# 前言

为了巩固已经学习的js基础、DOM操作、BOM操作，从今天开始，我们一起做购物车项目。通过项目实战打牢基础，将所学的知识真正应用到实战中。

# 知识技能篇

## 单元一 了解本项目各个页面结构

### 学习目标

了解各页面的页面结构

### 任务1.1各个页面结构、项目整体目录结构和开发工具

#### 任务描述

了解本次项目各个页面的构成。

了解项目的目录构成。

以及使用的开发工具。

#### 任务实施

1.各个页面的构成。

1.1登录页面



1.2注册页面



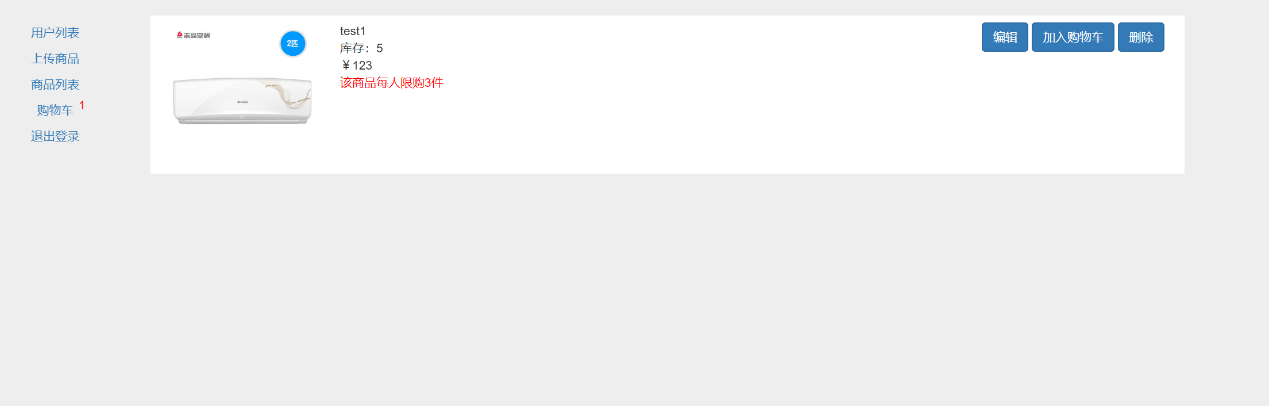
1.3用户列表



1.4 上传商品



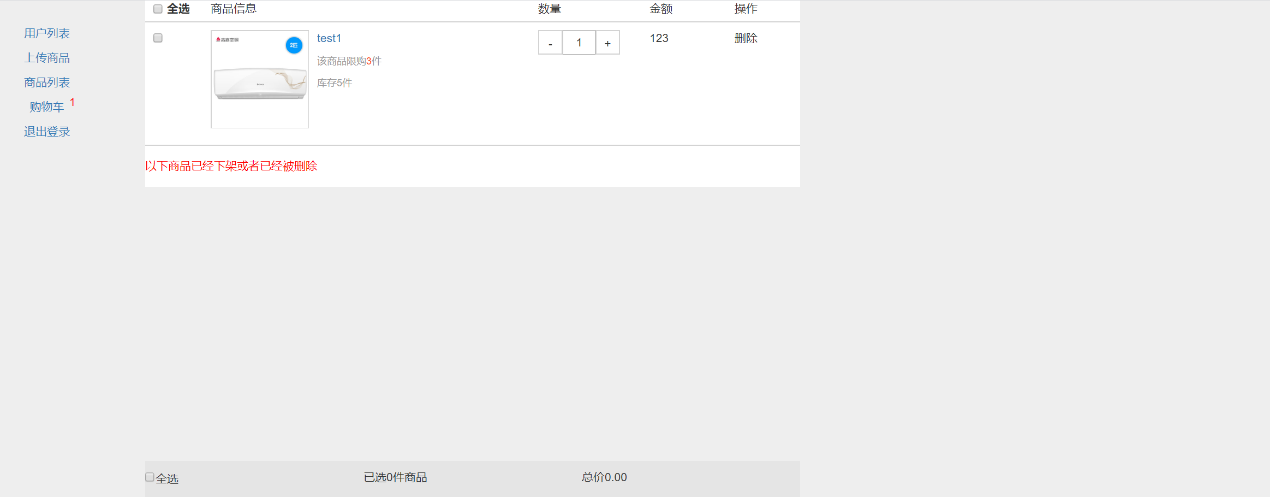
1.5 商品列表



1.6 编辑商品

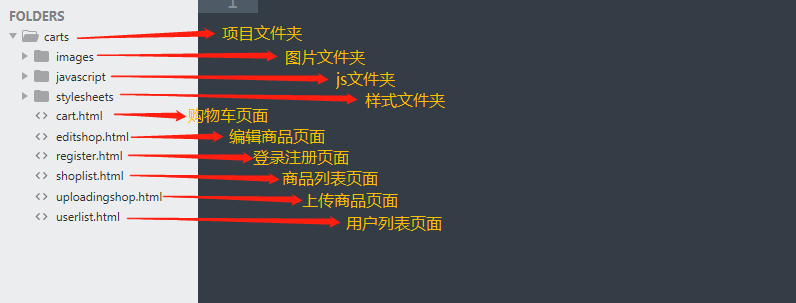


1.7 购物车列表



1.8退出登录

2.目录构成



3.开发工具

本次项目开发，使用的是sublime，也可以使用其他编译器。

3.1 打开sublime，需要安装一个网络服务的插件。

3.2找到 Preferences => Package Control，打开后输入 Install Package 后点击，然后等待一段时间，输入 Sublime Server 然后等待安装。

3.3安装完之后，打开Tools => SublimeServer => start SublimeServer启动插件

3.4右击编辑器内容 => view in SublimeServer

# 技术应用篇

## 单元二 封装XMLHttpRequest对象

### 学习目标

掌握xmlhttprequest对象的使用，掌握json相关方法，掌握获取id元素的方法，掌握Promise对象的使用。

### 任务2.1 封装 $\_ajax 和封装$\_getID

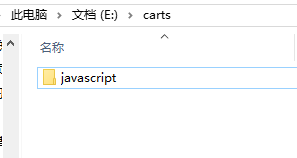
#### 任务描述

封装一个函数用来向后台发送数据请求。

封装一个dom通过id获取节点的函数。

#### 任务实施

1. 在电脑的磁盘上，新建一个carts文件夹，在carts中创建一个javascript文件夹，在javascript文件夹中创建一个ajax.js文件，公共部分的js我们都会放到这个文件夹内。（注意：全程不能有中文目录名。）



2.打开ajax.js文件，封装$\_ajax。

2.1创建一个全局变量$\_http用来保存后台接口地址，在开发的时候我们需要根据后台地址，修改这个值即可。

本次的演示的后台接口为 http://49.235.117.27:3000

window.$\_http = ' http://49.235.117.27:3000'; //替换为后台接口

2.2创建一个全局函数$\_ajax接收type，url，data参数，将以下代码放到ajax.js文件中。

window.$\_ajax = function(type,url,data){

}

2.3将接收到的type值转化为大写并判断type值是post还是get，如果是get形式则将参数拼接在url后面，如果是post则将数据转化为json字符串。将以下代码放到$\_ajax函数内部。

url = $\_http + url;

if(type.toUpperCase() === "GET"){

url += '?';

var arr = [];

for(var i in data){

arr.push(i+'='+data[i])

}

url += arr.join('&');

data = null;

}else{

data = JSON.stringify(data);

}

2.4因为ajax是异步请求，所以我们使用return返回一个promise对象，将以下代码放到上一步的后面。

return new Promise(function(resolve,reject){

var xhr = new XMLHttpRequest();

xhr.open(type,url);

xhr.setRequestHeader("Content-Type",'application/json')

if(localStorage.getItem('token')){

xhr.setRequestHeader("token",localStorage.getItem('token'))

}

xhr.onreadystatechange = function(){

if(xhr.readyState === 4 && xhr.status === 200){

var data = JSON.parse(xhr.responseText);

if(data.code === -99){

location.href = data.redirect

}else{

resolve(data);

}

}

}

xhr.send(data)

})

3. 创建一个全局函数$\_getID，返回获取到的元素对象，将以下代码复制到$\_ajax方法下面。

window.$\_getID = function(el){

return document.getElementById(el);

