一、安装Vue-cli(脚手架)工具

1. npm i –g vue-cli

2. 初始化一个vue-cli项目 版本2.XX 在版本3.0+ 用 vue create #项目名称

(1) npm install –g @vue/cli-init

(2) vue init webpack 项目名称

(3)?Project name(项目名称)

(4) ? Project description (A Vue.js project) 项目描述

(5) ? Author (Gavin1997 <15108466849@163.com>) 作者

(6) ? Vue build (Use arrow keys) #webpack配置文件 选+

(7) ? Install vue-router? (Y/n) 是否使用vue-router 选Y

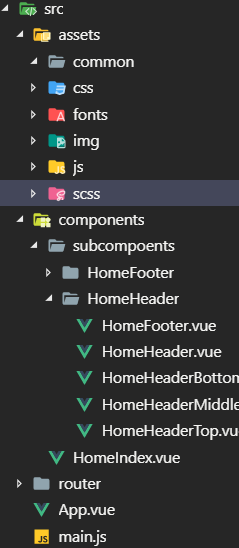
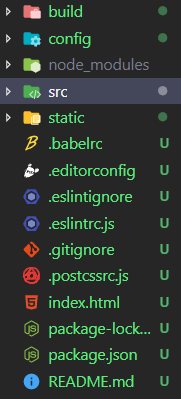
(8)? Use ESLint to lint your code? (Y/n) 是否使用Eslint检查语法 N

(9) ? Pick an ESLint preset (Use arrow keys) 选 Standard

(10) ? Set up unit tests (Y/n) 单元测试 N

(11) ? Setup e2e tests with Nightwatch? (Y/n) N

(12) ? Should we run `npm install` for you after the project has been created? (recommended) (Use arrow keys) 选use NPM



在Vue项目中使用scss 必须安装下面三个插件

1. npm install node-sass --save-dev

npm install sass-loader --save-dev

npm install style-loader --save-dev

我们想要全局引用scss 需要安装

npm install sass-resources-loader --save-dev

在build/utils.js文件中修改配置，把scss: generateLoaders('sass')改为：

return {

css: generateLoaders(),

postcss: generateLoaders(),

less: generateLoaders('less'),

sass: generateLoaders('sass', { indentedSyntax: true }),

scss: generateLoaders('sass').concat({

loader:'sass-resources-loader',

options: {

resources:path.resolve(\_\_dirname,'../src/assets/scss/\_color.scss')

}

}),

stylus: generateLoaders('stylus'),

styl: generateLoaders('stylus')

}

}

**2.图片引用的路径问题**

资源里的背景图片,不像index.html 中加载资源一样,../../assets/images/logo.png 会被相对打包成static/img,所以要保留css引用图片的正常路径

在build下的uils.js中

添加公共路径

if (options.extract) {

return ExtractTextPlugin.extract({

use: loaders,

fallback: 'vue-style-loader',

publicPath:'../../../',

})

} else {

return ['vue-style-loader'].concat(loaders)

}

}

**3.各种问题之大坑**

**1. 跨域问题:在服务器端** 引入 cors模块 专门处理跨域问题

注意引入位置 在创建服务器之后 在监听端口之前

const prodcuts = require("./routes/products");

var app = express();

//加载处理跨域模块

const cors = require("cors");

//允许那个地址跨域访问

app.use(cors({

origin: 'http://127.0.0.1:8080',

credentials: true

}));

var server = app.listen(3000);

2. **在vue-cli中 main.js 配置axios** credentials 如果用到session 因为axios默认不携带cookie,所有在 main.js中

//配置axios

*Vue*.prototype.$http = axios;

//基本访问路径

axios.defaults.baseURL = "http://127.0.0.1:3000/";

//设置axios session的credentials

axios.defaults.withCredentials = true;

**3. 在vue-cli2.x中 vue-router跳转后滚轮位置不为0保持不变的bug解决办法**

在app.vue中

export default {

name: 'App',

watch:{

'$route':function(*to*,*from*) {

document.body.scrollTop = 0;

document.documentElement.scrollTop = 0;

}

}

}

**4.当前css高亮的解决办法,不用重新刷新页面,直接写一个简单的逻辑判断**.(在同一个v-for循环中)

<li *v-for*="(c,i) of res.categories" *:class*="{booking\_active:changedActive==i}" *@click.prevent*="changeDetails(c.tid,i)">

