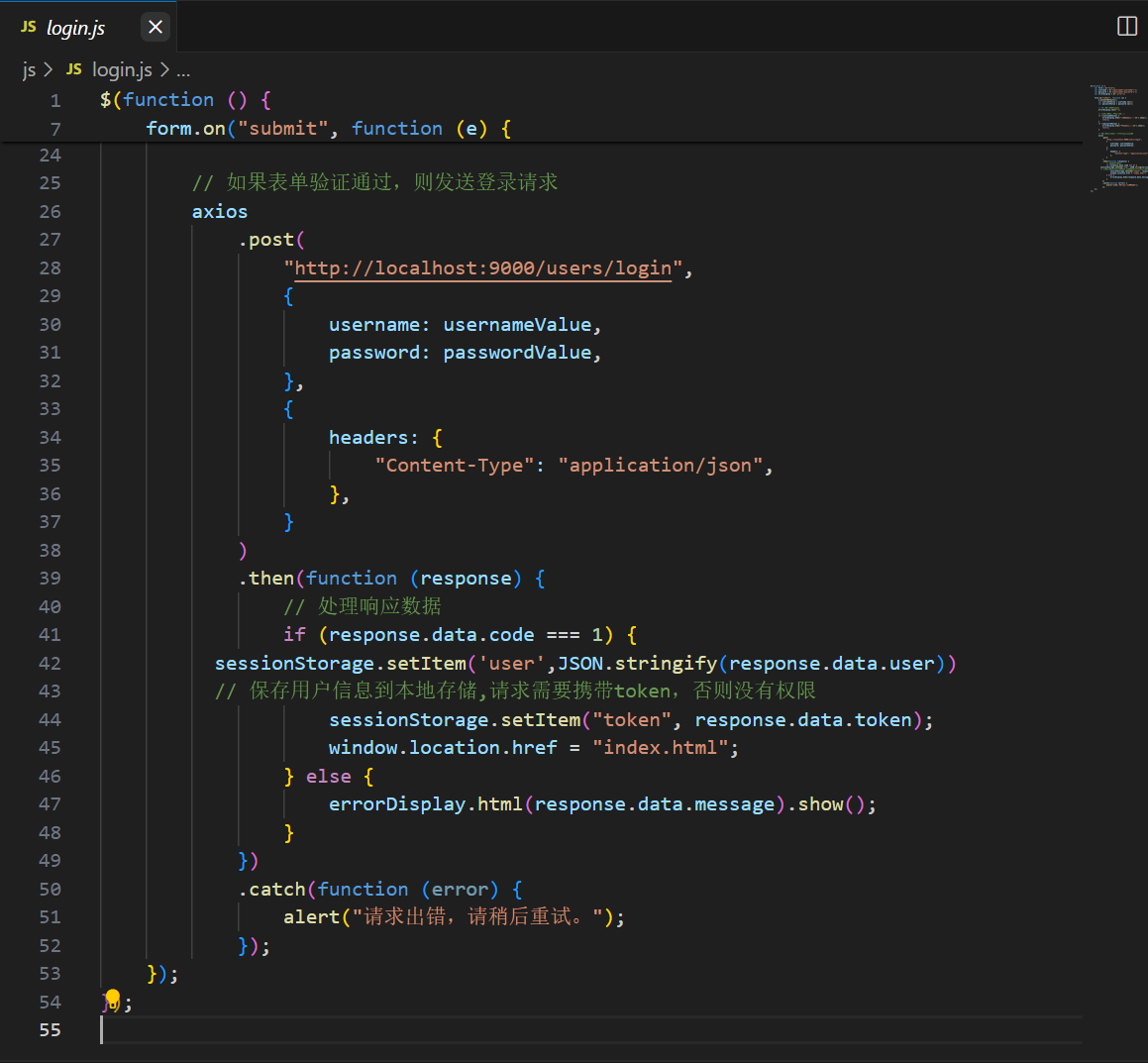
## 1、login.html





**Login.js代码如图所示**





首先用jqurey获取数据，包括表单的账号，密码，以及弹出的错误信息。

然后是当表单提交时，会触发一个事件处理器。e.preventDefault();阻止表单的默认提交行为，即刷新页面。然后是从用户名和密码输入框中获取用户输入的值。清空并验证输入。

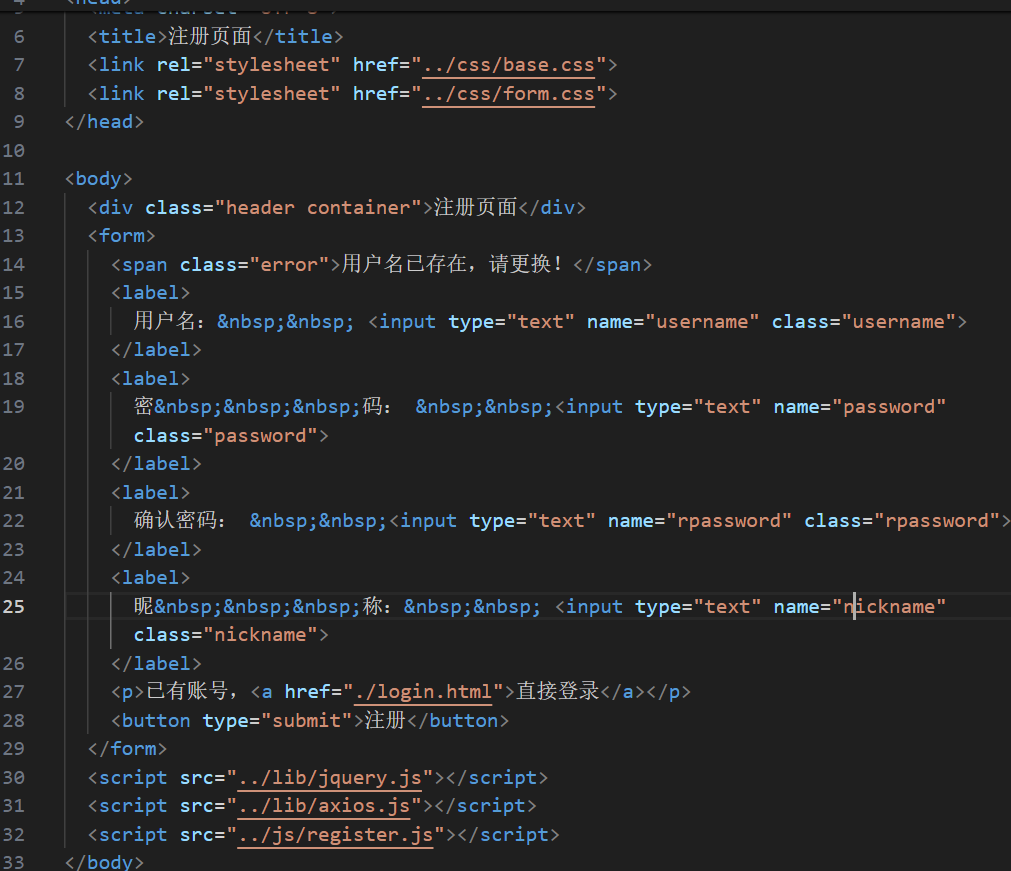
首先清空先前的错误信息，然后验证用户名和密码是否为空。如果为空，则显示相应的错误信息并停止后续代码的执行。

