# 数据

## 知识点

### 构造类

class 类名｛

constructor(形参列表){

//构造函数

//属性列表

}

//方法列表

方法名(){

}

｝

### 构造对象

var 对象名=new 类名（实际参数列表）

## 数据结构

商品类-- Product

|  |  |
| --- | --- |
| id---编号  title—名称  price—价格  imgSrc---图片地址 | class product {  constructor(id, title, imgSrc, price) {  this.id = id;  this.title = title;  this.imgSrc = imgSrc;  this.price = price;  }  } |

订单类—Order

|  |  |
| --- | --- |
| id---商品编号  title—商品名称  price—商品价格  imgSrc--商品图片地址  qty--数量  selectStatus—选择状态 | class Order {  constructor(product, qty, selectStatus) {  this.id = product.id;  this.title = product.title;  this.imgSrc = product.imgSrc;  this.price = product.price;  this.qty = qty;  this.selectStatus = selectStatus;  }  }; |

购物车类—CartData—构建时是空购物车状态

|  |  |
| --- | --- |
| 订单列表  总样本数  总件数  总价格 | class CartData {  constructor() {  this.orderList = new Array();  this.totalQty = 0;  this.totalAmount = 0;  this.units = 0;  }  } |

# 购物车数据操作

知识点：

// localStorage.setItem("key","value");//存储变量名为key，值为value的变量

// localStorage.key = "value"//存储变量名为key，值为value的变量

// localStorage.getItem("key");//获取存储的变量key的值

// localStorage.key;//获取存储的变量key的值

// localStorage.removeItem("key")//删除变量名为key的存储变量

购物车类（操作）

|  |  |
| --- | --- |
| setDataToLocalStorage:数据写入localStorage  getDataFromLocalSatorge :读取数据  addToCart(order):将订单加入购物车 | Class ShoppingCart{  setDataToLocalStorage(cartData) {  }  getDataFromLocalSatorge()｛  ｝  addToCart(order)｛  ｝  。。。。。。。  } |

## 数据写入localStorage—ShoppingCart. setDataToLocalStorage()

### 技术

（1）将数据（字符串）写入本地存储 localStorage.setItem("key","value");//存储变量名为key，值为value的变量

（2）变量转换为JSON字符串JSON.stringify(变量名)

### 解题思路

入口参数 cartData(购物车数据)

1. 将cartData转换为JSON字符串

JSON.stringify(cartData)

1. 将cartData字符串写入localStorage

localStorage.setItem("key","value");//存储变量名为key，值为value的变量

|  |
| --- |
| // 将购物车数据写入本地存储中  setDataToLocalStorage(cartData) {  //清除原有存储写入新列表  localStorage.removeItem('lzzyCart');  //写入本地存储  localStorage.setItem('lzzyCart', JSON.stringify(cartData));  } |

## 从localStorage读取数据

### 技术

（1）从本地存储中获取指定key的值-- localStorage.getItem("key")

（2）JSON字符串转JSON变量JSON.parse(字符串)

### 解题思路

1. localStorage.getItem("key");//获取存储的变量key的值
2. 判断读取到到数据是否为null或者‘’
   1. 不为空转换json字符串🡪js对象 JSON.parse(字符串)
   2. 为空，创建新购物车数据对象
3. 返回js对象

|  |
| --- |
| // 从本地存储中获取购物车数据  getDataFromLocalStorage() {  let lzzyCart = localStorage.getItem('lzzyCart');  // 判断购物车是否为空  if (lzzyCart == null || lzzyCart == '') {  return new CartData();  }  else {  return JSON.parse(lzzyCart);  }  } |

## 订单数据加入购物车（本地存储）

### 技术

for-in 遍历数组或对象

for (key in object) {

object[key]//如果object是对象则object[key]等价于引用object的key成员 等价于object.key

}

for (key in object) {

object[key]//如果object是数组则object[key] 等价于object数组等第key个元素

}

### 解题思路

1. 读取本地存储中的购物车数据 cartData=this. getDataFromLocalSatorge()
2. 把订单加入订单列表中
3. 完善总计等数据
4. 将新购物车数据写入本地存储 this. setDataFromLocalSatorge(cartData)

### 代码实现

|  |
| --- |
| addToCart(order) {  // 从本地存储中获取购物车的数据  let cartData = this.getDataFromLocalStorage();  // 获取购物车json数据中的订单列表  let orderList = cartData.orderList;  //设置标志位判断是否为购物车新商品，默认为是新商品  let isNewProduct = true;  // 遍历订单列表，判断新加入商品是否在购物车中  for (let i in orderList) {  if (order.id == orderList[i].id) {  // 若新加入订单商品已经在购物车中，则变更订单列表中对应商品数量，且变更新商品标志位  orderList[i].qty += order.qty;  isNewProduct = false;  break;  }  }  //如果是新商品  if (isNewProduct) {  // 购物车总样本+1  cartData.units++;  // 导入新商品置入购物车  orderList.push(order);  }  //修改购物车总金额及商品总数量  cartData.totalAmount += order.qty \* order.price;  cartData.totalQty += order.qty;  // 写入localStorage  this.setDataToLocalStorage(cartData);  } |