<a>{{c.category}}</a>

</li>

自定义方法单击事件 将tid和下标传入 class属性为true的时候显示. data(reutrn{})中 changeActive默认为0;

changeDetails(*tid*,*i*) {

*console*.log(*i*)

*this*.$router.push("/product\_details/" + *tid*);

*this*.$http.get("details?tid=" + *tid*).then(*res* => {

*this*.res = res.data;

});

*this*.changedActive=*i*;

**5. 当前css高亮的解决办法,不用重新刷新页面,直接写一个简单的逻辑判断**(**不在同一个v-for循环中**)

在当前页面的data(return{})中 自定义一个数字n,默认为0;

每次点击下一页的时候 将这个n++; 上一页的时候 n--;

然后在watch中监听从后台拿到的pno(当前页面)

watch:{

pno(){

if(*this*.pno = *this*.n){

*this*.changedActive = *this*.n;

}else{

*this*.changedActive = "";

}

}

},

**解决当前页面的按钮为高亮状态问题: v-for(item,i) @click=”nowNum(i)”**

1. 在上一页下一页的事件中

pno = 1; n=1;

让this.changedActive = this.n-1;//因为this.changedActice与下标相同

2. 在按钮绑定的事件 nowNum(i)中让

//点击当前页

now\_num(*i*) {//0,1,2,3

*this*.pno = *i*+1;//1,2,3,4

*this*.changedActive = *i*; //控制变化的变量=下标 让其高亮

*this*.n = *i*+1; //控制与pno之间搭建起来的关系 让其高亮

if(*i*>=0 &&*i*<*this*.pageCount){

*this*.disabled\_info\_next="";

*this*.disabled\_info\_prev="";

}

**5.1 分页当前页高亮和下一页上一页等高亮解决方法**

<button *@click*="prev()" *:class*="disabled\_prev\_info">上一页</button>

<button *v-for*="(item,i) pageCount" *:class*="{active:changedActive==i}" *@click*="num(i)">{{i+1}}</button>

<button *@click*="next()" *:class*="disabled\_next\_info">下一页</button>

**1.在next中**

next() {

if (*this*.kw != "") {

*this*.pno++;

if (*this*.pno >= *this*.pageCount && *this*.pno > 0) {

Toast({

message: "已经是最后一页了哦,没有更多了,输入别的看看吧",

duration: 1000

})

*this*.pno = *this*.pageCount - 1;

*this*.changedActive = *this*.pno;

*this*.disabled\_next\_info = "disabled"

} else {

*this*.disabled\_prev\_info = "active"

*this*.getMsg();

}

**2. 在num中**

num(*i*) {

*this*.pno = i;

*this*.getMsg();

}

**3. 监听watch {} 当前页数 让当前控制高亮的变量等于当前页数**

pno() {

*this*.changedActive = *this*.pno;

}

**6.实现一个简单的轮播图**

<li><img *:src*="imgArray[nowindex]"*alt*=""*style*="height:420px;width:100%; " *v-if*="ifshow"></li>

1 .img的图片列表地址保存在data(return{})中 一个数组; 设置v-if=”ifshow” 默认为true

nowindex:0; imgArry:[]

2. 在mounted里设置一个定时器

mounted() {

setInterval(() => {

*this*.nowindex++

if (*this*.nowindex === *this*.imgArray.length) {

*this*.nowindex = 0;

} else {

*this*.nowindex + 1

}

}, 2000)

},

每两秒让nowindex+1,当它等于获取图片列表数组的长度时,就代表播放完毕,让其等于0;

小圆点高亮问题:

span *v-for*="(item,index) in imgArray" *:class*="{on:index===nowindex}" *class*="carousel\_circle ml-2"*@click*="goto(index)"></span>

个数根据获得图片列表的长度来,循环得到.

calss属性是动态获取 on:当index===nowindex的时候为true;

goto(*index*) {

*this*.ifshow = true;

*this*.nowindex = index;

},

**7. 父子组件之间事件的传递**

在父组件中,引入的子组件中

<products-top *@get*="getnum"></products-top>

自定义事件名 父组件的方法名

在子组件中的methods中

自定义函数名(){

this.$emit(“get”) --@get

}

wode(){

*this*.$emit("get");