然后是发送登录请求，使用axios发送POST请求到服务器的登录接口。如果请求成功，会进入.then()的处理函数

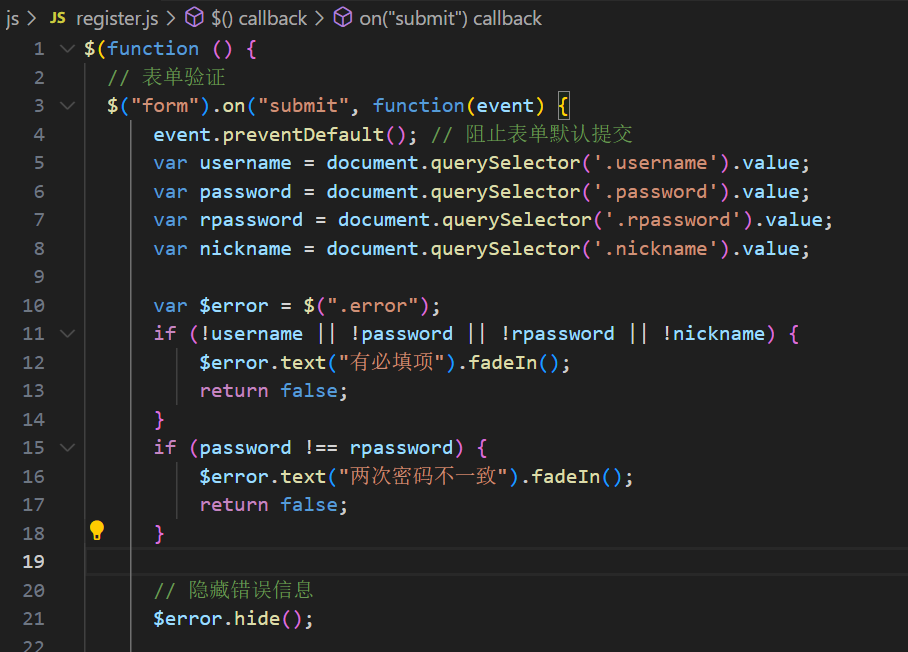
如果登录成功也就是服务器返回code: 1，则将用户信息和token保存到sessionStorage中，并跳转到index.html页面。如果登录失败，则显示服务器返回的错误信息。

如果请求失败，会进入.catch()的处理函数。

## register页面：



Register.js



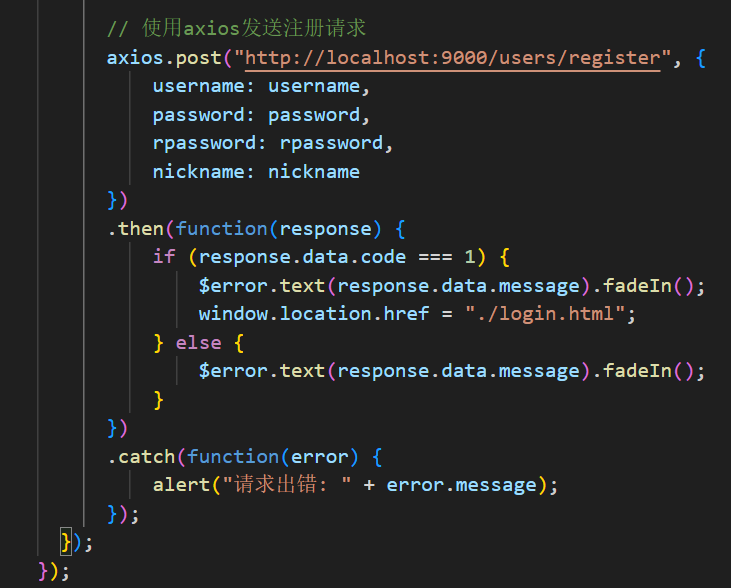
这段代码是一个使用jQuery和原生JavaScript编写的表单验证脚本，它用于在用户尝试提交注册表单时执行一系列验证，并使用axios库向后端发送注册请求。

先是jquery的自执行函数。表单提交事件监听，监听所有<form>元素的提交事件，然后阻止默认提交行为，这样我们可以进行自定义验证和处理。

使用原生JavaScript的document.querySelector方法获取表单中各个输入字段的值。

初步验证，检查用户名、密码、确认密码和昵称是否都已填写。但这里有一个逻辑错误，应该使用逻辑与&&而不是逻辑或||。如果有必填项未填写，显示错误信息。检查用户输入的密码和确认密码是否一致。javascript复制代码

