今天学习的内容

1:复习上周重点

2:今天的目标

2.1:学子商城--购物车(删除/全选)

-删除用户指定的一个商品流程图



-分析功能

(1)用户点击购物车中某个商品删除按钮

(2)显示交互式提示框(防止用户误操作)

(3)如果用户选择"确定"

(4)发送ajax请求并且获取服务器返回结果

(5)如果删除成功,重新获取新购物车中内容

-特殊注意事件

(1)删除是一种有一定风险操作,原因数据如果

被删除**很难**恢复.

(2)如果有重要备份之后再删除

2.3:学子商城--购物车-删除一个商品(操作)

-数据库表

xz\_cart(id/lid/img\_url/price/count/uid/lname)

-app.js

(1)参数 购物车id

(2)sql DELETE FROM xz\_cart WHERE id = ?

(3)json {code:1,msg:"删除成功"}

-Cart.vue

(1)为"删除"按钮绑定点击事件

(2)获取购物车id

<mt-button @click="delItem"

:data-id="item.id">删除</mt-button>

delItem(e){ var id = e.target.dataset.id}

(3)显示交互式提示框,请用户再次确认是否删除

this.$messagebox.confirm("是否删除指定数据")

.then(res=>{//用户选择确定(4)(5)})

.catch(err=>{//用户选择取消})

(4)发送ajax /cart?id=1

(5)删除成功后获取返回结果,重新获取购物车列表

2.4:学子商城--购物车-删除多个商品(操作)



(1)xz\_cart

[id/lid/price/lname/....]

(2)app.js

参数: ids=2,3

sql:DELETE FROM xz\_cart id IN (2,3)

(3)Cart.vue

2.5:学子商城--购物车-全选

<input type="checkbox" @change="selectAll"/>

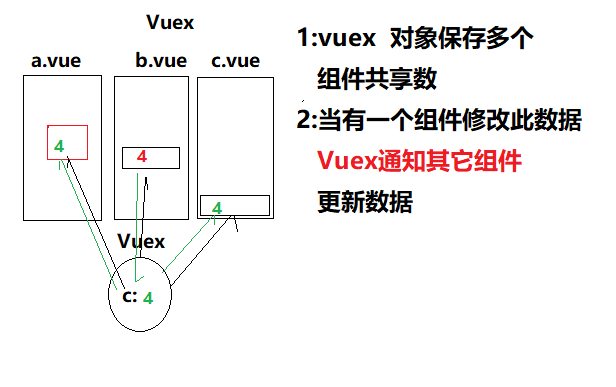
(1)全选按钮'选中'其它商品'选中'

全选按钮'取消'其它商品'取消'

(2)获取全选按钮状态 true checked表示状态

(3)list:[{cb:true}]

2.6:Vuex



Vuex是一个专为Vue应用程序的数据状态管理模式,

它采用集中式存储管理程序中所有组件数据状态

并以相应的规则保存状态以一种**同步方式**保证数据状态

统一

#Vuex 本身容器保存多个组件中需要*共享*的数据(状态)

#默认情况Vue组件自己管理自己数据(状态)

#data(){}

#A.vue data(){} B.vue data(){}

#有时数据需要多个组件之间共享(购物车中商品数量)

示例:

(1)将购物车中数量保存Vuex对象中(集中管理)

(2)Cart.vue 获取购物车数量

(3)Cart.vue 修改购物车中数量

2.7:Vuex 使用方式

(1)下载Vuex第三方模式

-脚手架下载 npm i vuex - S

#下载完成 node\_modules/vuex/...

(2)在main.js 对Vuex配置

-引入Vuex 第三方模块

import Vuex from "vuex"

-创建存储实例对象

var store = new Vuex.Store({

state:{cartCount:0},

mutations:{},

getters:{},

actions:{}

});

#state: 保存多个组件共享数据

#mutations: 添加***修改***共享数据函数

var i = 0; i++; vuex不能使用运算符直修改 错

#getters: 添加函数通过函数获取共享数据

#actions: 添加函数通过函数调用使用异步操作?

-注册Vuex

Vue.use(Vuex)

-将实例对象直接添加Vue对象中

new Vue({store}) #store存储实例对象

|  |
| --- |
| 示例:创建存储实例对象,保存购物车数量  var store = new Vuex.Store({  state:{  cartCount:0 //购物车中商品数量  },  mutations:{  //购物车商品数量减1  subItem(state){state.cartCount--},  //购物车商品数量加1  addItem(state){state.cartCount++},  //清空购物车量  clearItem(state){state.cartCount=0}  }  getters:{  //获取购物车数量  getCartCount(state){ return state.cartCount}  }  //用户有一些操作不是立即执行,加载  //更多数据,  //等待数据加载成功后 异步数据(修改数据)  actions:{  //修改数据函数保存mutations clear  //异步执行操作  //context 上下文对象=>$store对象  modifyCount:(context)=>{  //异步等待500毫秒调用clear函数  setTimeout(()=>{  context.commit("clear")  },500)  }  }  }) |

(3)在需要使用Vuex数据组件调用

-修改共享数据 调用mutations中添加函数

this.$store.commit("clearItem")

-获取共享数据(在模板中)

{{$store.getters. getCartCount}}

-执行actions里面函数

this.$store.dispath(' modifyCount ');

作业1:

(1)复习video 视频 弹幕