}

## 单元三 封装提示框方法

### 学习目标

掌握css3动画，掌握设置定时器和清除定时器的方法。

### 任务3.1 文件结构，css样式，弹框js

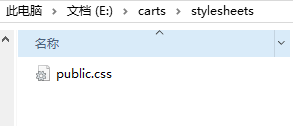
#### 任务描述



#### 任务实施

1.文件结构

在carts文件夹中新建stylesheets文件夹，在这个文件夹中新建public.css文件。



2.编写css样式。

在编辑器中打开public.css文件，在public.css文件中写入如下样式。

#no-title-hint{

padding:20px 10px;

min-width:400px;

position:fixed;

top:100px;

left:50%;

transform: translateX(-50%) scale(0);

transition: all 0.2s ease;

background:#fff;

text-align: center;

border-radius: 4px;

font-size: 16px;

color: #fff;

background: rgba(0, 0, 0, 0.8);

}

3.编写弹框js

3.1创建一个全局变量$\_golbal\_timer，用来保存setTimeout的返回值，当用户重复触发这个方法的时候，我们清除上一个定时器，这样可以拥有更好的用户体验。创建一个$\_hint全局函数，接收一个text字符串，当做提示文本。将以下代码放到ajax.js文件末尾。

window.$\_global\_timer = null;

window.$\_hint = function(text){

}

3. 2.编写$\_hint函数主体。先定义一个变量，通过id获取id名为no-title-hint的元素，赋值给该变量。判断页面中是否存在id名为no-title-hint的元素，有则把传进来的值用innerText添加到节点中。添加动画样式，向左移动自身宽度的50%，还有缩放的形式显示在屏幕中。

var box;

clearTimeout($\_global\_timer);

box = document.getElementById('no-title-hint');

if( box ){

box.innerText = text;

box.style.cssText = "transform: translateX(-50%) scale(1);"

}else{

var box = document.createElement('div');

box.id = 'no-title-hint';

box.innerText = text;

document.body.append(box);

box.style.cssText="transform: translateX(-50%) scale(1);";

}

3.3设置定时器，1.5秒后该提示框消失。

$\_global\_timer = setTimeout(function(){

box.style.cssText = "transform: translateX(-50%) scale(0);"

},1500)

## 单元四 登录注册页面

### 学习目标

完成登录注册功能。

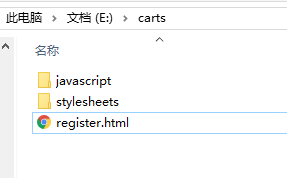
### 任务4.1 搭建文件结构，设置默认css样式

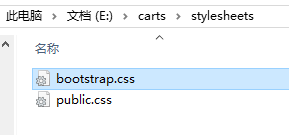
#### 任务描述

创建页面，设置常用css样式，还有首页html页面。

#### 任务实施

1. 在carts文件夹内新建register.html文件。在stylesheets文件夹中将下载的bootstarp.css文件粘贴进来。





用编辑器打开register.html，先编写模板。并引入bootstrap.css和publick.css两个样式表，代码如下。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link rel="stylesheet" href="./stylesheets/bootstrap.css">

<link rel="stylesheet" href="./stylesheets/public.css">

<title>登录、注册页面</title>

</head>

<body>

</body>

</html>

2.首先在<body>中，写入一个<div>标签，设置类名为container box。并在该div标签中定义一个form表单，并设置该表单的一些样式。

<div class="container box">

<form style="width: 400px;margin:100px auto;">

</form>

</div>

3.在form表单中，用h2标签作为标题

3.1加入三个类名为input-box的div，在这三个div中分别放着一个span标签和一个input标签。

3.2一个a超链接

3.3一个id为 btn 的input标签。代码如下。

<h2 id="title">登录</h2>

<div class="input-box">

<span class="fl">用户名</span>

<input class="form-control" placeholder="2-6个数字和中文，必须中文开头" type="text" id="username">

</div>

<div class="input-box">

<span class="fl">密码</span>

<input type="password" class="form-control"

placeholder="长度5-16位的字母数字组合，必须字母开头" id="ps">

</div>

<div class="input-box" id="codebox">

<span class="fl">验证码</span>

<input type="text" class="form-control" placeholder="验证码" id="code">

</div>

<a href="javascript:" class="pull-right" id="goLogin"></a>

<div id="btn" class="btn btn-lg btn-primary btn-block" style="margin-top: 30px;">登录</div>

3.4在head标签中，加入对应的样式。

<style>

.form-control {

height: 40px;

}

.input-box {

position: relative;

padding-bottom: 10px;

overflow: hidden; /\*清除浮动\*/

}

.box .form-control {

float: right;

width: 80%;

}

.box .input-box span {

line-height: 40px;

}

</style>

4. 清除默认样式和编写公共样式。

打开public.css文件，在public.css文件中写入如下样式。

\*{

margin: 0;

padding: 0;

}

html,body{

height: 100%;

background:#eee;

}

ul,ol{

list-style: none;

}

a{

text-decoration: none;

}

.fl{

float: left;

}

.fr{

float: right;

}

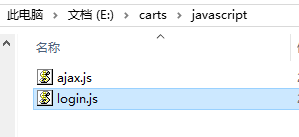
### 任务4.2 js交互。

#### 任务描述

完成首页登录注册js功能

#### 任务实施

在javascript文件夹中，创建 login.js 文件。



1.打开login.js文件

定义一个onload函数，定义用户名和密码的两个正则表达式。

判断当前页面是登录页面还是注册页面，如果url参数中含有login字符串则表示当前是登录页面，否则切换到注册页面。点击‘还没有账号，去注册’或者‘已有账号，去登陆’均会切换到对应的页面。

如果是登录页面的话设置验证码的盒子隐藏。

window.onload = function () {//在全部的html加载完成后开始执行js

var psReg = /^[a-zA-Z]{1}[a-zA-Z0-9]{4,15}$/;//数字和字母 字母开头 长度5-16

var usernameReg = /^[\u4e00-\u9fa5]{1}[\u4e00-\u9fa50-9]{1,5}$/;

var txt1 = "", txt2 = "", hrefTxt = "";//定义一些变量

if (location.search.indexOf('login') !== -1) {//通过判断url参数中是否有login字符串来判断当前是登录操作还是注册操作。

txt1 = '登录';//当前是登录操作

txt2 = '还没有账号，去注册';

hrefTxt = 'http://localhost/carts/register.html?type=register';

$\_getID('codebox').style.display = 'none'

} else {

//当前是注册操作

txt1 = "注册";

txt2 = '已有账号，去登陆';

hrefTxt = 'http://localhost/carts/register.html?type=login';

}

//将变量存到对于的html中

$\_getID('btn').innerText = txt1;

$\_getID('title').innerText = txt1;

$\_getID('goLogin').innerText = txt2;

$\_getID('goLogin').setAttribute('href', hrefTxt);

// js代码1

}

2.点击登录或者注册按钮，获取用户填写的信息，使用上面定义的正则表达式判断用户信息填写的是否符合要求，通过判断变量txt1的值来确定用户需要执行登录还是注册。注册用户验证码请填 ‘hkcwjq’。

2.1注册：

判断为注册的话，先在外部定义一个$timer变量，每次注册的时候先清除一下$timer的定时器。

判断用户验证码是否为空。不为空则判断账号是否正确，密码因为在外面已经判断，所以此处不用判断。

判断完成后，调用$\_ajax方法请求/register接口，将用户名，密码，验证码作为参数传给后台，将返回的信息调用提示框方法。提示客户是否注册成功。

添加定时器，等提示框缩放完后，该定时器执行跳转到登录页面。

2.2登录：

调用$\_ajax方法请求/login接口，将用户名和密码作为参数传给后台。请求成功后，利用请求的返回值进行判断。如果登录成功。则将token值放在浏览器本地存储中。然后跳转到用户列表页。

代码如下，将以下代码复制到//js代码1的下一行实现上述功能。

//点击注册或者登录按钮的时候，提交用户信息到后台做验证

$\_getID('btn').onclick = function(){

var flag = true;

var hint = "";

var zh = $\_getID('username').value;

var ps = $\_getID('ps').value;

if( !usernameReg.test(zh) ){

flag = false;

hint+='用户名必须是2-6位以汉字开头汉字和数字组合，可以全部为汉字'

}

if( !psReg.test(ps) ){

flag = false;

hint+='密码必须同时包含字母、数字，必须字母开头';

}

if( flag ){

var $timer = null;

if( txt1 === '注册' ){//判断当前是注册功能还是登录功能

clearTimeout($timer);

var code = $\_getID('code').value;

if( !code ){

$\_hint('请输入验证码')

return;