在通过所有验证后，隐藏错误信息。



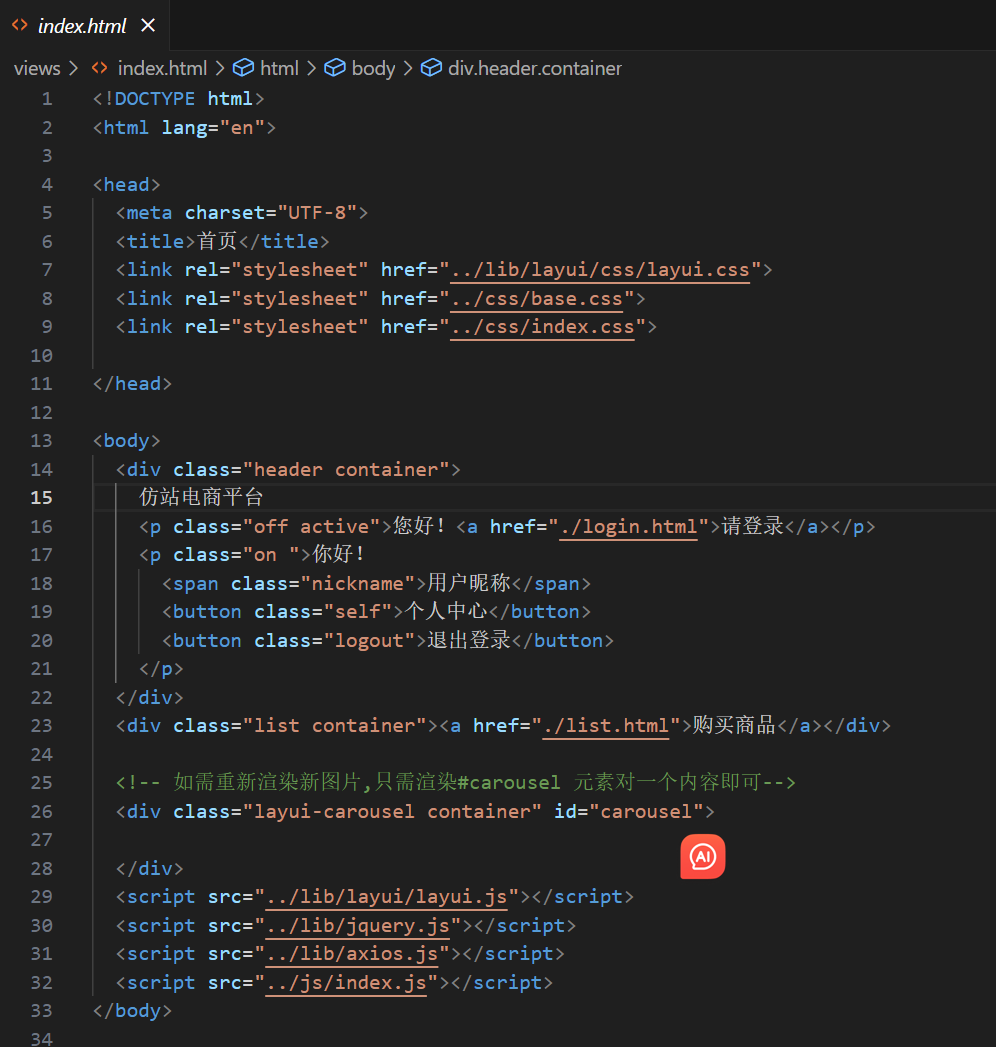
发送注册请求

使用axios库向后端发送POST请求，注册新用户。

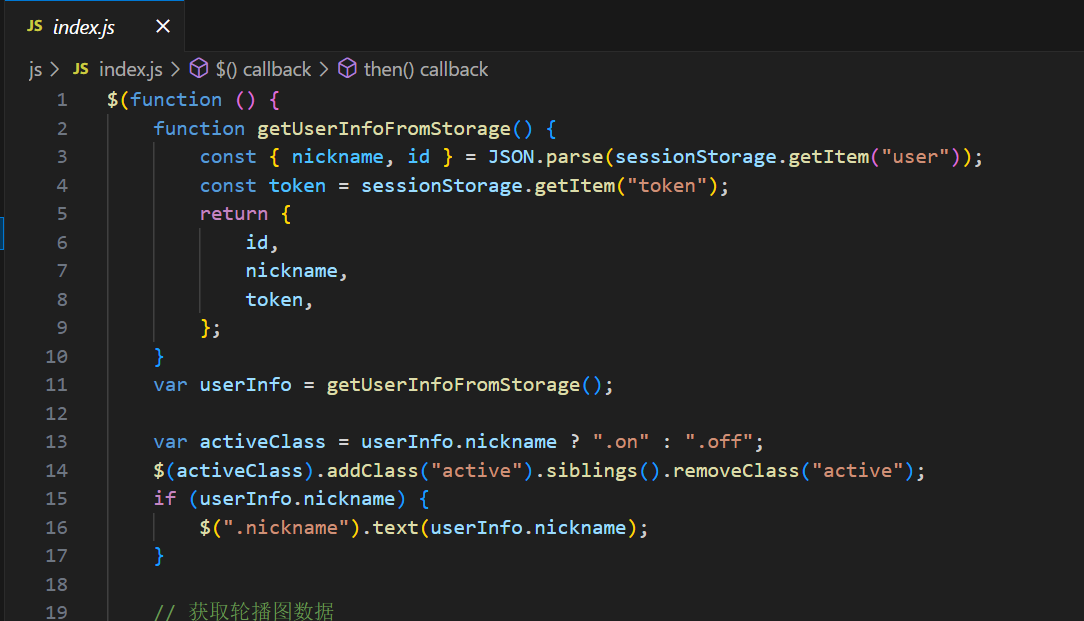
如果注册成功（假设后端返回code: 1），则显示消息并跳转到登录页面。

如果注册失败，则显示错误信息。

## 3、Index静态页面



Index.js



当页面加载完毕后，getUserInfoFromStorage 函数从 sessionStorage 中获取用户信息（昵称和ID）以及一个token,用于权限认证。如果用户信息存在，则会设置 userInfo 变量，并根据昵称是否存在来设置一个名为 activeClass 的变量，达到没用户则显示请登录，有用户则显示用户名。



使用 axios 发送 GET 请求到 http://localhost:9000/carousel/list 以获取轮播图数据。

如果请求成功并且返回的数据状态码为1，则会遍历返回的图片列表，为每个图片生成一个 div 元素并插入到 HTML 字符串中。

生成的 HTML 字符串会被设置为 ID 为 carousel 的元素的 innerHTML。

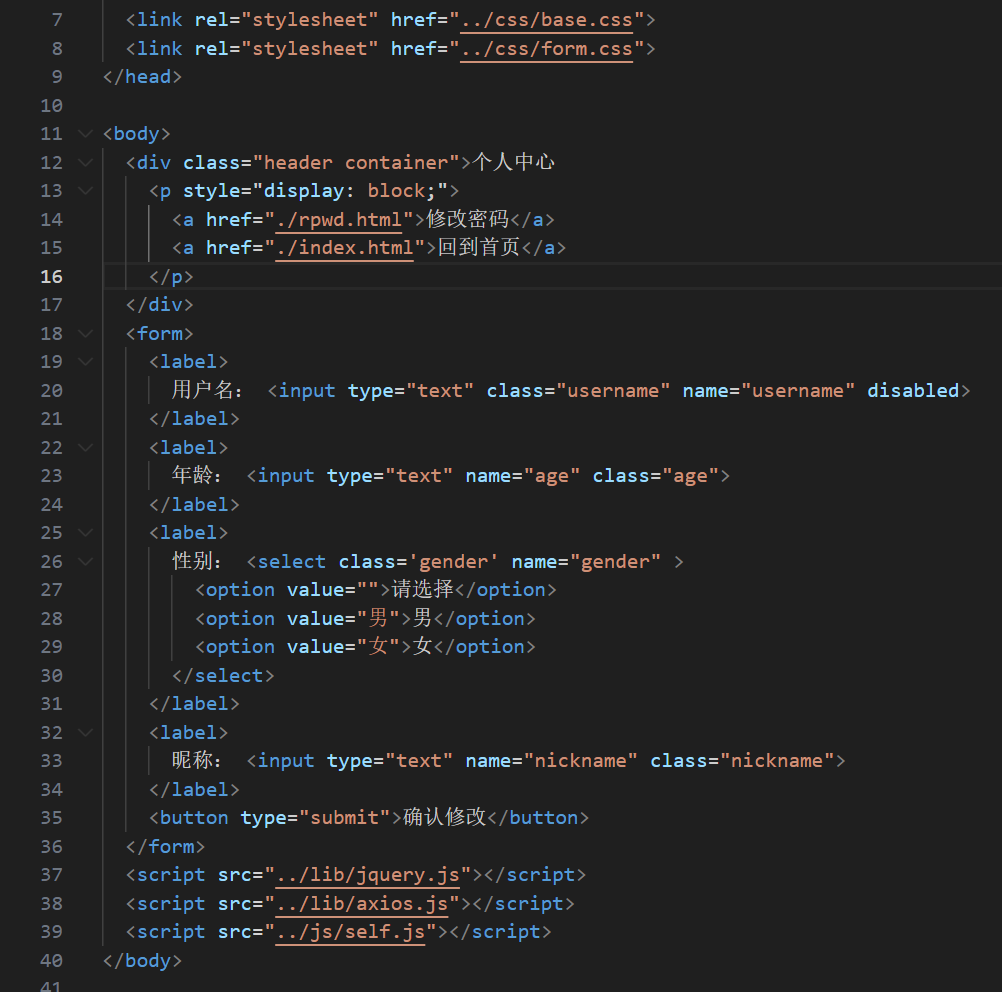
使用 layui.carousel.render 方法来初始化轮播图，并设置相关参数。