}

**8. 当需要触发键盘事件时,用@keydown**

<input *type*="text" *name*="keyword" *placeholder*="搜索国家、城市、产品" *class*="my-form-control" *list*="s\_advice" *@keydown*="search($event)" *v-model*="kw"

当一个搜索框需要搜索当按下回车搜索内容时. 先绑定 v-model 获取输入的值

在methods中 判断 e.keyCode==13时,就进行传参 kw

if(e.keyCode==13){

*this*.$router.push({path:"/products/"+*this*.kw.trim()})

}

},

Kw在DOM,BOM中的获取方式: decodeURI(location.search.split(“=”)[1]);

Kw在vue-cli中的获取方式:this.$route.params.kw

**9. 如何使用鼠标移入事件更多当前图片,小图变大图, @mouseenter;**

**不论鼠标指针穿过被选元素或其子元素，都会触发 mouseover 事件。**

**只有在鼠标指针穿过被选元素时，才会触发 mouseenter 事件**

1. 在动态获得的图片列表下 –大图

<img *:src*="res.pics[`${i}`].md" *alt*=""*style*="width:100%;height:100%;">

在data({return})中初始化i=0;

2. 在下方小图中

<li *class*="ml-1 " *v-for*="(pic,i) in res.pics" *:key*="i">

<img *:src*="pic.sm" *@mouseenter*="changeImg(i)">

</li>

3.在methods中

changeImg(*i*){

*this*.i=*i*;

}

**10.动态绑定style属性 一般用于计算偏移量 margin;**

1. 在标签中

<div *class*="slidearea">

<ul *class*="list-unstyle sm\_img\_list d-flex " *:style*="move">

设置其父元素 overflow:hidden;

2.在data中设置一个变量 跟随移动的数量 n

3. 在事件方法中,每一次调用更改n的值

4. 在计算属性 components:{}中 监听move(){return `margin-left:${-80 \* this.n}px`};

computed: {

move() {

return `margin-left:${-80 \* *this*.n}px`;

}

},

**11. 收藏功能遇到的问题:**

1. 如何保存用户已经加入收藏清单的产品让其收藏按钮保持在高亮状态,如果高亮状态点击则取消高亮,并将收藏列表中的产品删除.

2. 判断当前是否有用户没登录

发一个ajax请求到后台查询是否登录,并保存响应回来的session中的值

<li *class*="mylove\_collection" *:class*="{love\_active:iscollection}" *@click.prevent*="addcollection(res.product.title,res.product.subtitle,res.product.price,res.product.old\_price,res.product.sold\_count,res.pics[0].md,res.product.tid)"><a>收藏</a></li>

在点击按钮处设置一个变量 iscollection bool值来判断当前按钮是否高亮,并且绑定单击事件,传入参数,注意位置顺序的问题.

3. 在created()函数中, 有两个ajax异步请求,但是前后有依赖关系,所以利用es7新特性 用一个 (async function(self){})(this); 然后 var res = await self.$http.get(“url”);

注意this的执向,在匿名函数中的this默认指向全局,这些将vue中正确的this传参进来

4. 因为在同一个组件中,参数的变化不会刷新页面,所以无法再次获得最新的产品的数据

解决:在watch中 监听路由参数的变化,对应的给出处理函数.重新获得当前的tid(路由的参数),再进行判断

watch: {

// 想对路由参数的变化作出响应的话，可以简单地 watch (监测变化) $route 对象：

'$route'(*to*, *from*) {

var *tid* = *this*.$route.params.tid;

*this*.$http.get(`/collection/iscollection?uname=${*this*.uname}&tid=${*tid*}`).then(*res* => {

*console*.log(*res*)

if (res.data.code == 1) {

*this*.iscollection = true;

} else {

*this*.iscollection = false;

}

})

},

},

**12.使用锚点跳转需要在router.js设置**

mode:'history',

scrollBehavior(*to*,*from*,*savedPosition*){

if(to.hash){

return {

selector:to.hash

}

}

}

自定义事件实现锚点跳转

<li><a *@click*="goAnchor('#anchor-'+index)" *:class*="{text\_actived:index==n}">商品简介</a></li>

//锚点内容

<div *class*="prebook\_parent" *id*="anchor-1">

在methods{}方法中

goAnchor(*selector*){

var *anchor* = *this*.$el.querySelector(*selector*);

*this*.n = anchor.id.split("-")[1];

//当前元素到顶部的高度+之前元素的站位高度

document.documentElement.scrollTop= anchor.offsetTop+1050;

