Vue项目步骤第04天

# 购物车功能的实现

1.1、将购物车数据存储在localStorage中

1. 规定数据格式，以json字符串的形式进行存储

{

87:{

id:87,

count:1,

title:标题,

price:"100"

},

88:{

id:88,

count:1,

title:标题,

price:"100"

}

}

b、添加到购物车的时候存储数据

addToCart:function(){

//将购物车数据存储在localStorage中

var cartDataStr = localStorage.getItem("cartData") || "{}";

var cartObj = JSON.parse(cartDataStr);

var savedItem = cartObj[this.info.id];

if(savedItem) {

//已经存在数据了

savedItem.count = parseInt(savedItem.count) + parseInt(this.inputNumberCount)

} else {

//从来没有这个商品的数据

savedItem = {};

savedItem.id = this.info.id;

savedItem.title = this.info.title;

savedItem.price = this.info.sell\_price;

savedItem.count = this.inputNumberCount;

}

cartObj[this.info.id] = savedItem;

localStorage.setItem("cartData",JSON.stringify(cartObj));

//更新徽章的数量显示

}

c、在App.vue加载的时候，读取localStorage的值，更新徽章显示

created(){

var savedObj = JSON.parse(localStorage.getItem("cartData")||"{}");

for(var key in savedObj) {

this.total += savedObj[key].count;

}

}

d、将和localStorage相关的功能封装到一个js文件中utils.js

export function addNumber(goodsInfo,number){

var cartDataStr = localStorage.getItem("cartData") || "{}";

var cartObj = JSON.parse(cartDataStr);

var savedItem = cartObj[goodsInfo.id];

if(savedItem) {

//已经存在数据了

savedItem.count = parseInt(savedItem.count) + parseInt(number);

} else {

//从来没有这个商品的数据

savedItem = {};

savedItem.id = goodsInfo.id;

savedItem.title = goodsInfo.title;

savedItem.price = goodsInfo.sell\_price;

savedItem.count = number;

}

cartObj[goodsInfo.id] = savedItem;

localStorage.setItem("cartData",JSON.stringify(cartObj));

}

export function getTotalCount() {

var savedObj = JSON.parse(localStorage.getItem("cartData")||"{}");

var totalCount = 0;

for(var key in savedObj) {

totalCount += savedObj[key].count;

}

return totalCount;

}

export function updateTo(id,newNumber){

var cartDataStr = localStorage.getItem("cartData") || "{}";

var cartObj = JSON.parse(cartDataStr);

var savedItem = cartObj[id];

savedItem.count = parseInt(newNumber);

cartObj[id] = savedItem;

localStorage.setItem("cartData",JSON.stringify(cartObj));

}

export function deleteItem(id){

var savedObj = JSON.parse(localStorage.getItem("cartData")||"{}");

delete savedObj[id];//删除属性

}

export function getAll() {

var savedObj = JSON.parse(localStorage.getItem("cartData")||"{}");

return savedObj;

使用如下：

import { addNumber } from "../../kits/utils.js";

addNumber (this.info,this.inputNumberCount);

1.2、添加商品到购物车更新徽章显示

在goodsinfo.vue组件中，点击添加到购物车按钮，需要让App.vue中的数字进行更新。

goodsinfo.vue和App.vue不存在父子组件的关系，不能使用props的方式，因此需要另辟蹊径。

仔细理解一下$emit()的含义：产生了一个事件，谁监听了这个事件，将会得到响应。

复杂关系的组件传值通用解决方案：

1. 在window中，存储入口函数中new出来的Vue对象

new Vue({

el:'#app',

router:router1,

render:function(create){

return create(App);

},

created:function(){

window.eventBus = this;

}

});

1. 点击添加到购物车时，发出一个事件

window.eventBus.$emit("cartUpdated","");

1. 在App.vue中监听这个事件

注册一个事件的时机？created，相当于在window.onload里面注册onclick

created(){

//………

window.eventBus.$on('cartUpdated', function(val) {

};