当点击具有 .self 类的元素时，页面将导航到 ./self.html。

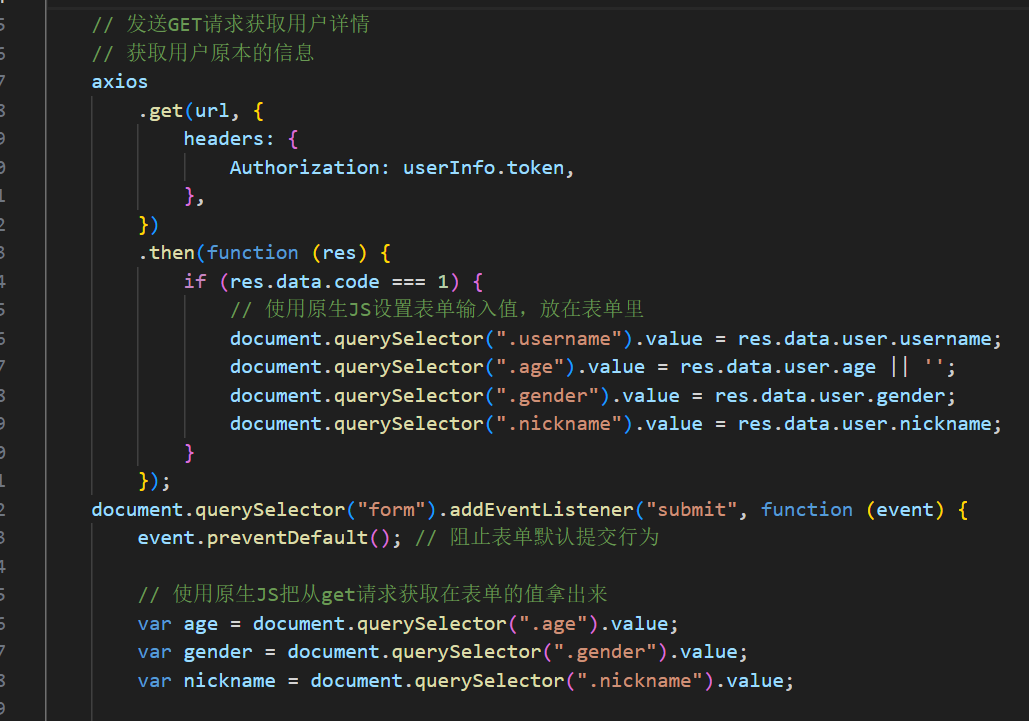
当点击具有 .logout 类的元素时，会发送一个带有用户ID的 GET 请求到 http://localhost:9000/users/logout 以进行注销操作，然后进行跳转页面。

## 4、个人中心self静态页面：



Self.js



从sessionStorage中获取用户信息：代码首先从浏览器的sessionStorage中获取用户ID和权限信息。如果这些信息不存在，则会在控制台输出错误并终止后续操作。

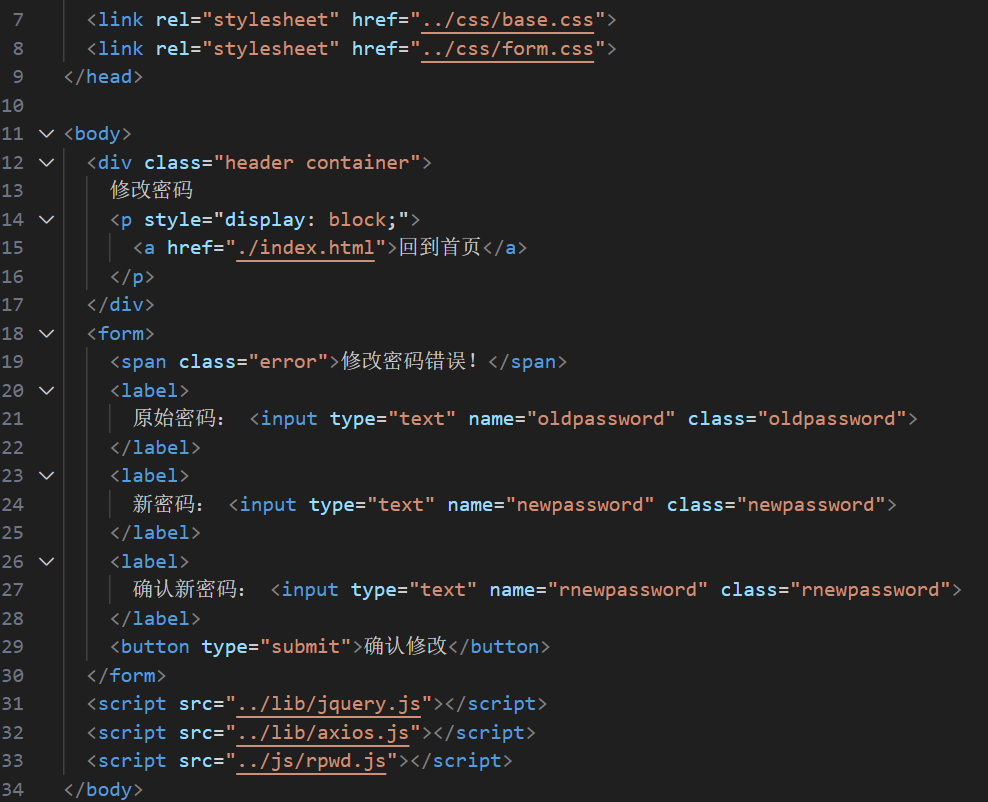
接着，代码构建了一个带有用户ID的请求URL，并使用axios发送GET请求到这个URL，以获取用户的详细信息。在请求头中，它使用了从sessionStorage中获取的token进行身份验证。