}

**13.自定义指令**

<div *class*="prebook\_parent" :*id*="`anchor-`+index"”>

Vue.directive(‘anchor’,{

inserted:function(el,bingding){

el.onclick = function(){

document.documentElement.scrollTop=document.querySelector(“’#achchor’-binding.value”).offsetTop

}

}

})

如果需要处理时间 实现24小时制度,使用过滤器

Vue.filter("dateFormat",function(*datestr*,*pattern*="YYY-MM-DD HH:mm:ss"){

return **new** *Date*(datestr).toLocaleString('chinese',{hour12:false});

**});**

**14.当前页需要登录,登录后返回当前页的解决方法**

1. 在登录事件中

gologin() {

*this*.$router.push({

path: "/user\_login",

query:{redirect:`/product\_details/${*this*.tid}`}

})

},

2. 在login.vue中 接收传过来的query对象

if(*this*.$route.query.redirect!=undefined){

*this*.$router.push({path:`${*this*.$route.query.redirect}`})

}else{

setTimeout(()=>{

*this*.$router.push({path:"/home"})

},2000)

}

**15.监听当前滚轮的位置,给出判断某些元素是否显示或者定位**

**1. 在方法中 定义**

handlerScroll() {

var *scrollTop* = document.pageYOffset || document.documentElement.scrollTop || document.body.scrollTop;

if (*scrollTop* >= 1200) {

*this*.ifshow = true;

*this*.iffixed = true;

} else {

*this*.ifshow = false;

*this*.iffixed = false;

}

},

**2.在mounted中添加监听事件,获取滚轮实时位置**

mounted() {

window.addEventListener("scroll", *this*.handlerScroll, true);

},

**3.添加 定时器递归的方法 让跳转更平滑**

jump(){

*this*.distance = document.documentElement.scrollTop || document.body.scrollTop ||window.pageYOffset;

*this*.step = *this*.distance/50;

if(*this*.distance>0){

*this*.distance-=*this*.step;

window.scrollTo(0,*this*.distance);

setTimeout(*this*.jump,1);

}

}

**16.如何较愉快的使用provide()和inject属性**

**1. 在app.vue跟组件中 定义指令是否显示其他单页面**

<router-view *v-if*="isRouterAlive"></router-view>

**2. 在vue实例创建的过程中,提供一个方法 刷新**

provide() {

return {

reload: *this*.reload,

}

},

data() {

return {

isRouterAlive: true,

**3. 在方法中定义reload**

methods: {

reload() {

*this*.isRouterAlive = false; //先让整个页面隐藏

*this*.$nextTick(() => {

*this*.isRouterAlive = true; //本轮回掉函数 最快让他显示出来

})

},

},

**4. 在子组件中 使用 inject:[‘reload’] 调用: this.reload()**

**17 图片懒加载**

**VueLazyLoad from ‘vue-lazyload’**

**Vue.use(VueLazyload)**

**<img :src> 🡪 <img v-lazy=””>**

**18.倒计时页面**

var timer = setInterval(()=>{

if(*this*.num<=0){

*this*.$router.push("/home");

clearInterval(timer)

}else{

*this*.num--

}

},1000)

**19. 如何较为愉快的使用Vuex**

使用vuex来集中保存一些共享保存数据. npm I vuex -S

1. main.js中 引入import Vuex from ‘vuex’; Vue.use(Vuex) 创建 var stroe = new Vuex.Store({})

2. 共享的数据

state:{ //共享的数据

carCount:0, //购物车中的数量

isLogin:false, //控制用户是否登录

uname:"",//用户的姓名

islogin\_register:false

},

3. 配置actions 一般配axios请求

3. actions:{

saveForm(*context*) {

axios({

method:'get',

url:'users/islogin',

// data:context.state.isLogin

}).then((*res*)=>{

if(res.data.ok==1){

store.state.isLogin = true;

store.state.uname = res.data.uname;

store.state.islogin\_register = false;

}else{

store.state.isLogin = false;

store.state.islogin\_register = true;

}

}).catch((*err*)=>{

*console*.info(err)

})

}

**4. 操作共享数据的的方法**

mutations:{ //操作共享数据的方法

increment(*state*,*count*){

state.carCount+=count;

},

substract(*state*,*count*){

state.carCount-=count;

if(state.carCount<=0){

state.carCount=0;