}

$\_ajax('post','/register',{userName:zh,ps:ps,code:code}).then(res=>{

$\_hint(res.msg)//调用之前写好的提示框，来提示用户，详细信息可以查看后台开发文档

})

$timer = setTimeout(function(){

location.href = 'http://localhost/carts/register.html?type=login'

},1500)

}else{

$\_ajax('post','/login',{userName:zh,ps:ps}).then(res=>{

if( res.state ){

localStorage.setItem('token',res.token)

location.href="http://localhost/carts/userlist.html";

}else{

$\_hint('用户名或密码错误')

}

})

}

}else{

alert(hint);

hint="";

flag = true;

}

}

3.js文件编辑完之后打开register.html文件，在<body/>标签外部，引进ajax.js和login.js文件。

<script type="text/javascript" src="javascript/ajax.js"></script>

<script type="text/javascript" src="javascript/login.js"></script>

## 单元五 侧边栏

### 学习目标

innerHTML和insertAdjacentHTML的使用。

使用js完成侧边栏。

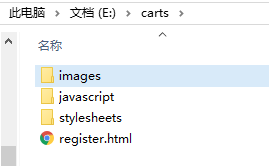
### 任务5.1 侧边栏的js交互

#### 任务描述

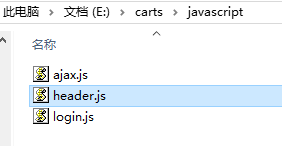
字符串拼接规则和技巧

#### 任务实施

在项目文件夹下面创建一个images的文件夹，将所需要使用到的图片添加到里面。



1.在javascript文件夹中新建header.js文件。因为侧边栏除了登录和注册页面外，其他页面都需要用到，所以在header.js文件中，封装成一个函数。



2.打开header.js，封装publicheader方法，在publicheader方法中放入左边栏这个公共样式，把所有节点标签写完存到一个变量中，然后把这个变量添加到页面中。写完后调用该方法。

function publicheader(){

var str = `

<ul class="sidebar">

<li>

<a href="userlist.html">用户列表</a>

</li>

<li>

<a href="uploadingshop.html">上传商品</a>

</li>

<li>

<a href="shoplist.html">商品列表</a>

</li>

<li>

<a href="cart.html" id="sidebar-cart">

购物车

<div class="cart-num">

<image src='./images/loading.png' class='loading-animate' style='width:18px;height:18px;'>

</div>

</a>

</li>

<li data-type="exit" id="exit">

<a href="javascript:" >退出登录</a>

</li>

</ul>

`;

document.body.insertAdjacentHTML('beforeend', str);

}

publicheader();

3.获取购物车内商品数量，创建getCArtNumber方法，在方法内部调用$\_ajax方法，接口为/cart/getlength，传入参数，根据后台接口返回数据给予用户提示，并将返回的数据添加到侧边栏购物车的标签中。将以下代码放到publicheader()这段代码的下一行即可；这个方法也是每个页面需要用到的。所以一起封装在header.js文件中。

function getCArtNumber(){

$\_ajax('post','/cart/getlength').then(res=>{

var divnumber = document.getElementsByClassName('cart-num')[0];

if( res.state ){

divnumber.innerText = res.number;

}else{

divnumber.innerText = 0;

}

})

}

getCArtNumber();

4.退出登录。

用户点击退出登录后，js调用后台接口/user/logout，后台返回数据后，跳转到登录页面。将以下代码放到header.js的末尾

document.getElementById('exit').onclick = function(){

$\_ajax('get','/user/logout',{})

.then(res=>{

location.href="/carts/register.html?type=login"

})

}

5.编写css。

在public.css公共样式文件中中添加以下样式美化页面。

.sidebar{

width: 200px;

position: fixed;

top: 20px;

text-align: center;

}

.sidebar li{

margin-top: 10px;

}

#sidebar-cart{

position: relative;

}

.cart-num{

width: 20px;

height: 20px;

border-radius: 50%;

text-align: center;

line-height: 20px;

position: absolute;

right: -20px;

top: -8px;

color: red;

}

@keyframes ani{

0%{

transform: rotate(0deg);

}

100%{

transform: rotate(360deg);

}

}

.loading-animate{

animation: ani 0.8s linear 0s infinite;

}

## 单元六 用户列表

### 学习目标

掌握DOM元素操作。完成用户列表。

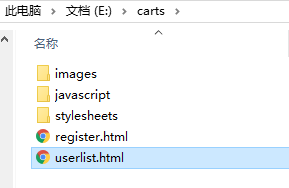
### 任务6.1 用户列表的页面

#### 任务描述

完成用户列表页面。

#### 任务实施

1. 在carts文件中新建userlist.html文件。



创建完成打开该文件，搭建基本框架，引入bootstrap.css和public.css文件，并引入ajax.js和header.js文件，代码如下。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>用户列表</title>

<link rel="stylesheet" href="stylesheets/bootstrap.css">

<link rel="stylesheet" href="stylesheets/public.css">

</head>

<body>

</body>

<script type="text/javascript" src="javascript/ajax.js"></script>

<script type="text/javascript" src="javascript/header.js"></script>

<script type="text/javascript">

//js代码1

</script>

</html>

2. 创建一个表单table标签。设置thead标签作为表头，添加一个tr标签，表示1行，在tr标签内放置3个td标签，表示3列，分别设置为 用户名、最近登录时间、操作。tbody作为表体，存放表格内容并添加id为table-tbody。

将以下代码放在body标签中。

<table class="table table-striped" style="width: 500px;margin:50px auto;">

<thead>

<tr>

<td>用户名</td>

<td>最近登录时间</td>

<td>操作</td>

</tr>

</thead>

<tbody id="table-tbody">

</tbody>

</table>

### 任务6.2 js交互

#### 任务描述

根据接口获取用户列表并展示，点击删除，弹出确认框，点击确定删除用户。

#### 任务实施

1．用户列表实现。创建格式化时间函数，传入时间戳返回年月日时分秒格式的字符串。在//js代码1的下一行写入如下代码

function formatTime(time){

var d = new Date(time);

return d.getFullYear() +'-'+(d.getMonth()+1)+"-"+d.getDate()+" "+ d.getHours()+':'+d.getMinutes()+":"+d.getSeconds()

}

2.调用$\_ajax方法获取用户列表数据，并渲染到table-tbody元素中。首先我们先请求数据，拿到数据后遍历数据，动态创建表单节点，把拿到的数据传入每个节点中。定义一个str变量，把创建好的节点和数据放到str变量中。获取表单中id为table-tbody的节点，把str变量的内容放到该节点内。

$\_ajax('get','/user/userall',{}).then(res=>{

var str = "";

for( var i = 0; i < res.content.length; i++ ){

str+="<tr>";

str+="<td>"+ res.content[i].username +"</td>";

str+="<td>"+ (!res.content[i].time ? "注册后未登录" : formatTime(res.content[i].time)) +"</td>";

str+="<td class='del' data-name="+ res.content[i].username +" data-id="+ res.content[i]["id"] +" data-type='del'>删除</td>";

str+="</tr>";

}

document.getElementById('table-tbody').innerHTML = str

})

//js代码2

3. 删除用户，第二步操作完成后，我们看到每个用户后面都有一个删除按钮，当用户点击删除的时候，需要弹出确认框，用户确认后发送请求。

我们给tbody添加点击事件。然后点击的时候，通过event对象判断点击的是否为删除按钮，是的话获取该标签的用户名和id值。然后弹出一个确认框，提示用户，用户点击确定后，调用$\_ajax方法，请求/user/del接口，将用户名和id作为参数传给后台。然后根据后台返回值判断，删除成功则将该节点移除并调用$\_hint方法提示用户删除成功。删除失败则根据后台返回的msg值，把msg值通过调用$\_hint方法提示用户。

代码如下，在//js代码2的下一行写入如下代码。

$\_getID('table-tbody').addEventListener('click',function(event){

// console.log(event)

if(event.target.getAttribute('data-type')==='del'){

var username = event.target.getAttribute('data-name');

var id = event.target.getAttribute('data-id');

if( confirm('确定删除用户' + username + '吗') ){

$\_ajax('post','/user/del',{username:username,id:id})

.then(res=>{

if(res.state){

$\_hint('删除成功');

event.target.parentNode.remove();

}else{

$\_hint(res.msg)

}

})

}

}

})