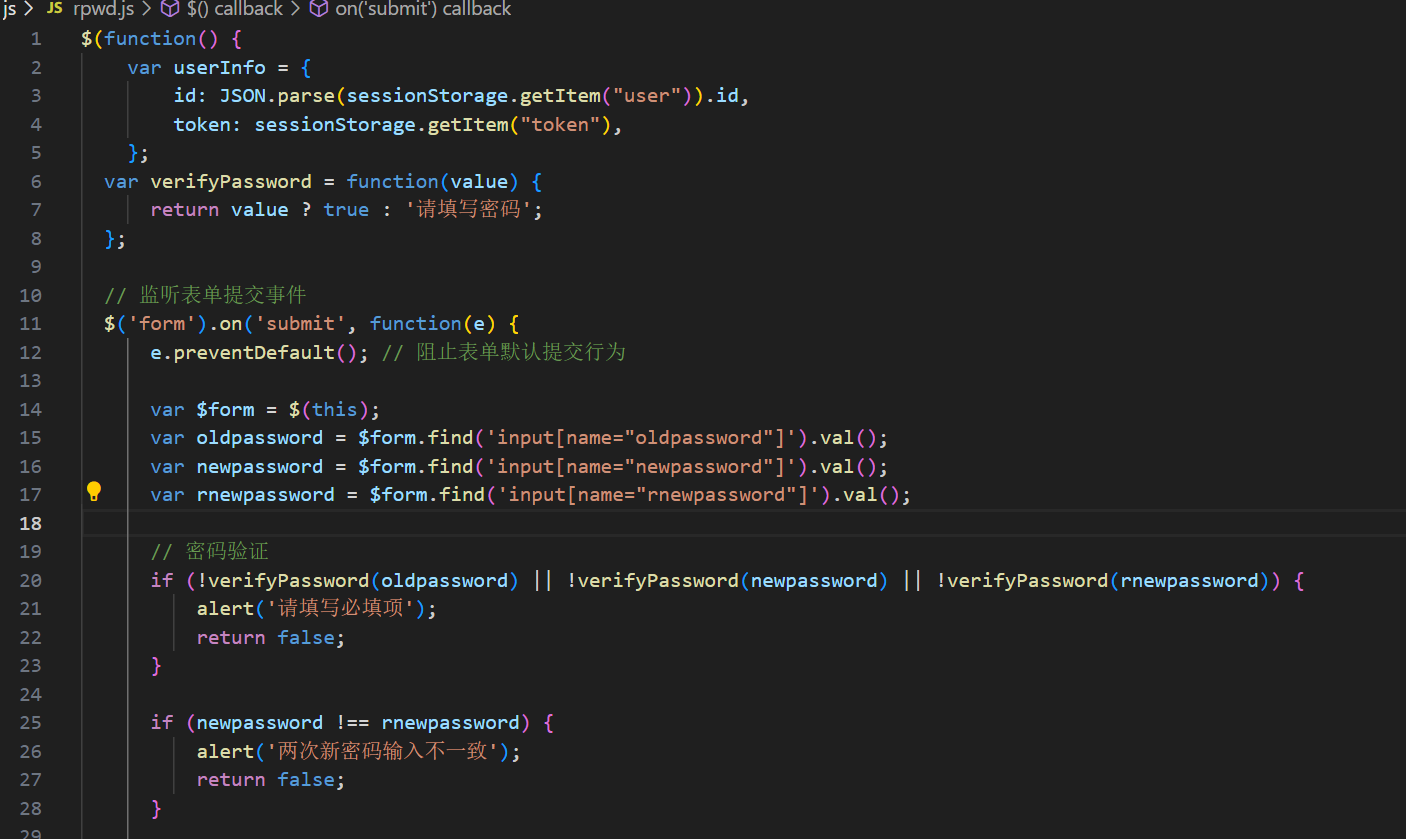
设置表单输入值：如果GET请求成功且返回的数据中code字段的值为1，则代码会使用原生JavaScript将返回的用户信息（如用户名、年龄、性别和昵称）设置到页面的表单输入框中。，然后阻止表单默认行为，拿遍历获取表单的用户信息



当表单被提交时，事件监听器首先阻止表单的默认提交行为，然后使用axios发送一个POST请求到/users/update接口，以更新用户的年龄、性别和昵称。POST请求的主体包含了用户的ID和从表单中获取的新值。