}

},

clearcount(*state*){

state.carCount=0;

},

},

**5. 获取返回值并监听数据的方法**

getters:{ //获取并监听数据的方法

optCount:function(*state*){

return state.carCount

},

optIsLogin:function(*state*){

return [state.isLogin,state.uname,state.islogin\_register]

}

}

**6. 在使用共享数据的时候**

**① 如果使用actions中的ajax请求 this.$store.dispatch(“saveForm”)**

**② 一般读取**

*v-if*="$store.getters.optIsLogin[0]">{{$store.getters.optIsLogin[1]}}

//退出登录

signout() {

*this*.$axios.get("users/signout").then(*res* => {

*this*.$store.state.isLogin = false;

*this*.$store.state.islogin\_register = true;

})

}

**4.在数据库mysql当中**

**当希望修改一个表里的列的字符段内容时,可用concat函数拼接**

**比如 希望在图片的src前面加上** [**http://127.0.0.1:3000/**](http://127.0.0.1:3000/)

1. 可用[UPDATE](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/update.html) qy\_product\_pic [set](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/set.html) sm = CONCAT("http://127.0.0.1:3000/",sm)

希望更新有替代的内容

UPDATE qy\_product\_pic set sm = REPLACE(sm,’ <http://47.107.120.137>’,’旧的内容’)

1. 夸表查询

[SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html) y.qid,y.details,x.md FROM qy\_index\_product y INNER JOIN qy\_product\_pic x ON y.qid = x.pid

或者

SELECT title,price,tid, (SELECT sm FROM qy\_product\_pic WHERE product\_id=tid LIMIT 1) AS sm FROM qy\_travel\_product LIMIT ?,?

**常见错误**

**1. handlers[i].call is not a function"**

处理程序 调用的不是函数

在vue中钩子函数created() { ….} 而不是created:{}

**2. 在引入子组件的路径中**

**There are multiple modules with names that only differ in casing.  
有多个模块同名仅大小写不同  
This can lead to unexpected behavior when compiling on a filesystem with other case-semantic.  
这可能导致在一些文件系统中产生不是预期的行为  
Use equal casing.   
使用唯一的写法**

原因1: improt时,文件引入的路径描述不统一,所以保留一种风格引入 尽量使用最近的路径来引入

原因2:文件名和引入的不一致,**区分大小写**

在Header组件中 引入子组件

import HomeHeaderTop from "./HomeHeaderTop"

**后面引入时也应该 ./HomeHeaderTop 切忌不要写成TOP**

**3.组件之间传参时 一定要注意参数的正确性**

**1. 当从首页的商品跳转到详情页组件时,**

getDetails(*id*){

*this*.$router.push("/product\_details/"+*id*)

}

**此时id时当前详情页的id号**

**2. 在路由中:**

{

path:'/product\_details/:tid',

name:'ProductDetails',

component:*ProductDetails*

}

**此时 path /:tid为变量 接住从首页商品跳转过来的参数值(id)**

**3. 在当前页面中 data( return{})中**

tid:*this*.$route.params.tid,

**当前页面要用的tid变量是params.后的变量 从路由传参中接到的tid**

**在axios发送请求的时候**

*this*.$http.get("details?tid="+*this*.tid).then(*res* => {

**在微信小程序中 如何动态的去操作数组或者添加值**

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

var isSelect = "productList["+i+"].isSelect"

*this*.setData({

[isSelect]:false

})

//改变数量

modifyCount(*e*) {

var el = e.currentTarget.dataset;

var index = el.index;

var num = *this*.data.productList[index].count;

if (el.isadd === "true") {

var key = "productList[" + index + "].count";

num++

*this*.setData({

[key]: num

})

} else if (el.isadd === "false") {

var key = "productList[" + index + "].count";

num--

if (num > 0) {

*this*.setData({

[key]: num

})

} else {

*this*.setData({

[key]: 1

})

}

}

},

<view *class*="cart-num" *hover-class*="button-hover">

<view *class*="btn" *bindtap*="modifyCount" *data-isadd*="false" *data-index*="{{index}}" *data-count*="{{item.count}}">-</view>

<text>{{item.count}}</text>

<view *class*="btn" *bindtap*="modifyCount" *data-isadd*="true" *data-index*="{{index}}" *data-count*="{{item.count}}">+</view>