)

1. 实现业务逻辑

created(){

this.total = getTotalCount();

window.eventBus.$on("cartUpdated",function(){

var currentDom = document.querySelector(".mui-badge");

currentDom.innerText = getTotalCount() + "";

});

}

1.3、添加商品到购物车动画效果的实现

1. 实现动画效果

增加div，设置好样式，并且给这个div增加v-show属性

<div v-show="show" class="ball"></div>

show:false,

.ball{

width: 20px;

height: 20px;

border-radius: 10px;

background-color: red;

position: absolute;

left: 150px;

top: 10px;

transition:all 3s;

z-index: 1000;

}

给div外面增加一个transition标签，在transition标签中对动画的事件进行监听

<transition v-on:before-enter="beforeEnter" v-on:enter="enter" v-on:after-enter="afterEnter">

<div v-show="show" class="ball"></div>

</transition>

在before-enter中设置初始位置，在enter中设置结束位置，在after-enter中设置div隐藏

beforeEnter:function(el){

el.style.transform = "translate(0px,0px)";

},

enter:function(el,done){

el.offsetWidth;

var targetX = 100;

var targetY = 100;

el.style.transform = "translate("+targetX+"px,"+targetY+"px)";

done();

},

afterEnter:function(el){

this.show = false;

}

在添加购物车按钮的时候，将show变为true

addToCart:function(){

//将购物车数据存储在localStorage中 addNumber(this.info,this.inputNumberCount);

//更新徽章的数量显示

window.eventBus.$emit("cartUpdated","");

this.show = true;

},

1. 偏移量的计算

得到徽章相对于屏幕的x轴坐标y轴坐标

得到小红点相对于屏幕的x轴坐标y轴坐标

function getTop(e){

var offset=e.offsetTop;

if(e.offsetParent!=null) offset+=getTop(e.offsetParent);

return offset;

}

function getLeft(e){

var offset=e.offsetLeft;

if(e.offsetParent!=null) offset+=getLeft(e.offsetParent);

return offset;

}

可以将这两个方法放到utils中去

二者一减得到translate的值

注意：需要将屏幕滚动考虑进去

var targetX = localStorage.getItem("bagedLeft")-getLeft(el);

var targetY = localStorage.getItem("bagedTop") - (getTop(el) - window.scrollY );

注意，需要在App.vue加载完的时候，将徽章的坐标记录在localStorage中，时机点在mounted中

mounted:function(){

var badge = document.querySelector(".mui-badge"); localStorage.setItem("bagedLeft",getLeft(badge));

localStorage.setItem("bagedTop",getTop(badge));

}

1.4、购物车组件界面实现

静态界面的实现

<template>

<div id="tmpl">

<div class="row" v-for="(item,index) in datalist">

<mt-switch class="switch"></mt-switch>

<img class="img" alt="商品图片">

<div class="inforight">

<h4>标题</h4>

<div class="bottom">

<ul>

<li>￥200元</li>

<li>

<inputnumber></inputnumber>

</li>

<li>

<a href="javascript:void(0)">删除</a>

</li>

</ul>

</div>

</div>

</div>

<div id="total">

<div class="desc">

<ul>

<li>总计（不含运费）</li>

<li>已勾选商品5件,总价:￥0元</li>

</ul>

</div>

<div id="button">

<mt-button type="danger" size="small">去结算</mt-button>

</div>

</div>

</div>

样式配置：

<style scoped>

.row{

border-bottom: 1px solid rgba(0,0,0,0.3);

height: 102px;

display: flex;

padding: 5px;

}

.switch{

flex:0 0 52px;

}

.img{

margin-left: 5px;

height: 75px;

width: 75px;

flex: 0 0 85px;

}9o

.inforight{

margin-left: 5px;

flex:1;

}

.inforight ul{

padding-left: 0px;

}

.inforight li{

list-style: none;

display: inline-block;

}

.inforight h4{

color: #0094ff;

font-size: 14px;