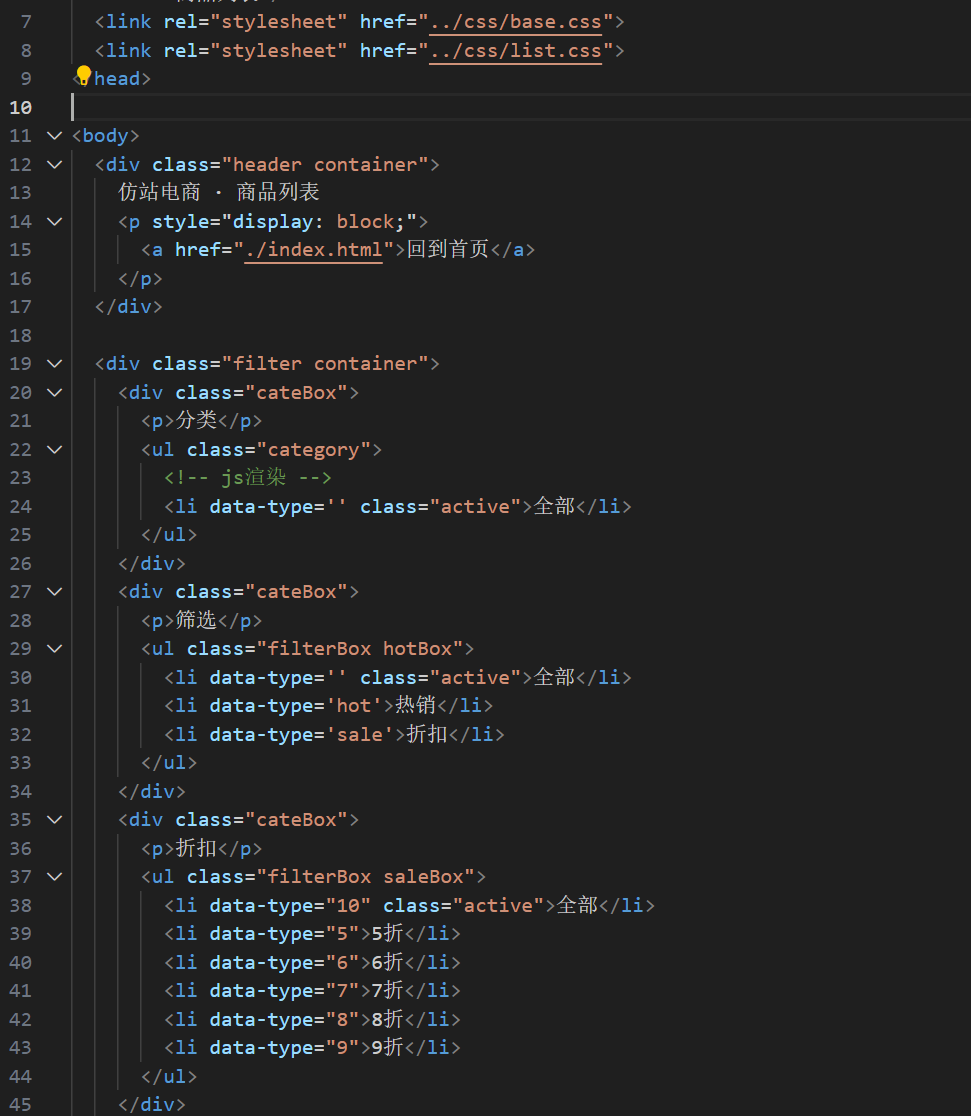
## 5、Rpwd静态页面：

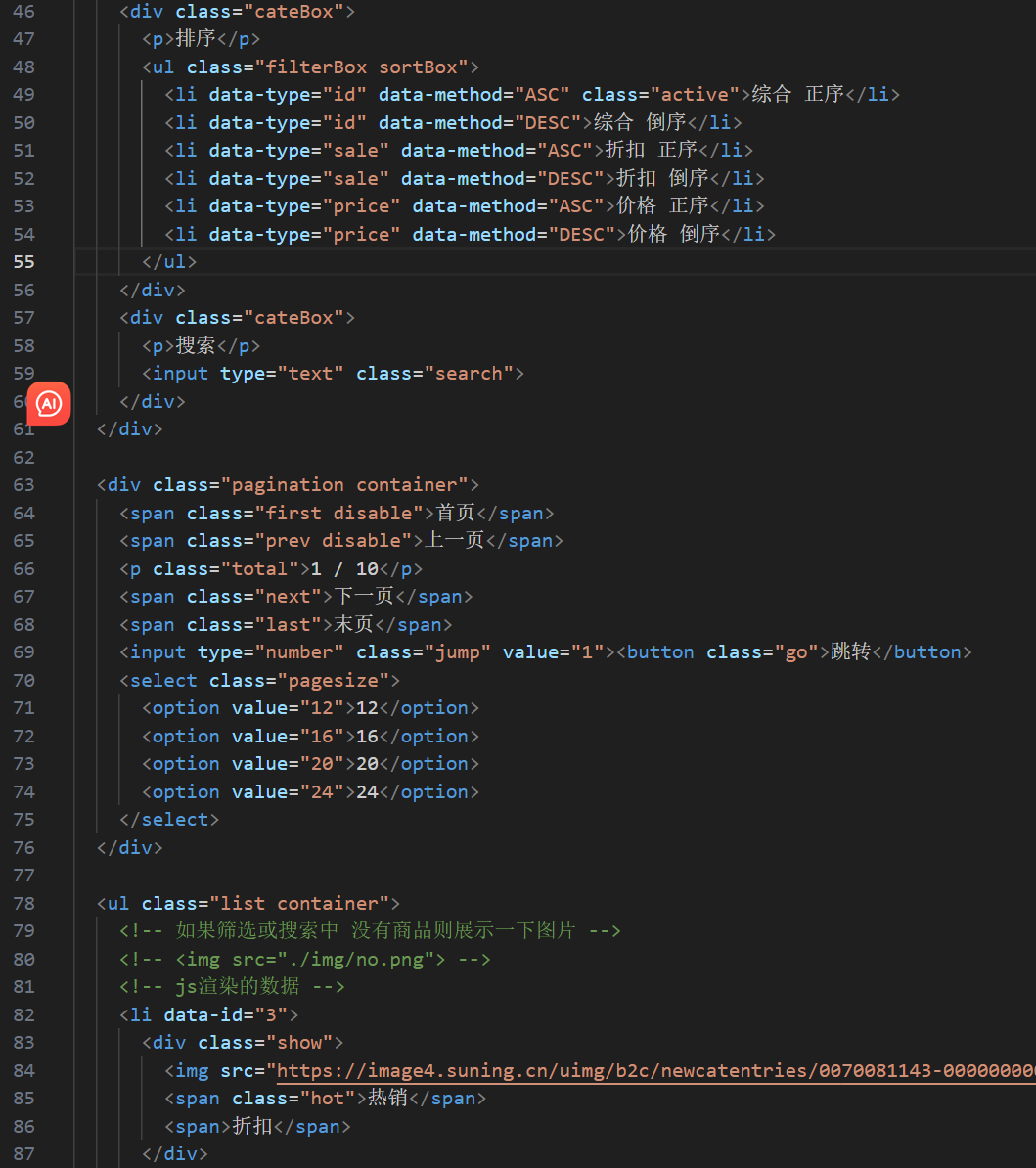
rpwd.js

从sessionStorage中获取用户信息：代码首先从浏览器的sessionStorage中获取用户ID和权限信息。如果这些信息不存在，则会在控制台输出错误并终止后续操作。

如果修改密码的框框为空，则弹出提示框。用jqurey获取提交表单，阻止默认行为，用jq获取表单的值，若为空则说明没有填写，会弹出提示。还有进行检测新密码是否一致，否则弹出框框

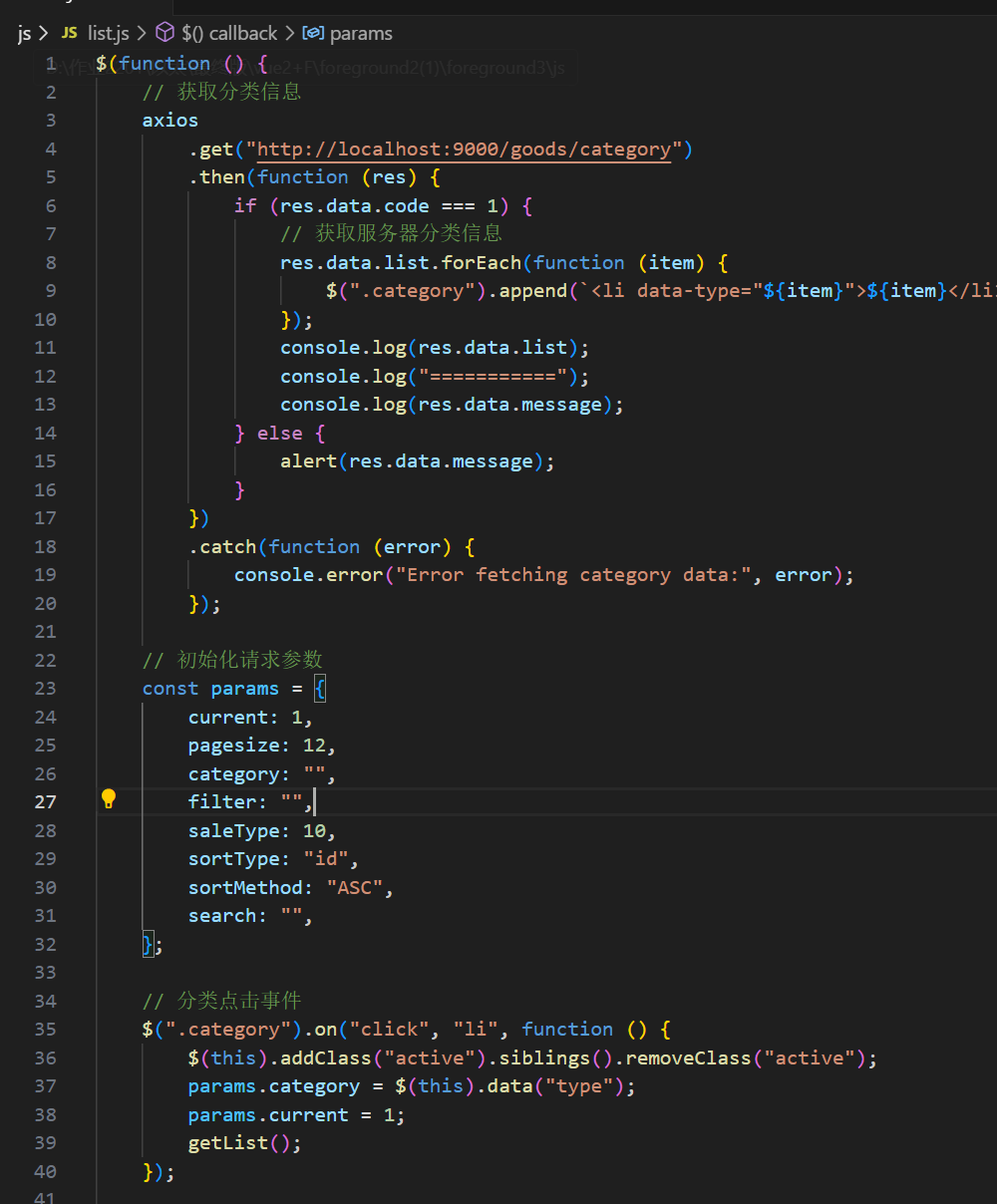
用post方法于服务器发送信息，发送头为token的权限认证，认证成功则弹出提示信息。且跳转页面