## 单元七 商品上传

### 学习目标

掌握formData对象，掌握onchange事件的用法，完成商品上传。

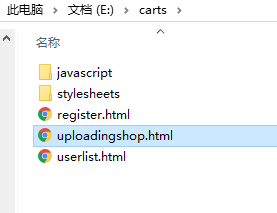
### 任务7.1 商品上传页面

#### 任务描述

完成商品上传页面。

#### 任务实施

1.在carts文件夹中新建uploadingshop.html文件。



创建完成后打开uploadingshop.html文件，搭建基本框架，引入bootstrap.css和public.css文件，并引入ajax.js和header.js文件，代码如下。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>上传商品</title>

<link rel="stylesheet" href="./stylesheets/bootstrap.css">

<link rel="stylesheet" href="./stylesheets/public.css">

</head>

<body>

</body>

<script src="./javascript/ajax.js"></script>

<script src="./javascript/header.js"></script>

<script>

//js代码1

</script>

</html>

1.1创建一个类名为container的div标签。包裹该页面所有内容。

在里面创建一个form表单。表单中设置7个div，每个div中都有一个盒子，用来存放商品图片、商品名称、价格、库存、商品状态、是否限购、（如果有限购，则添加）限购数量，还有一个盒子用来存放input标签。最后一个id为btn的div标签，作为按钮。

在form表单外，存放着一个id为showimg的div盒子。存放上传的图片。

<div class="container" style="position: relative;">

<form style="width: 600px; margin-top: 20px; margin-left: 100px;">

<div class="input-box clearfix">

<div class="col-xs-4">商品图片：</div>

<div class="col-xs-8"><input type="file" accept="image/gif, image/jpeg" id="img"></div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">商品名称：</div>

<div class="col-xs-8"><input id="shopname" class="input" type="text"></div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">价格：</div>

<div class="col-xs-8"><input id="shopprice" class="input" type="text"></div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">库存：</div>

<div class="col-xs-8"><input id="inventory" class="input" type="text"></div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">商品状态：</div>

<div class="col-xs-8" id="shop-state">

<label for="shop-normal">

正常<input type="radio" value="1" class="shop-state" checked name="state" id="shop-normanbsp;">

</label>

<label for="shop-del">

已下架<input type="radio" value="2" class="shop-state" name="state" id="shop-del">&nbsp;

</label>

</div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">是否限购：</div>

<div class="col-xs-8"><input id="islimit" class="checkbox" type="checkbox"></div>

</div>

<div class="input-box clearfix" style="display: none;" id="limit" >

<div class="col-xs-4">限购数量：</div>

<div class="col-xs-8"><input id="limitnumber" class="input" type="text" placeholder="请输入限购"></div>

</div>

<div id="btn" class="btn btn-lg btn-primary btn-block" style="margin: 30px auto 0;">提交</div>

</form>

<div class="img" id="showimg"></div>

</div>

2．添加样式，将以下代码添加到head标签中。

<style>

.img{

position: absolute;

top: 10px;

right: 0;

width: 400px;

height: 400px;

border: 1px solid #fff;

}

.input-box{

line-height: 32px;

text-align: right;

margin-top: 10px;

}

.input{

width: 80%;

border: 1px solid #fff;

border-radius: 4px;

outline: none;

}

.col-xs-8{

text-align: left;

}

.btn-block{

width: 80%;

}

</style>

### 任务7.2 js交互。

#### 任务描述

先完成图片上传功能，图片上传成功后后台接口将会返回图片网络路径，用户填写商品信息后，点击提交，将用户填写的信息和图片路径通过接口发送到后台。

#### 任务实施

1．上传图片。

创建FormData事件。定义imagePath变量=“”和shopState变量 =1 。

在uploadingshop.html文件中 //js代码1的下一行写入如下代码。

var formData = new FormData();

var imagePath="";

var shopState = 1;

给input文件上传按钮添加onchange事件，用户选择图片之后，当id为img的标签域内容改变时，触发onchange事件。用append方法将图片对象放到formData对象中。

创建 XMLRequest()对象。请求上传图片接口。在接口面前加上$\_http。使之变为网络路径。并设置请求头，将token值穿进去。请求成功后，执行回调函数，根据返回值进行判断。如果图片上传成功，则调用$\_hint函数提示用户，并将返回的图片路径复制给imagePath变量，并获取id为showimg的节点，将图片添加进去。

最后，将formData传给后台，代码如下。

$\_getID('img').onchange = function(event){

//console.log(event)

formData.append('filsde',event.target.files[0])

var xhr = new XMLHttpRequest();

xhr.open('post',$\_http+'/shop/imageupload')

xhr.setRequestHeader("token", localStorage.getItem('token'))

xhr.onreadystatechange = function(){

if( xhr.readyState === 4 && xhr.status === 200){

var data = JSON.parse(xhr.responseText)

console.log(data)

if( data.state ){

$\_hint('图片上传成功')

$\_getID('showimg').innerHTML = "<img width='400px' height='400px' src=" + $\_http+data.imagepath +" />"

imagePath = $\_http+data.imagepath;

}

}

}

xhr.send(formData)

}

2．上传图片功能完成后，我们先在carts中新建一个images文件夹用来存放项目中用到的图片，在images文件夹中放一张商品图片一会写静态页面时使用（任意图片都可以），我们测试一下，点击选择文件，选择一张图片，然后点击确认。如果出现如下页面则表示此功能已实现。



3．判断是否限购复选框是否被点击。如果勾上，则显示限购数量的盒子。

$\_getID('islimit').onclick = function(){

// console.log(event)

if(this.checked){

$\_getID('limit').style.display = "block"

}else{

$\_getID('limit').style.display = "none"

}

}

4．上传商品信息。给提交按钮注册点击事件，用户点击后，创建一个data对象，用来存放数据。

获取用户输入的图片路径、商品名称、商品价格、库存、状态。并传入对象data中。

判断是否限购选择框，限购的话，把限购的信息设置为true并传给data对象。并判断限购数量是否有输入，没输入则提醒用户。如果有输入则把输入的值传给data对象。没有限购则把限购的信息设置为false。

用户输入的信息，如果通过验证则开始上传，如果有数据不符合要求就弹出相关提示框。

$\_getID('btn').onclick = function(){

var data = {};

data.imagePath = imagePath;

data.shopName = $\_getID('shopname').value.trim();//商品名称

data.shopPrice = $\_getID('shopprice').value.trim()\*1;//商品价格

data.shopNum = $\_getID('inventory').value.trim()\*1;//库存

data.ShopState = shopState \* 1;//状态

if($\_getID('islimit').checked){

data.islimit = 'true';

if($\_getID('limitnumber').value.trim() === ""){

alert('请输入限购数量');

return;

}

data.limitNumber = $\_getID('limitnumber').value \* 1;

}else{

data.islimit = 'false';

}

$\_ajax('post','/shop/addshop',data).then(res=>{

if(res.state){

$\_hint('上传成功');

setTimeout(function(){

location.href = 'http://localhost/carts/shoplist.html'

},1500)

}

})

}

5.判断用户是否点击商品状态，默认是正常状态。如果点击了状态，则触发该事件。将点击的元素的value值复制给shopState变量。待用户点击上传按钮，再传入给data对象。并传给后台。

$\_getID('shop-state').onclick = function(event){

if(event.target.className === "shop-state"){

shopState = event.target.value;

}

}

## 单元八 商品列表

### 学习目标

掌握事件对象上常用的属性和方法。完成商品列表。

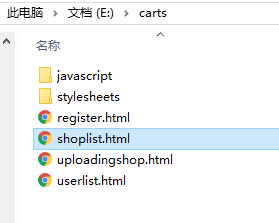
### 任务8.1 商品列表的页面

#### 任务描述

完成商品列表页面。

#### 任务实施

1.在carts文件夹中新建shoplist.html文件。



2. 根据商品列表页面编写静态页面，搭建基本框架，引入bootstrap.css和public.css文件，并引入ajax.js和header.js文件，代码如下。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>商品列表</title>

<link rel="stylesheet" href="./stylesheets/bootstrap.css">

<link rel="stylesheet" href="./stylesheets/public.css">

</head>

<body>

</body>

<script src="./javascript/ajax.js"></script>

<script src="./javascript/header.js"></script>

<script>

//js代码1

</script>

</html>

3. 创建<ul>标签，设置id为box。在里面放入一个<li>标签，这里每个<li>标签代表的是一条数据。里面放着3个div，利用bootstrap的栅格系统。