</view>

**2. 购物车逻辑**

<view *class*="divider"></view>

<view *class*="cart-product-warp" *wx:for*="{{productList}}" *wx:for-index*="index" *wx:key*="{{productList.pid}}">

<view *class*="selected-warp">

<image *class*="item-select" *bindtap*="switchSelect" *data-index*="{{index}}" *data-id*="{{item.pid}}" *src*="{{item.isSelect?'../../assets/nav\_icons/comment\_select.png':'../../assets/nav\_icons/comment\_normal.png'}}" />

</view>

<image *src*="{{item.md}}" *alt*="" *lazy-load*="{{true}}" />

<view *class*="cart-product-details">

<text *class*="text-ellipse">{{item.title}}</text>

<text>原价:<text>¥ {{item.old\_price}}</text> </text>

<text> 秒杀特价:<text>¥ {{item.price}}</text> </text>

</view>

<view *class*="cart-num" *hover-class*="button-hover">

<view *class*="btn" *bindtap*="modifyCount" *data-isadd*="false" *data-index*="{{index}}" *data-count*="{{item.count}}">-</view>

<text>{{item.count}}</text>

<view *class*="btn" *bindtap*="modifyCount" *data-isadd*="true" *data-index*="{{index}}" *data-count*="{{item.count}}">+</view>

</view>

</view>

<!-- 底部固定结算 -->

<view *class*="cart-bottom-warp">

<view *class*="selected-warp" *bindtap*="allSelect">

<image *class*="item-select" *data-index*="{{index}}" *data-id*="{{index}}" *src*="{{isAllSelect?'../../assets/nav\_icons/comment\_select.png':'../../assets/nav\_icons/comment\_normal.png'}}" />

<text *class*="allselect">全选</text>

</view>

<view *class*="total">

<text>合计：</text>

<text>¥ {{totalMoney}}</text>

<view *class*="pay-button" *bindtap*="toBuy">结算</view>

</view>

</view>

// 当选中的状态放生变化的时候

switchSelect:function(*e*) {

//设置总价格

var Allprice = 0;

//当前触发操作的元素

var el = e.currentTarget.dataset;

//获和下标

var index = parseInt(el.index);

*this*.data.productList[index].isSelect = !*this*.data.productList[index].isSelect;

//2. 价钱统计

//价钱统计

if (*this*.data.productList[index].isSelect) {

*this*.data.totalMoney = Math.round(*this*.data.totalMoney + *this*.data.productList[index].price\**this*.data.productList[index].count);

}else{

*this*.data.totalMoney = Math.round(*this*.data.totalMoney - *this*.data.productList[index].price\**this*.data.productList[index].count);

}

//3.判断是否全选

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

Allprice = Math.round(Allprice+*this*.data.productList[i].price\**this*.data.productList[i].count);

}

if(Allprice == *this*.data.totalMoney){

*this*.data.isAllSelect = true;

}else{

*this*.data.isAllSelect = false;

}

*this*.setData({

productList:*this*.data.productList,

totalMoney:*this*.data.totalMoney,

isAllSelect:*this*.data.isAllSelect

})

},

allSelect: function (*e*) {

//处理全选逻辑

var i = 0;

if (!*this*.data.isAllSelect){

for (i = 0; i < *this*.data.productList.length; i++) {

*this*.data.productList[i].isSelect = true;

*this*.data.totalMoney = Math.round(*this*.data.totalMoney + *this*.data.productList[i].price\**this*.data.productList[i].count);

}

} else {

for (i = 0; i < *this*.data.productList.length; i++) {

*this*.data.productList[i].isSelect = false;

}

*this*.data.totalMoney=0;

}

*this*.setData({

productList: *this*.data.productList,

isAllSelect: !*this*.data.isAllSelect,

totalMoney: *this*.data.totalMoney,

})

},

// 去结算

toBuy() {

wx.showToast({

title: '去结算',

icon: 'success',

duration: 3000

});

},

//改变数量

modifyCount(*e*) {

var el = e.currentTarget.dataset;

var index = el.index;

var num = *this*.data.productList[index].count;

if (el.isadd === "true") {

var key = "productList[" + index + "].count";

num++

*this*.setData({

[key]: num

})

} else if (el.isadd === "false") {

var key = "productList[" + index + "].count";

num--

if (num > 0) {

*this*.setData({

[key]: num

})

} else {

*this*.setData({

[key]: 1

})

}

}