6、List静态页面：





List.js



使用 jQuery 的文档就绪函数 $(function () {...}); 来确保 DOM 完全加载后再执行代码。

axios 的 get 方法向 http://localhost:9000/goods/category 发送一个 GET 请求，以获取商品分类信息。获取后的.then(function (res) {...}); 是对成功响应的处理。

res 是一个包含服务器响应的对象。

检查 res.data.code 是否为 1。为1表示请求成功。

如果成功，遍历 res.data.list（这是一个包含所有分类的数组），并对于每个分类，使用 jQuery 的 .append() 方法在具有 .category 类的元素中动态添加一个 <li> 元素。这个 <li> 元素包含一个 data-type 属性，其值为当前分类，以及一个文本节点，显示当前分类的名称。

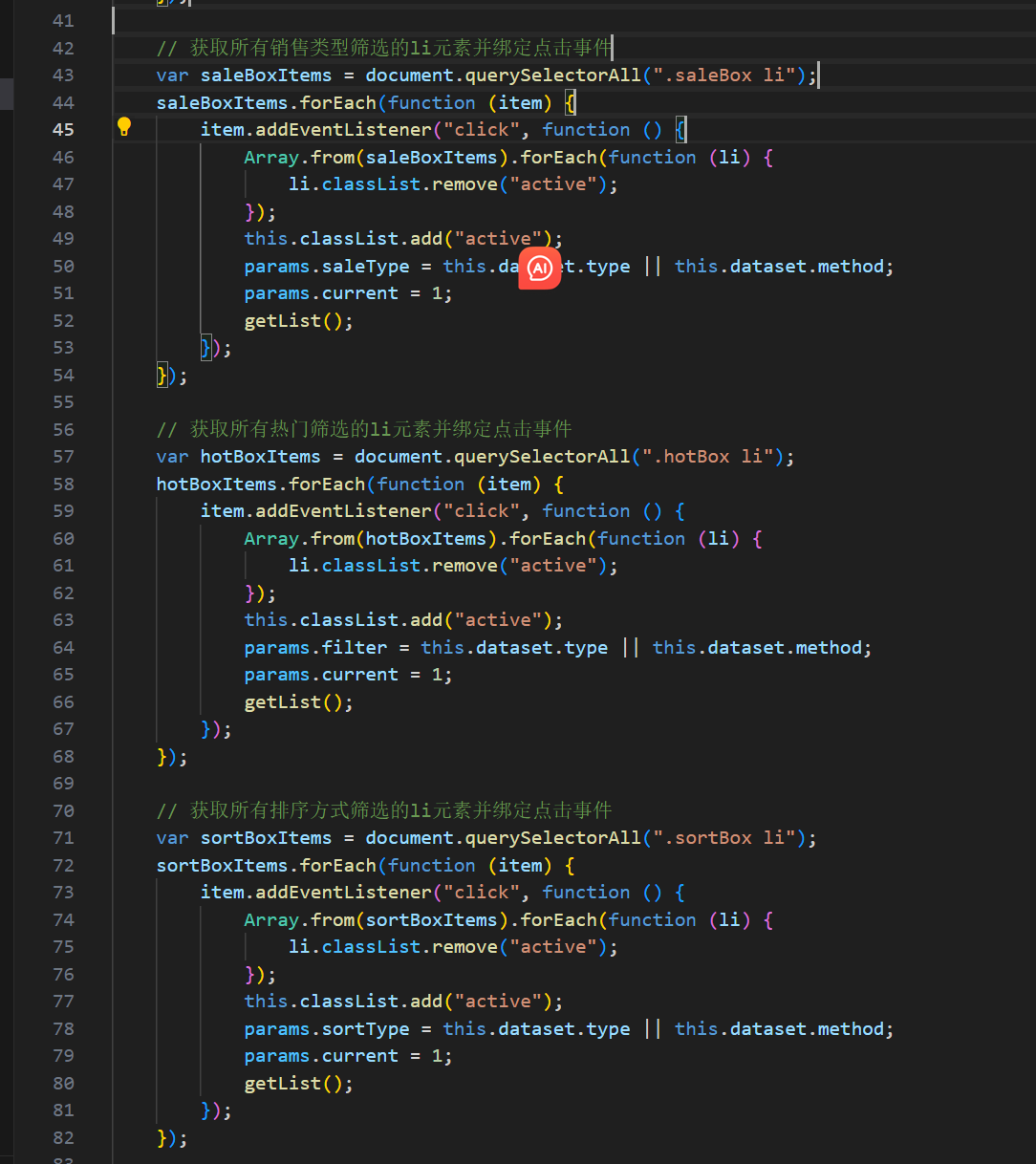
如果 res.data.code 不是 1，则请求失败，弹出一个警告框，显示 res.data.message（这通常是一个错误消息）。

当点击 .category 下的 li 元素时，会触发这个事件处理函数。

给被点击的 li 元素添加 active 类，同时移除其同级兄弟元素的 active 类，以实现视觉上的高亮效果。更新 params 对象的 category 属性为被点击 li 的 data-type 值。

将 params 对象的 current 属性重置为 1，表示要重新从第一页开始加载数据。

调用 getList() 函数



通过 document.querySelectorAll 选择所有 .saleBox li 元素，并将它们存储在 saleBoxItems 变量中。对每个 saleBoxItems 中的元素添加点击事件监听器。当某个 li 被点击时，会移除所有 .saleBox li 元素的 active 类，然后给被点击的 li 添加 active 类。更新 params 对象的 saleType 属性，取值为被点击 li 的 data-type 或 data-method同样将 params.current 页面重置为 1，并调用 getList() 函数。

选择所有 .hotBox li 元素，并为它们添加点击事件监听器。

点击事件中的逻辑与销售类型筛选相同：移除所有 active 类，给被点击的 li 添加 active 类，更新 params 对象（这次是更新 filter 属性），并调用 getList() 函数。

综上所述，这些代码片段共同实现了网页上分类、销售类型和热门筛选的动态交互功能。当用户点击不同的筛选条件时，页面会根据用户的选择更新显示的数据列表。

第三段使用了document.querySelectorAll方法选择页面上所有具有.sortBox类名的元素的直接子元素<li>。选择的结果是一个NodeList对象，它包含了所有匹配的<li>元素。然后代码使用forEach方法遍历sortBoxItems中的每个<li>元素，并为每个元素添加一个点击事件监听器。

在forEach的回调函数中，为每个<li>元素添加了一个点击事件监听器。当用户点击其中一个

然后使用排他思想，移除所有<li>元素的active类，给被点击的<li>元素添加active类

在点击事件处理函数的上下文中，this关键字引用的是被点击的<li>元素。因此，这行代码给被点击的<li>元素添加了active类，使其在页面上高亮显示。

然后是更新排序类型，这行代码从被点击的<li>元素的数据属性中获取排序类型。它首先尝试获取data-type属性的值，如果data-type不存在，则尝试获取data-method属性的值。然后，它将获取到的值赋给params对象的sortType属性。params对象可能是一个全局对象，用于存储当前页面的各种参数和状态。再将页码变为1。最后，调用getList函数