第一个div占1/6，放图片和一个该商品已下架的标签内容，设置类名为cover。

第二个div占1/3，放着商品名称、库存、价格、限购数量。

第三个div占1/2，放着编辑，加入购物车，删除3个按钮。这三个按钮利用了bootstrap的按钮样式，并携待id。

<ul class="box" id="box">

<li>

<div class="pull-left col-xs-2">

<div style="position:relative">

<img src="./images/卡卡西.jpg" alt="">

<div class="cover" style="display:none">

<div>该商品已下架</div>

</div>

</div>

</div>

<div class="pull-left col-xs-4">

<div>test1</div>

<div>库存：12</div>

<div>￥123</div>

<div style="color:red">该商品每人限购3件</div>

</div>

<div class="pull-left col-xs-6" style="min-height:100px;text-align:right">

<a href="/editshop.html?id=5e6b568752f7d8142c98408b" class="btn btn-primary addcart" data-id="5e6b568752f7d8142c98408b">编辑</a>

<button class="btn btn-primary addcart" data-id="5e6b568752f7d8142c98408b">加入购物车</button>

<button class="btn btn-primary delcart" data-id="5e6b568752f7d8142c98408b">删除</button>

</div>

</li>

</ul>

4．添加css样式，在<head>头部标签内，添加以下代码。

<style>

.box{

width: 1000px;

margin: 20px auto;

}

ul,ol{

list-style: none;

}

.box img{

max-width: 100%;

border-radius: 4px;

}

.box li{

margin-top: 8px;

background: #fff;

padding: 8px;

}

.box li:after{

content: "";

clear: both;

display: block;

}

.box .cover{

position: absolute;

width: 100%;

height: 100%;

z-index: 100;

background: rgba(0, 0, 0, 0.3);

top: 0;

left: 0;

text-align: center;

font-size: 18px;

color: #fff;

}

.box .cover > div{

position: absolute;

top: 50%;

transform: translateY(-50%);

width: 100%;

left: 0;

font-weight: 700;

color: #fff;

text-shadow: 2px 2px 1px black;

}

</style>

### 任务8.2 js交互。

#### 任务描述

获取所有商品，并将商品信息按照要求显示在网页上并且实现相关功能。

#### 任务实施

1．定义limit方法，参数为是否限购str，限购数量num。然后进行判断，限购的话，则返回一个<div>节点。没有限购则返回空。

在shoplist.html文件中 //js代码1的下一行写入如下代码。

function limit(str,num){

if( str === 'false' ){

return ""

}else{

return "<div style='color:red'>该商品每人限购"+num+"件</div>"

}

}

2．调用$\_ajax方法，请求/shop/shoplist接口。成功后执行回调函数。

2.1判断返回的内容是否为空。

不为空则定义一个空变量str。遍历后台返回的内容，并创建节点，将后台返回的内容一一填入节点中。并将这些内容追加到变量str中。获取类名为box的盒子，并将str的内容动态添加进去。

如果返回的内容为空。则获取类名为box的盒子，将 <h1 style="text-align:center">没有任何商品</h1> 这段信息添加进该盒子。

2.2注意：

图片路径前面需加$\_http。

判断状态是否为下线，用三元表达式，如果状态为正常，则隐藏“该商品已下架”节点，状态为下架，则显示“下架”节点。

限购数量节点：调用limit方法。把是否限购的状态值和限购数量传给该方法，如果有限购，那么在调用方法处返回节点。

将以下代码放到//js代码1的下一行。

$\_ajax('post','/shop/shoplist',{}).then(res=>{

console.log(res)

if( res.length !== 0 ){

var str = ""

for( var i = 0; i < res.length; i++ ){

str+=`

<li id="${res[i]['\_id']}">

<div class="pull-left col-xs-2" >

<div style="position:relative">

<img src="${res[i].imagePath}" alt="">

<div class="cover" ${res[i]['ShopState'] == 2 ? '' : 'style="display:none"'} >

<div>该商品已下架</div>

</div>

</div>

</div>

<div class="pull-left col-xs-4">

<div>${res[i].shopName}</div>

<div>库存：${res[i].shopNum}</div>

<div>￥${res[i].shopPrice}</div>

${

limit(res[i].islimit,res[i].limitNumber)

}

</div>

<div class="pull-left col-xs-6" style="min-height:100px;text-align:right">

<a href="/carts/editshop.html?id=${res[i]['\_id']}" class="btn btn-primary addcart" data-id="${res[i]['\_id']}">编辑</a>

<button class="btn btn-primary addcart" ${res[i]['ShopState'] == 2 ? 'disabled title="该商品已下架"' : ''} data-id="${res[i]['\_id']}">加入购物车</button>

<button class="btn btn-primary delcart" data-id="${res[i]['\_id']}">删除</button>

</div>

</li>`

}

document.getElementsByClassName('box')[0].innerHTML = str

}else{

document.getElementsByClassName('box')[0].innerHTML = '<h1 style="text-align:center">没有任何商品</h1>'

}

})

3．实现添加购物车功能和删除商品功能。

在这里我们给id为box的元素添加点击事件，通过事件冒泡来判断当前用户点击的是‘加入购物车’还是‘删除’按钮。

如果点击的是‘加入购物车’，首先获取该商品的id，然后调用/cart/joincart接口，将商品id传给后台。请求成功后，根据后台返回的信息提示用户‘添加成功’并调用getCArtNumber方法更新侧边栏购物车右上角的值。请求失败则将返回的信息提示用户。

如果点击的是‘删除’，首先获取该商品的id，然后弹出一个‘确定要删除当前商品吗？’的对话框，如果点击取消，则不作任何操作。点击确定则调用$\_ajax方法请求，调用/shop/del接口，将商品id传给后台。请求成功后，将后台中该id的商品移除。并提示用户‘删除成功’。

在//js代码2的下一行添加以下代码。

$\_getID('box').onclick = function(event){

if( event.target.classList.contains('addcart')){

console.log(event.target.getAttribute('data-id'));

var id = event.target.getAttribute('data-id');

$\_ajax('post','/cart/joincart',{id:id}).then(res=>{

if( res.state ){

getCArtNumber()

$\_hint('添加成功')

}else{

$\_hint(res.msg)

}

})

}else if( event.target.classList.contains('delcart') ){

var id = event.target.getAttribute('data-id');

if( !confirm('确定要删除当前商品吗？') ){

return ;

}

$\_ajax('get','/shop/del',{id:id}).then(res=>{

if( res.state ){

$\_getID(id).remove(true);

$\_hint('删除成功')

}

})

}

}

## 单元九 购物车

### 学习目标

掌握javascript的综合运用。完成购物车页面和功能。

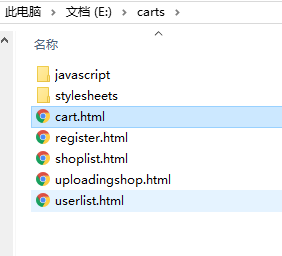
### 任务9.1 购物车的页面、css

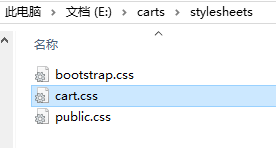
#### 任务描述

完成购物车页面。

#### 任务实施

1.在carts文件夹中新建cart.html文件夹，创建完成后打开该文件。搭建基本框架，创建cart.css文件。





并引入bootstrap.css、public.css和cart.css文件，并引入ajax.js和header.js文件，代码如下。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>购物车</title>

<link rel="stylesheet" href="./stylesheets/bootstrap.css">

<link rel="stylesheet" href="./stylesheets/public.css">

<link rel="stylesheet" href="./stylesheets/cart.css">

</head>

<body>

</body>

<script src="./javascript/ajax.js"></script>

<script src="./javascript/header.js"></script>

<script>

//js代码1

</script>

</html>

2.在<body>标签中放置默认内容。创建一个类名为 box 的<div>标签。再创建一个div来包裹着所有内容并设置样式。

先创建首行的div。全选按钮、商品信息、数量、金额、操作。

然后创建内容的div，设置选择按钮，商品信息下放一张图片、商品名称、限购数量、库存。数量下放一个数量，左右放着一个 + 和 – 的按钮。操作放着删除操作。

代码如下。

<div class="box">

<div style="width: 800px;background: #fff;">

<div class="shop-title clearfix">

<div class="all public-checkbox">

<input id="checkbox-all" type="checkbox">

<label for="checkbox-all">

全选

</label>

</div>

<div class="shop-info public-info">

商品信息

</div>

<div class="shop-number public-number">

数量

</div>

<div class="shop-price public-price">

金额

</div>

<div class="shop-option public-option">

操作

</div>

</div>

<div id="cart-list">

<ul class="box1 clearfix">

<li class="public-checkbox"><input data-id="5e7eda2c2855590be22e82d1" class="shop-checked" type="checkbox"></li>