}

.bottom li:first-child{

color:red;

margin-right: 5px;

}

.bottom li:last-child{

margin-left: 5px;

}

#total{

height: 100px;

background-color: rgba(0,0,0,0.1);

display: flex;

padding: 5px;;

}

#total ul {

padding-left: 0px;

}

#total li{

list-style: none;

font-size: 14px;

}

#button{

flex:0 0 60px;

margin: 30px 0 0 0 ;

}

.desc{

flex:1;

}

</style>

从localStorage中读出数据进行展示

<script>

import inputnumber from "../subcom/inputNumber.vue";

import {getAll} from "../../kits/utils.js";

export default{

components:{

inputnumber:inputnumber

},

data:function(){

return {

datalist:[1,3,2]

}

},

created:function(){

this.datalist = getAll();

}

}

</script>

将界面中写死的数据变活，例如：

<h4>{{item.title}}</h4>

将localStorage中的某个商品的数量传给inputnumber这个组件

在inputNumber.vue中

props:["initcount"],

created:function(){

if(this.initcount){

this.count = this.initcount;

}

}

在car.vue中

<inputnumber v-bind:initcount="item.count"></inputnumber>

点击inputnumber左右两个按钮的时候，需要改变localStorage的值。存储的时机不应该写在inputnumber中，因为在商品详情界面中inputnumber的增加和减少操作不会写localStorage，因此写localStorage应该放在car.vue中，写完localStorage之后，car.vue还需要通知App.vue去刷新徽章

<li>

<inputnumber v-on:dataobj="getMessage" v-bind:initcount="item.count" v-bind:goodsid="item.id"></inputnumber>

</li>

methods:{

getMessage:function(input){

var goodsId = input.split(":")[0];

var finalNumber = input.split(":")[1];

updateTo(goodsId,finalNumber);

window.eventBus.$emit("cartUpdated","");

}

}

在inputNumber.vue中，修改部分代码

sendmessage(){

this.$emit('dataobj',this.goodsid + ":" + this.count);

}

发出的事件中，除了带有数量之外，还带有商品id

修改goodsinfo.vue文件

getcount(input){

this.inputNumberCount = parseInt(input.split(":")[1]);

},

1.5、购物车组件操作功能的实现

给mt-switch配置v-model，绑定每一个item中的checked属性

计算总价格

代码如下：

<mt-switch class="switch" v-model="item.checked" v-on:change="switchChange"></mt-switch>

<div id="total">

<div class="desc">

<ul>

<li>总计（不含运费）</li>

<li>已勾选商品{{totalAmount}}件,总价:￥{{totalPrice}}元</li>

</ul>

</div>

<div id="button">

<mt-button type="danger" size="small">去结算</mt-button>

</div>

</div>

<script>

import inputnumber from "../subcom/inputNumber.vue";

import {getAll,updateTo} from "../../kits/utils.js";

export default{

components:{

inputnumber:inputnumber

},

data:function(){

return {

datalist:[1,3,2],

totalAmount:0,

totalPrice:0

}

},

created:function(){

this.datalist = getAll();

for(var key in this.datalist) {

this.datalist[key].checked = false;

}

this.calc();

},

methods:{

getMessage:function(input){

var goodsId = input.split(":")[0];

var finalNumber = input.split(":")[1];

updateTo(goodsId,finalNumber); window.eventBus.$emit("cartUpdated","");

this.datalist[goodsId].count = parseInt(finalNumber);

this.calc();

},

calc:function(){

this.totalAmount = 0;

this.totalPrice = 0;

for(var key in this.datalist) {

var obj = this.datalist[key];

if(obj.checked) {

this.totalAmount += parseInt(obj.count);

this.totalPrice += parseInt(obj.count) \* parseInt(obj.price);

}

}

},switchChange:function(){

this.calc();

}

}

}

</script>

# 2、其他

* 1. 返回功能的实现

返回效果的实现

返回按钮的隐藏和显示

* 1. 打包部署

webpack -p