## 清空购物车

### 技术

移除指定键位的本地存储localStorage.removeItem(“key”);

清除所有本地存储 localStorage.clear

### 解题思路

移除指定键位的本地存储

### 代码实现

|  |
| --- |
| // 清空购物车（移除本地存储购物车项）  clearCart() {  localStorage.removeItem('lzzyCart');  } |

## 获取选中订单的总件数

### 解题思路

1. 获取购物车数据
2. 获取订单列表
3. 遍历订单列表，逐一判断每个订单的选择状态
   1. 选中的，累加订单数量
   2. 未选中，就不统计该订单数量
4. 返回选中状态的总数量

### 代码实现

|  |
| --- |
| // 获取选中商品的总数量  getSelectedQty() {  let cartData = this.getDataFromLocalStorage();  let orderList = cartData.orderList;  let selectedQty = 0;  for (const key in orderList) {  if (orderList[key].selectStatus) {  selectedQty += orderList[key].qty;  }  }  return selectedQty;  } |

## 获取选中订单的总价格

### 解题思路

1. 获取购物车数据
2. 获取订单列表
3. 遍历订单列表，逐一判断每个订单的选择状态
   1. 选中的，累加 （订单数量\*订单价格）
   2. 未选中，就不统计
4. 返回选中状态的总价格

### 代码实现

|  |
| --- |
| // 获取选中商品的总价格  getSelectedAmount() {  let cartData = this.getDataFromLocalStorage();  let orderList = cartData.orderList;  let selectedAmount = 0;  for (const key in orderList) {  if (orderList[key].selectStatus) {  selectedAmount += orderList[key].qty \* orderList[key].price;  }  }  return selectedAmount;  } |

## 设置某个订单（id）的选择状态（selectStatus）

### 解题思路

1. 获取购物车数据
2. 获取订单列表
3. 遍历订单列表，逐一判断每个订单的id是否等于指定id（形式参数）
   1. 找到对应id,设置选择状态selectStatus（形式参数） 结束循环
   2. 没找到，继续遍历直到订单列表遍历完成
4. 如果状态设置成功，将新购物车数据写入本地存储。

### 代码实现

为了获取指定id对应OrderList的订单，设计了一个find(id) 函数

|  |
| --- |
| // 查找指定id的订单  find(id, orderList) {  for (const i in orderList) {  if (id == orderList[i].id) {  return orderList[i];  }  }  return null;  } |

|  |
| --- |
| // 获取购物车数据  let cartData = this.getDataFromLocalStorage();  let orderList = cartData.orderList;  // 查找id对应的订单  let order = this.find(id, orderList);  //判断位置，位置为空报错提示，如果不为空就设置状态  if (order == null) {  // 没有找到id  console.log('订单id有误');  } else {  // 找到对应id  order.selectStatus = selectStatus;  // 写入本地存储  this.setDataToLocalStorage(cartData);  } |

## 删除某个指定（id）订单

### 关键技术

JS数组操作<http://www.runoob.com/jsref/jsref-obj-array.html>

删除元素：

Pop() 删除最后一个元素

[shift()](http://www.runoob.com/jsref/jsref-shift.html) 删除第一个元素

splice() 删除指定元素

splice()

语法

array.splice(要删除元素的下标,删除元素个数,[新增元素1,.....,])

查找指定元素在数组中的索引

array .indexOf(需要查找的元素，查找开始的位置)

### 解题思路

1. 获取购物车数据
2. 获取订单列表
3. 遍历订单列表，逐一判断每个订单的id是否等于指定id（形式参数）
   1. 找到对应id,，删除订单，修改总样本数、总价格、总件数
   2. 没找到，继续遍历直到订单列表遍历完成，提示没有指定id
4. 如果删除成功，数据写入本地存储

（3）-a)

删除订单：

查找指定id订单在订单数组中的索引—index

删除指定索引(index)的元素

修改各种总和

### 代码实现

|  |
| --- |
| // 删除指定ID商品  deleteItem(id) {  // 获取购物车数据  let cartData = this.getDataFromLocalStorage();  // 获取订单列表  let orderList = cartData.orderList;  // 获取指定id的订单(要删除的订单)  let order = this.find(id, orderList);  //定位要删除的订单在数组中的位置  let index = orderList.indexOf(order, 0);  if (index == -1) {  // 找不到需要删除的订单  console.log('订单id有误');  } else {  // 删除当前订单  orderList.splice(index, 1);  // 变更总商品总件数  cartData.totalQty -= order.qty;  //变更商品总价格  cartData.totalAmount -= order.qty \* order.price;  // 变更总商品件数  cartData.units--;  //数据回写购物车  this.setDataToLocalStorage(cartData);  }  } |