<li class="public-info clearfix">

<div class="shop-image fl">

<img src="http://49.235.117.27:3000/shops/upload\_cf0f4f858618b9dc2064ad13800366df.jpg" alt=""></div>

<div class="fl public-detail">

<a href="" class="shop-detail-title">test1</a>

<p class="limit-number">库存123件</p>

</div>

</li>

<li class="public-number">

<div class="box-number clearfix" data-shopid="5e7eda2c2855590be22e82d1">

<span class="fl reduce">-</span>

<input value="1" class="fl" type="text">

<span class="fl add">+</span>

</div>

</li>

<li class="public-price">123</li>

<li class="public-option J\_del\_shop" data-id="5e7eda2c2855590be22e82d1" data-name="test1">删除</li>

</ul>

</div>

<div id="disabledShop"></div>

<div class="shop-total-info">

<div><input id="checkbox-all1" type="checkbox">全选</div>

<div class="">

已选<span id="totlaNumber">0</span>件商品

</div>

<div>

总价<span id="totlePrice">0.00</span>

</div>

</div>

</div>

</div>

3．在stylesheets文件夹中，创建cart.css文件，打开cart.css文件，设置页面内容样式，将以下代码写入该文件中。

\*{

box-sizing: border-box;

}

.colorf40{

color: #f40;

}

.box{

padding-left: 350px;

margin: 0 auto;

}

.shop-title > div{

float: left;

}

.shop-title input{

vertical-align: text-bottom;

}

.public-checkbox{

width: 10%;

padding-left: 10px;

}

.public-info{

width: 50%;

}

.public-number{

width: 17%;

}

.public-price{

width: 13%;

}

.public-option{

width: 10%;

}

.box1{

min-height: 100px;

border-top: 1px solid lightgray;

}

.cart-null{

font-size: 30px;

color: lightgray;

text-align: center;

line-height: 100px;

}

.box1 .shop-image{

width: 30%;

border: 1px solid lightgray;

}

.box1 .shop-image img{

width: 100%;

}

.public-detail{

width: 70%;

padding-left: 10px;

}

.public-detail a{

display: block;

}

.box1{

padding: 10px 0;

}

.box1 > li{

float: left;

}

.box-number span{

width: 30px;

height: 30px;

border: 1px solid lightgray;

line-height: 30px;

text-align: center;

cursor: pointer;

}

.box-number input{

width: 40px;

height: 30px;

text-align: center;

}

.limit-number{

font-size: 12px;

color: #9c9c9c;

}

.shop-total-info{

position: fixed;

bottom: 0;

background: #e5e5e5;

z-index: 99;

left: 350px;

width: 800px;

height: 50px;

padding: 10px 0 ;

}

.shop-total-info > div{

float: left;

width: 33.333%;

}

#disabledShop{

border-top: 1px solid lightgray;

}

#disabledShop .box-number span{

cursor:no-drop;

}

#disabledShop .box-number input{

cursor:no-drop;

}

.public-option{

cursor: pointer;

}

### 任务9.2 js交互

#### 任务描述

这个单元是这个教程对综合知识的运用要求最高的部分，这章我们需要完成以下功能：商品全选、取消全选、点击选中商品计算价格、点击增加或减少商品计算价格及对商品数量的判断，接下来我们就要开始完成这些功能。

#### 任务实施

1．清除id为cart-list中的html代码，这部分和id为disabledShop元素中的html均是根据后台数据生成html代码，接下来我们就要开始编写这一部分代码。

2．定义3个全局变量，cart Timer、global\_cartShop、global\_checkedCartId。

3．调用$\_ajax函数调用 cart/cartlist接口。请求成功后，执行.then回调函数。

3.1判断是否请求成功和返回的内容是否不为空，不为空则执行下面的操作。

3.2 定义data变量存放返回的内容。定义空变量validShop。

3.3创建一个‘放着下架商品或者删除的商品’的盒子。

3.4遍历获得的内容，将所有内容添加到global\_cartShop变量中，动态生成商品信息。

3.5创建完之后，将正常的商品信息放置validShop变量中，有下架的或者被删除的商品放置上面步骤3.3创建的盒子中。

3.6将这两个变量放置页面中相对应的地方。

注意：

1、根据商品状态值判断商品是否下架或者被删除，如果是的话，则前面的按钮设置为不可操作。

2、根据携带的islimit值判断商品是否限购，限购的话需要在创建存放加减数量的盒子上设置data- limitnumer用来放置限购数量。没有则不需要。每次放置完需把str里面的内容清空，再继续遍历添加到其中。

在cart.html文件中 //js代码1的下一行写入如下代码。

var cartTimer;

var global\_cartShop = [];

var global\_checkedCartId = [];

$\_ajax('get', '/cart/cartlist', {}).then(res => {

console.log(res.content)

if (res.state && res.content.length !== 0) {

var data = res.content;

var validShop = "";

var disabledShop = "<div style='color:red;padding:15px 0'>以下商品已经下架或者已经被删除</div>";

var str = "";

for (var i = 0; i < data.length; i++) {

data[i].ShopState == '1' ? global\_cartShop.push(data[i]) : '';

str += '<ul class="box1 clearfix">'

str += '<li class="public-checkbox">';

if (data[i].ShopState == '1') { //商品状态 1：正常 2：下架 3：删除

str += '<input data-id="' + data[i]['\_id'] + '" class="shop-checked" type="checkbox">';

} else {

str += '<input type="checkbox" disabled title="该商品已下架">';

}

str += '</li>';

str += '<li class="public-info clearfix">'

str += '<div class="shop-image fl">'

str += '<img src="' + data[i].imagePath + '" alt="">'

str += '</div>'

str += '<div class="fl public-detail">'

str += '<a href="" class="shop-detail-title">' + data[i].shopName + '</a>';

if (data[i].islimit === "true") {

str += '<p style="padding-top:10px" class="limit-number">该商品限购<span class="colorf40">' + data[i].limitNumber + '</span>件</p>'

}

str += '<p class="limit-number">库存' + data[i].shopNum + '件</p>'

str += '</div>'

str += '</li>'

str += '<li class="public-number">'

if (data[i].islimit === "true") {

str += '<div class="box-number clearfix" data-limitnumer="' + data[i]['limitNumber'] + '" data-shopid="' + data[i]['\_id'] + '">'

} else {

str += '<div class="box-number clearfix" data-shopid="' + data[i]['\_id'] + '">'

}

str += '<span class="fl reduce">-</span>'

str += '<input ' + (data[i].ShopState == '2' ? 'disabled' : '') + ' value="' + data[i].number + '" class="fl" type="text">'

str += '<span class="fl add">+</span>'

str += '</div>'

str += '</li>'

str += '<li class="public-price">' + data[i].shopPrice + '</li>'

str += '<li class="public-option J\_del\_shop" data-id="' + data[i]['\_id'] + '" data-name="' + data[i].shopName + '">删除</li>'

str += '</ul>';

if (data[i].ShopState === 1) {

validShop += str;

} else {

disabledShop += str;

}

str = "";

}

$\_getID('cart-list').innerHTML = validShop;

$\_getID('disabledShop').innerHTML = disabledShop;

}

})

4．我们接下来要完成的是点击两个全选实现选中全部商品。

4.1 先给全选按钮设置点击事件。将页面上所有类名为shop-checked的标签全部赋值给遍历input。（这是每个商品的选择按钮）

4.2 定义空数组global\_checkedCartId。获取该全选按钮的状态值，用于下面的判断。

4.3根据选框数组的长度，去遍历所有选框，使之选择按钮勾上。

4.4如果全选框选中时，则根据商品长度遍历所有商品，获取所有商品的id，赋值给global\_checkedCartId。否则global\_checkedCartId的值为空。

4.5给下方全选按钮注册点击事件，点击之后调用上面全选按钮的点击事件。

//点击全选按钮，实现选中所有商品

$\_getID('checkbox-all').onclick = function (event) {

console.log(event)

var input = document.getElementsByClassName('shop-checked');//根据类名获取所有选框

var flag = event.target.checked;

global\_checkedCartId = [];