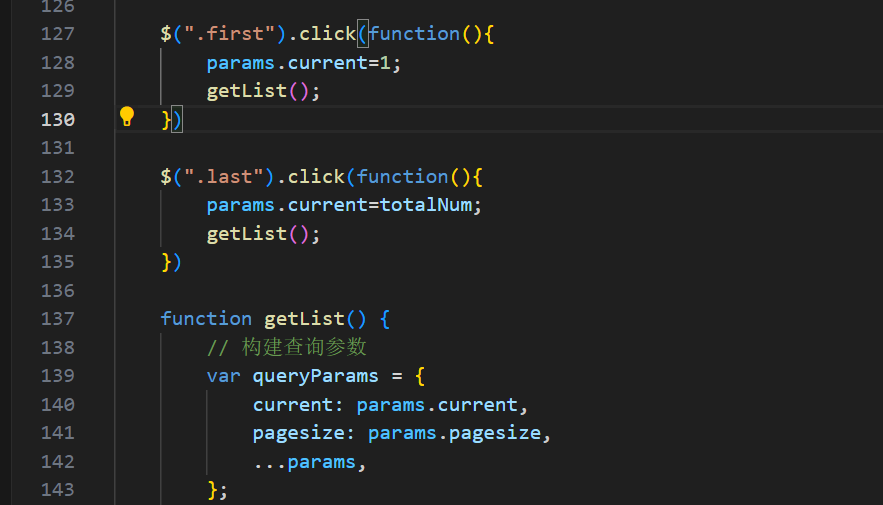


然后是失去搜索框失去焦点时，将搜索框的值给初始化的接口信息，将当前状态变为1

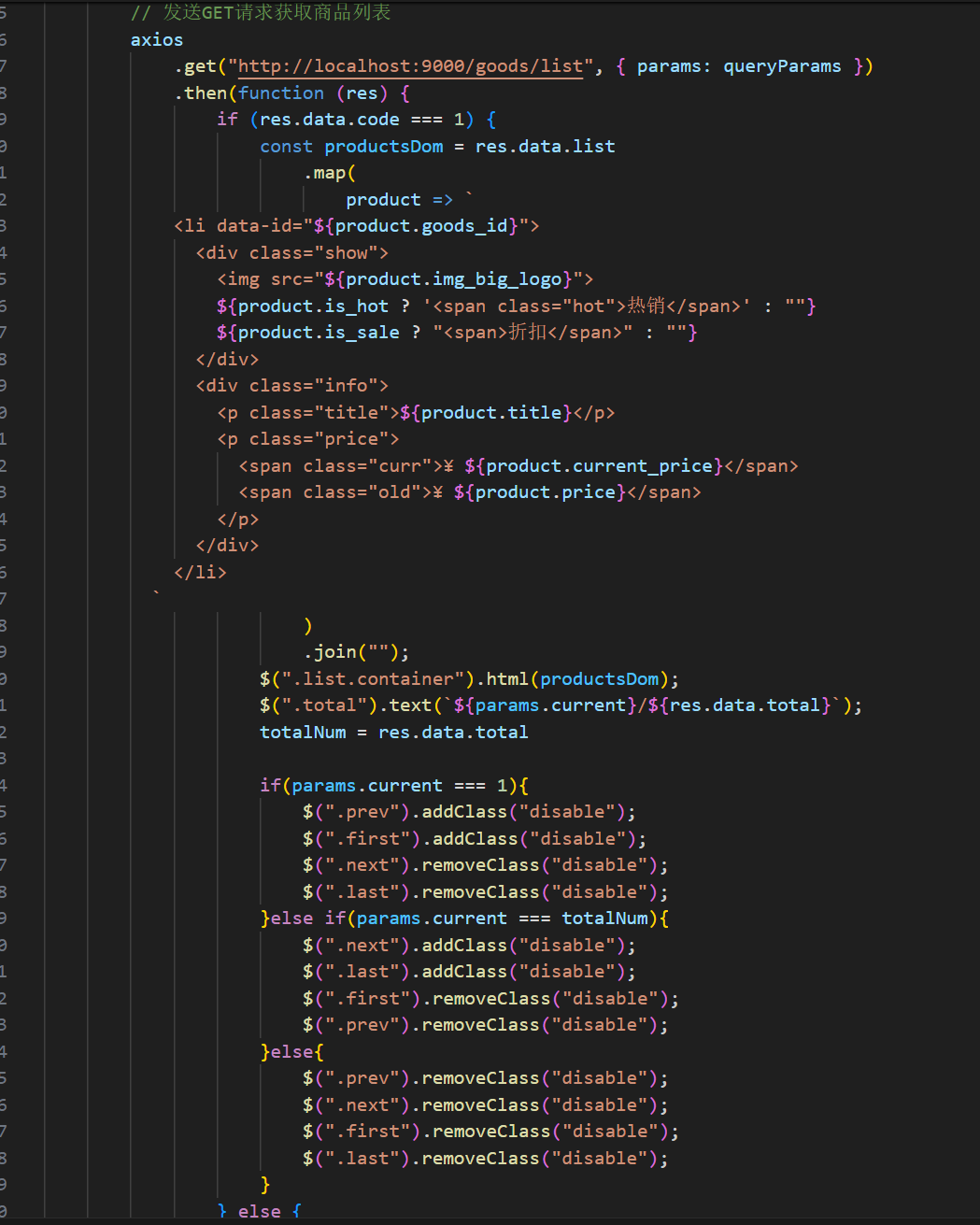
然后是跳转页面的点击事件，先判断页码是否合法，不合法则提示，合法则将页码对应更新

然后是每页最大显示数量，会把用户选中的页码变为数字，根据用户选择的页码而变成不同值

然后是点击跳转上一页或者下一页，如果超过最大页码或者小于0则显示禁用



给首页绑定点击事件，点击后页码变为第一页，然后对应页面跟着变化。末页同理



然后是给服务器发送get请求，获得服务器的页码信息，如果请求成功，遍历 res.data.list 中的每个商品对象。

使用模板字符串和 .map() 方法为每个商品创建一个 HTML 列表项 (<li>)。

使用 .join("") 将所有列表项合并为一个字符串。

使用 jQuery 的 .html() 方法将生成的 HTML 字符串插入到 .list.container 元素中。

更新 .total 元素的文本以显示当前页码和总页数。

然后是上一页首页以及下一页和首页是否禁用的判断，当页面为第一页时，清除下一页和，末页的鼠标禁用，上一页和首页为禁用。最后一页时，加入下一页和末页的禁用，清楚首页与上一页禁用。均不是则谁都不禁用



错误信息会打印到控制台

绑定点击商品详情页面，获取该商品id，将其存储到本地，跳转详情页面

## 7、详情页detail静态页面：



Detail.js



当页面加载时（通过 $(function () {...}); jQuery 的文档就绪函数）从 sessionStorage 中获取一个名为 goodsId 的项目。检查是否成功获取到 goodsId。如果没有获取到（即 goodsId 为空或不存在），则在控制台打印一条错误消息并停止后续操作。

如果成功获取到 goodsId，则使用 axios 发起 GET 请求到服务器地址

当请求成功返回时（通过 .then(function (res) {...});），检查返回的数据中的 code 是否为 1。为一则表示请求成功。如果成功则提取商品详情，并使用 jQuery 将这些信息插入到页面的 DOM 元素中。如果失败，则显示一个警告框，内容为返回的错误消息（res.data.message）。