## 某个指定订单数量减1

## 某个指定订单数量加1

# 购物车页面

扩展技术参考：

HTML标签参考手册：<http://www.runoob.com/tags/html-reference.html>

HTML属性参考手册：<http://www.runoob.com/tags/ref-standardattributes.html>

DOM

Document <http://www.runoob.com/jsref/dom-obj-document.html>

Element <http://www.runoob.com/jsref/dom-obj-all.html>

Attribute http://www.runoob.com/jsref/dom-obj-attributes.html

EVENT <http://www.runoob.com/jsref/dom-obj-event.html>

## 购物车页面模块结构

### 模块结构设计

|  |  |  |
| --- | --- | --- |
| 序号 | 模块 | 具体内容 |
| 1 | 页面数据初始化模块（语句组） |  |
| 2 | 显示订单列表模块（函数）  function displayOrderList(){…} |  |
| 3 | 显示总数据（函数）  Function displayTotalData(){…} |  |
| 4 | 注册事件（函数）  function regEvent() |  |
| 5 | 事件触发函数（函数组）  function 事件名EventFun(){…} |  |
| 6 | 页面加载时首先要执行的函数（函数）  function init(){…} |  |
| 7 | 调用页面加载时需要首先执行函数init(); |  |

### 一次页面加载各模块的调用执行顺序

1--》7--》6--》2--》3--》4

### 一次用户操作执行顺序

5-》根据需要处理—》2—》3

## 本地存储的购物车数据呈现到购物车列表页displayOrderList()

### 关键技术

将数据按设计HTML结构置入DOM中

1. 创建节点，设置节点属性，挂接到父元素
2. 拼接HTML字符串，拼接属性，赋值到父元素到innerHtml
3. 克隆样本节点，设置相关数据到具体到属性

找到样本节点 element.clone(true)

找到孩子节点 设置属性

1. 为了更容易找到相关节点而设计自定义属性 data-\*\*\*，通过CSS属性选择器抓取对应节点

例如：获取所有element 下到data-name属性值为imgSrc的所有节点

element.querySelectorAll('[data-name="imgSrc"]');

获取element 下第一个data-name属性值为imgSrc的所有节点

element.querySelector('[data-name="imgSrc"]');

1. 移除某个节点到某个属性

element.classList.remove('属性名');

扩展链接

DOM-元素操作http://www.runoob.com/jsref/dom-obj-all.html

### 设计样本节点

（1）为样本节点设计id

<div class="row p-3 mt-1 bg-light d-none" id="orderExample">…</div>

（2）为所有涉及到到操作节点(将来需要在该节点加载事件)，设计自定义属性

data-operator=“操作名”

例如：删除按钮

<button class="btn btn-light" data-operator='deleteItem'>删除</button>

（3）为所有涉及到到数据设计自定义属性

data-name=”数据名” 数据名与数据结构保持一致

例如：单价

<span data-name="price">0.00</span>

### 将数据显示到样本节点

获取购物车数据

获取购物车订单列表

遍历订单列表｛

克隆一个样本节点

设置一个新id

挂接到父元素

获取所有到数据节点data-name 依次将对应到数据送入节点对应属性

移除新节点到隐藏属性 d-none

｝

|  |
| --- |
|  |

### 为订单操作设计data-id属性

技术element.setAttribute(属性名，值)

# 注册事件函数

## 思路

（1）需要注册事件到节点：

订单选择、加号、减号、删除（父节点某个订单节点）

全选、清空购物车、删除选中到商品、结算（父节点是购物车）

（2）具体做法：

某个操作节点

为该节点注册事件

为该节点写触发函数

## 代码实现

|  |
| --- |
| // 为相关节点注册事件  function regEvent() {  // 获取清空购物车节点  // 注册单击事件触发函数  }  为某个节点写触发函数 |

### 为清空购物车操作节点注册事件

（1）思路：

获取清空购物车节点--写在注册函数里面

注册单击事件--写在注册函数里面

事件触发函数—写在注册函数外面

（2）代码实现—注册函数内

|  |
| --- |
| function regEvent() {  // 获取清空购物车节点  let element = cartRoot.querySelector(operatorGlobal.clearAll);  console.log(element);  // 注册单击事件触发函数  element.onclick = clearAllEventFun;  } |

1. 清空触发函数—独立写

|  |
| --- |
| // 清空事件触发函数  function clearAllEventFun() {  cart.clearCart();  console.log('清空');  } |

### 为删除订单操作节点注册事件

思路：

（1）注册事件

regEvent()

获取一组订单删除按钮

为每一个删除按钮设计单击事件—触发deleteItemEventFun

（2）设计单击触发函数---独立函数

deleteItemEventFun（e）()

（2-1）获取当前被单击的删除按钮 e.target

（2-2）根据当前被单击删除按钮找到当前订单id(之前需要在订单加载时给删除按钮注入data-id属性 )

element.getAttribute()

element.setAttribute()

---》调用购物车类删除订单方法 cart. deleteItem(id)

根据当前订单id找到当前订单节点

—》删除节点 订单列表节点.removeChild(当前订单节点)

修改各种总数据displaySelectedTotal()

# 页面加载时需要处理到函数init()

## 思路

显示订单列表

显示总数据

为操作节点注册事件

## 代码实现

|  |
| --- |
| // 初始化  function init() {  displayOrderList();  regEvent();  displayTotalData();  } |