$\_getID('checkbox-all1').checked = flag;

for (var i = 0; i < input.length; i++) { //根据选框数组的长度，去遍历所有选框，使之选择上

input[i].checked = flag;

}

if (flag) { //如果全选框选中时，则获取所有商品的id，赋值给global\_checkedCartId

for (var i = 0; i < global\_cartShop.length; i++) {

global\_checkedCartId.push(global\_cartShop[i]['\_id']);

}

} else {

global\_checkedCartId = [];

}

imputedPrice();

}

//点击下方的全选按钮，调用上面的按钮的方法，实现全选。

$\_getID('checkbox-all1').onclick = function () {

$\_getID('checkbox-all').click();

}

5．实现点击全选后我们还要实现在选中全部商品的时候两个全选按钮也要被选中。

5.1 选中商品则把商品id赋值给变量id。

5.2 如果勾选上，则把该变量id添加到global\_checkedCartId数组中。

如果取消勾选，则把该变量从global\_checkedCartId数组中删除。

5.3 调用imputedPrice方法。

以下代码可以实现上述功能

//给商品列表添加点击冒泡事件。

$\_getID('cart-list').addEventListener('click', function (event) {

if (event.target.classList.contains('shop-checked')) { //如果选中的是商品的选框时

var id = event.target.getAttribute('data-id') //获取该商品的id

if (event.target.checked) {

global\_checkedCartId.push(id); //勾选上则将该商品的id添加到 global\_checkedCartId 数组中

} else {

var index = global\_checkedCartId.indexOf(id);//获取id的起始位置

global\_checkedCartId.splice(index, 1);//未选中则从商品id的数组中删除掉该id对应的商品

}

imputedPrice();

}

})

6．我们在点击选中商品的时候会发现在控制台有一个报错信息，提示‘imputedPrice is not defined’，这个函数是计算商品价格的函数，为了消除这个错误，我们现在实现这个功能。

6.1 定义一个price的变量。获取商品的长度和商品id的长度。

6.2判断，如果商品的长度或商品id的长度为0，则把总数量和总价格设置为0。两者长度不为0则for循环遍历商品id的长度，再在里面嵌套一个for循环遍历商品的长度。如果遍历是商品id等于商品的属性id时，将该商品的数量和价格相乘，把结果赋值给变量price。

6.3当商品的长度跟商品id的长度相等时，则全部选中，把两个全选按钮勾上。如果不相等则取消全选按钮的选择。

6.4把得到的商品id的长度赋值给总数量，把得到的price变量中的值赋值给总价格。（总价格取小数点后两位）

代码如下

//计算商品价格函数

function imputedPrice() {

var price = 0;

var cartShopLength = global\_cartShop.length;

var cartIdLength = global\_checkedCartId.length;

if (cartShopLength === 0 || cartIdLength === 0) {

$\_getID('totlaNumber').innerHTML = 0;

$\_getID('totlePrice').innerHTML = '0.00';

return;

}

for (var j = 0; j < cartIdLength; j++) {

for (var i = 0; i < cartShopLength; i++) {

if (global\_checkedCartId[j] === global\_cartShop[i]['\_id']) {

price += global\_cartShop[i].shopPrice \* global\_cartShop[i].number

continue;

}

}

}

if (cartShopLength === cartIdLength) {

$\_getID('checkbox-all').checked = true;

$\_getID('checkbox-all1').checked = true;

} else {

$\_getID('checkbox-all').checked = false;

$\_getID('checkbox-all1').checked = false;

}

$\_getID('totlaNumber').innerHTML = cartIdLength;

$\_getID('totlePrice').innerHTML = price.toFixed(2);

}

7．接下来我们来编写点击增加减少商品时的逻辑。

7.1首先判断是否点击了增加或者减少按钮。是则获取该节点的父元素。取该父元素的商品id赋值给变量id、第一个input框（即是中间的数值的盒子）赋值给变量input、获取input的值赋值给变量number、获取限购数量赋值给limitNumber。

7.3判断，如果是点击减少，则当number的值为1的时候，则提示用户商品不能再减少了。当number的值大于1的时候，则number--。

7.4判断，如果是点击增加，判断是否有限购数量，如果有的话，则比较number和限购数量limitNumber。两者相等则提醒用户，该商品限购的数量。其他情况则number++。

7.5点击两个增加或者减少按钮后，则需要将number值重新给页面中的input框赋值。

7.6设置一个定时器用来请求后台。在每次发生改变时清除之前的定时器，再启用该定时器。调用$\_ajax函数请求/cart/changeNumber接口，将id 和number传给后台。请求成功后，则把对应的商品数量重新修改。并重新调用计算函数imputedPrice()。修改成功后根据返回值提示用户。

//添加 + - 事件

$\_getID('cart-list').addEventListener('click', function (event) {

if (event.target.classList.contains('reduce') || event.target.classList.contains('add')) {

var parent = event.target.parentNode;

var id = parent.getAttribute('data-shopid');

var input = parent.getElementsByTagName('input')[0];

var number = input.value;

var limitNumber = parent.getAttribute('data-limitnumer');//限购数量

if (event.target.classList.contains('reduce')) {

if (number == 1) {

$\_hint('商品不能再减少了');

return;

}

number--;

} else {

if (limitNumber) {

if (limitNumber == number) {

$\_hint('该商品每人限购' + limitNumber + '件');

return;

}

}

number++;

}

input.value = number;

clearTimeout(cartTimer)

cartTimer = setTimeout(function () {

$\_ajax('post', '/cart/changeNumber', { id: id, number: number }).then(res => {

if (res.state) {

console.log(res)

for (var i = 0; i < global\_cartShop.length; i++) {

if (global\_cartShop[i]['\_id'] === id) { //当商品的id和修改的商品的id一致时

global\_cartShop[i].number = number;

break;

}

}

imputedPrice();

}

$\_hint(res.msg)

})

}, 200)

}

})

8．删除商品。点击删除按钮时弹出对话框，是否删除xxx商品，点击确定则执行delShop（）函数，并把该商品的信息作为参数传进去。

8.1delShop（）函数，有一个参数el。首先获取该节点的商品id赋值给变量id。

8.2调用$\_ajax函数请求/cart /delShop接口，将id传给后台。请求成功后，遍历商品信息，商品信息中的id与该节点的商品id对应时，将后台数据中的这一条商品信息删除。遍历结束后，把该节点的 父元素用remove()方法从页面中删除掉。调用getCArtNumber()，更新购物车商品数量

8.3如果在商品id数组中找不到该删除的商品的id，则不作任何操作。如果找到了。则将它从商品id数组中去除。再调用imputedPrice()。

8.4 修改完成后根据返回值提示用户。

代码如下。

//删除按钮事件

$\_getID('cart-list').addEventListener('click', function (event) {

if (event.target.classList.contains('J\_del\_shop') && confirm('确定删除商品‘' + event.target.getAttribute('data-name') + '’?')) {

delShop(event.target)

}

})

$\_getID('disabledShop').addEventListener('click', function (event) {

if (event.target.classList.contains('J\_del\_shop') && confirm('确定删除商品‘' + event.target.getAttribute('data-name') + '’?')) {

delShop(event.target)

}

})

function delShop(el) {

var id = el.getAttribute('data-id')

$\_ajax('post', '/cart/delShop', { id: id }).then(res => {

if (res.state) {

for (var i = 0; i < global\_cartShop.length; i++) {

if (global\_cartShop[i]['\_id'] === id) {

global\_cartShop.splice(i, 1)

break;

}

}

el.parentNode.remove(true);

getCArtNumber();//更新购物车商品数量

(global\_checkedCartId.indexOf(id) === -1 ? '' : global\_checkedCartId.splice(global\_checkedCartId.indexOf(id), 1))

imputedPrice();

}

$\_hint(res.msg)

})

}

## 单元十 编辑商品

### 学习目标

了解页面之间传值方式。完成编辑商品页面。

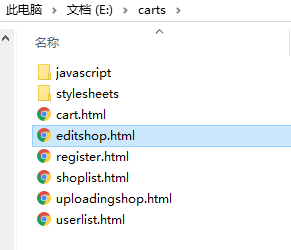
### 任务10.1 编辑商品页面

#### 任务描述

完成编辑商品页面。

#### 任务实施

1. 在carts文件夹中新建editshop.html文件夹。



创建完成后打开该文件。搭建基本框架，并引入bootstrap.css和public.css文件，并引入ajax.js和header.js文件，代码如下。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>编辑商品</title>

<link href="./stylesheets/bootstrap.css" rel="stylesheet">

<link rel="stylesheet" href="./stylesheets/public.css">

</head>

<body>

</body>

<script src="./javascript/ajax.js"></script>

<script src="./javascript/header.js"></script>

<script>

//js代码1

</script>

</html>

2．在<body>标签中，放入两个div，一个存放编辑的内容。一个存放图片。第一个盒子中存放上传图片按钮和一个提交按钮。其他内容后面动态创建。代码如下。

<div class="container" id="box">

<form style="width: 600px;margin-top: 20px;margin-left: 100px;">

<div class="input-box clearfix">

<div class="col-xs-4">商品图片：</div>

<div class="col-xs-8"><input type="file" accept="image/gif, image/jpeg" id="img"></div>

</div>

<div id="btn" class="btn btn-lg btn-primary btn-block" style="margin-top: 30px;">提交</div>

</form>

</div>

<div class="img" id="showimg"></div>

3．给页面中的元素设置样式，将以下代码放置<head>标签内。

<style>

.img{

position: absolute;

top: 10px;

right: 70px;

width: 400px;

height: 400px;

border: 1px solid #fff;

}

.input-box{

line-height: 32px;

text-align: right;

margin-top: 10px;

}

.input{

width: 80%;

border: 1px solid #fff;

border-radius: 4px;

outline: none;

}

.col-xs-8{

text-align: left;

}

</style>

### 任务10.2 js交互

#### 任务描述

在商品列表页面点击编辑按钮，携带商品id跳转到编辑商品页面。编辑商品页面根据商品id获取商品信息，用户点击提交的时候将商品传给后台。

#### 任务实施

1．定义一个FormData对象，一个imagePath空变量，和一个shopState = 1。

2．定义GetQueryString（name）方法，将地址栏上的信息获取到。

3．在location对象上获取商品id，赋值给变量shopid。

4．判断，如果shopid存在则调用$\_ajax请求/shop/queryshopinfo接口，根据id向后台获取详细商品详细信息。根据返回值，调用cleardata(data)方法，将返回的商品信息作为参数传给该方法。创建商品信息内容

5．将商品的图片放在图片显示区域。

在editshop.html文件中 //js代码1的下一行写入如下代码

var formData = new FormData();

var imagePath="";

var shopState = 1;

//获取url的值

function GetQueryString(name){

var reg = new RegExp("(^|&)"+ name +"=([^&]\*)(&|$)"); //取以name开头或者以&开头以&或者以$结尾的字符串

var r = window.location.search.substr(1).match(reg); //找出?之后匹配reg的字符串

if(r!=null)return unescape(r[2]); return null; //不为空则对所选的字符串进行解码

}

var shopid = GetQueryString('id'); //获取url地址中的id值

if( shopid ){

$\_ajax('get','/shop/queryshopinfo',{id:shopid}).then(res=>{

console.log(res[0]);

cleardata(res[0]); //调用下面的方法，创建商品信息内容

document.getElementById('showimg').innerHTML="<img width='400px' height='400px' src="+ res[0].imagePath +">"

//将图片放置图片盒子中

})

}

6． 将后台返回的商品信息，使用模板字符串拼接，将拼接好的字符串放在btn按钮前，显示到页面上。

//将后台返回的商品信息，拼接后，显示到页面上

function cleardata(data){

str = ` <div class="input-box clearfix">

<div class="col-xs-4">商品名称：</div>

<div class="col-xs-8">

<input id="shopname" class="input" type="text" value="${data.shopName}">

</div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">价格：</div>

<div class="col-xs-8">

<input id="shopprice" class="input" type="text" value="${data.shopPrice}">

</div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">库存：</div>

<div class="col-xs-8">

<input id="inventory" class="input" type="text" value="${data.shopNum}">

</div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">商品状态：</div>

<div class="col-xs-8" id="shop-state">

<label for="shop-normal">

正常

<input type="radio" value="1" class="shop-state" ${ data.ShopState == 1 ? 'checked' : '' } name="state" id="shop-normal">&nbsp;

</label>

<label for="shop-del">

已下架

<input type="radio" value="2" ${ data.ShopState == 2 ? 'checked' : '' } class="shop-state" name="state" id="shop-del">&nbsp;

</label>

</div>

</div>

<div class="input-box clearfix">

<div class="col-xs-4">是否限购：</div>

<div class="col-xs-8">

<input id="islimit" ${ data.islimit == "true" ? 'checked' : '' } class="checkbox" type="checkbox">

</div>

</div>

<div class="input-box clearfix" style="display: ${ data.islimit == "true" ? 'block' : 'none' };" id="limit" >

<div class="col-xs-4">限购数量：</div>

<div class="col-xs-8">

<input id="limitnumber" value="${ data.islimit ? data.limitNumber : 0 }" class="input" type="text" placeholder="请输入限购数量">

</div>

</div>`

$\_getID('btn').insertAdjacentHTML('beforebegin',str) //将创建好的内容放在该节点前方。

}

7．根据是否限购控制限购选项的显示与隐藏。

判断点击的按钮id是否等于islimit（是否限购的按钮id），如果否，则return。如果是，则判断按钮是否选择上。勾上则将id名为limit的盒子显示，未勾上则隐藏。

// 根据是否限购，设置按钮的状态

$\_getID('box').addEventListener('click', function(event){

if( event.target.id !== 'islimit' ){

return

}

if( event.target.checked ){

$\_getID('limit').style.display="block"

}else{

$\_getID('limit').style.display="none"

}

})

8．获取商品状态。将点击的商品状态的值赋值给变量shopState。

// 获取商品状态

$\_getID('box').addEventListener('click', function(event){

if( event.target.className==="shop-state" ){

shopState = event.target.value;

}

})

9．上传商品图片。

9.1 当id名为img的盒子发生变化时，把上传的图片添加到fromData对象中。

9.2创建XMLHttpRequest对象，请求/shop/imageupload接口，但是因为这里是网络图片，所以前面必须加上$http；

9.3因为是post请求，所以在请求头设置token值。

9.4执行回调函数，请求成功则提醒用户‘图片上传成功’，并将图片放在图片显示区域。将返回的图片路径赋值给imagePath变量。

9.5最后发送http请求，将formData作为参数传进去。

//上传商品图片

$\_getID('img').onchange = function(event){

formData.append('filsde',event.target.files[0])

var xhr = new XMLHttpRequest();

xhr.open('post',$\_http+'/shop/imageupload')

xhr.setRequestHeader("token", localStorage.getItem('token'))

xhr.onreadystatechange = function(){

if( xhr.readyState === 4 && xhr.status === 200){

var data = JSON.parse(xhr.responseText)

if( data.state ){

$\_hint('图片上传成功')

$\_getID('showimg').innerHTML = "<img width='400px' height='400px' src=" + $\_http+data.imagepath +" />"

imagePath = $\_http+data.imagepath;

}

}

}

xhr.send(formData)

}

10． 点击提交将修改好的商品信息提交给后台。

10.1给提交按钮注册点击事件，用户点击后，创建一个data对象，用来存放数据。

10.2获取用户输入的图片路径、商品名称、商品价格、库存、状态。并传入对象data中。

10.3判断是否限购选择框，限购的话，把限购的信息设置为true并传给data对象。并判断限购数量是否有输入，没输入则提醒用户。如果有输入则把输入的值传给data对象。没有限购则把限购的信息设置为false。

10.4用户输入的信息，如果通过验证则开始上传，如果有数据不符合要求就弹出相关提示框。

//点击提交将修改好的商品信息提交给后台。

$\_getID('btn').onclick = function(){

var data = {};

data.imagePath = imagePath;

data.shopName = $\_getID('shopname').value.trim();

data.shopPrice = $\_getID('shopprice').value.trim() \* 1;

data.shopNum= $\_getID('inventory').value.trim() \* 1;

data.ShopState = shopState \* 1;

if( $\_getID('islimit').checked ){

data.islimit = 'true';

if( $\_getID('limitnumber').value.trim() === '' ){

alert('请输入限购数量')

return;

}

data.limitNumber = $\_getID('limitnumber').value \* 1;

}else{

data.islimit = 'false';

}

data.id = shopid;

$\_ajax('post','/shop/updateShop',data).then(res=>{

if( res.state ){

$\_hint('修改成功');

setTimeout(function(){

location.href = 'http://localhost/carts/shoplist.html'

},1500)

}

})

}