时间：2021-08-08~2021-09-05，耗时接近一个月

地址：<http://lixiufeng.3vzhuji.net/dingdongProduct>

简介

技术栈：

项目主要是移动端，vue、vue-router、vuex、axios、

UI库：vant（需要装插件，切换成Rem）

其他插件：better-scroll（滚动），fastclick（击穿现象），mockjs（拦截ajax，模拟服务器数据），vue-content-loader（骨架屏），vue-i18n（中英文切换），vue-waterfall2（图片瀑布流）

1. 准备的环境

1、安装vant、axois运行依赖，把原来的vue/cli中不需要的代码删除

Vue/cli中的用法

1、img图片，需要配置一下vue.config.js

vue-cli找不到图片的，img绑定变量，的时候，需要用reqiure，要注意看最后打包的时候，搞成了什么样子。

不绑定跟绑定img，它们经过webpack的地址是不一样的，F12可以查看

|  |
| --- |
| vue.config.js  //配置 运行和打包时的文件路径  publicPath:process.env.NODE\_ENV==='production'?'./':'/', |

2、vant适配Rem

vant适配rem,官网有配置说明例子，一定要把所有文件都看一遍，它的说明可能没全部说出来

（1）装两个npm包

npm i -S amfe-flexible

npm i postcss-pxtorem@5.1.1 --save-dev

注意：npm install postcss-pxtorem --save-dev装成开发依赖，直接装最新6.0.0会出现一下错误，装个低版本就行，装了5.1.1没问题

Error: PostCSS plugin postcss-pxtorem requires PostCSS 8.

（2）在vue.config.js中添加

|  |
| --- |
| const autoprefixer = require('autoprefixer')  const pxtorem = require('postcss-pxtorem')  module.exports = {  css: {  loaderOptions: {  postcss: {  plugins: [  autoprefixer(),  pxtorem({  rootValue: 37.5,// 适配rem，配置基准值，以37.5  propList: ['\*']  })  ]  }  }  }  } |

（3）在main.js中引入

import 'amfe-flexible'

注意：flexible的方案好像弃用了，使用的是amfe-flexible，在适配vant已经完成好了rem的配置

3、移动端的click的处理，它需要fastclick插件使用，因为它有延迟，和touch会有点透情况，在npm上有相关的使用方法

（1）在main.js中

import attachFastClick from 'fastclick'

attachFastClick.attach(document.body)

（2）在滑动区域会报错

加上touch-action：none解决报错

4、引入vant dialog报错,后续在搞了。。

用this.$dialog.confirm（）。。。。。

要使用Dialog的话需要在对应vue去import然后注册。要有引入Dialog才能使用

在配置文件中搞不了，官方说的有点模糊

5、懒加载一直搞不了，不成功？？？答：不要用img图片那里的，它直接有个懒加载模块，用那个

6、骨架屏的处理

注意：骨架屏vue-content-loader不能装最新的，兼容vue3的，一开始版本装错了

For Vue 2, use vue-content-loader@^0.2 instead.

什么时候隐藏骨架屏也是有讲究的

修改布局前，先看原来的样子，修改后有问题，才方便快速定位，不然很难处理

7、要注意有一些vant的组件是跳转用的url，不是hash路由，它是有区别的，注意看，一般是有url和to

8、i18n如果没有写错的话，是没有问题的，在元素的属性中注意使用单引号 $t('xxxx')

知识点

1、插槽

（1）缩写

跟 v-on 和 v-bind 一样，v-slot 也有缩写，即把参数之前的所有内容 (**v-slot:**) 替换为字符 **#**。例如 v-slot:header 可以被重写为 #header：

|  |
| --- |
| 如果你希望使用缩写的话，你必须始终以明确**插槽名**取而代之：  <current-user #default="{ user }">  {{ user.firstName }}  </current-user> |

1. 作用插槽

插槽内容能够访问子组件中才有的数据是很有用的，在template内部可以使用，例如在element ui中，能**获取到当前组件id什么**的。子组件最终是由父组件来进行渲染的。

（3）具名插槽：就是用**名字**来区分插槽的位置

2、生命周期函数中的this

生命周期钩子的 this 上下文指向调用它的 Vue 实例（因为它是在构造函数中，构造函数中的this，指向的就是new创建出来的对象，这也是为什么不能用箭头函数，因为箭头函数中的this，是来自 new Vue() 这一层的）

|  |
| --- |
| new Vue({  data: {  a: 1  },  created: function () {  // `this` 指向 vm 实例  console.log('a is: ' + this.a)  }  }) |

所有的生命周期钩子自动绑定 **this 上下文到实例**中，因此你可以访问数据，对 property 和方法进行运算。这意味着你**不能使用箭头函数**来定义一个生命周期方法 (例如 created: () => this.fetchTodos())。这是因为箭头函数绑定了父上下文，因此 this 与你期待的 Vue 实例不同，this.fetchTodos 的行为未定义。

因为箭头函数并没有 this，this 会作为变量一直向上级词法作用域查找，直至找到为止，经常导致 Uncaught TypeError: Cannot read property of undefined 或 Uncaught TypeError: this.myMethod is not a function 之类的错误。

（1）父子组件的生命周期顺序

想获取父组件通过自定义属性传值的方式，只能在mounted时候才能获取，在created是获取不到，注意父子组件的生命周期执行顺序。

（2）props命名的方法

HTML 中的 attribute 名是大小写不敏感的，所以浏览器会把所有大写字符解释为小写字符。这意味着当你使用 DOM 中的模板时，camelCase (驼峰命名法) 的 prop 名需要使用其等价的 kebab-case (短横线分隔命名) 命名（在父组件html传值时候）：

1. 页面需要不同屏幕分辨率适配。

4、移动端知识补充（需要转移到对应的笔记中）

（1）硬件分辨率（出厂就决定了）

也称为“物理分辨率”是生产屏幕时就固定的了，它是不可被改变的。window.screen.width就能看到。

（2）缩放调节的分辨率（软件设置，如pc端可以进行缩放）

也称为“逻辑分辨率”是可以由软件上去决定的，例如可以进行放大（如PC端能够调）

写代码的时候参考的是逻辑分辨率，如果是物理的话，那同个网页版心，左右留空就不一样

（3）视口

使用meta标签设置视口宽度。默认情况下，网页的宽度和逻辑分辨率不相同。

（4）设计稿

设计稿，一般以iphone6/7/8(逻辑分辨率375x667)

设计稿一般是一张图片，假如设计稿是375的，你把它作为背景图，放到iphone上，能充满，但是会被放大，模糊了。所以设计稿会给你750的。

多倍图是为了让图片不模糊，不失真。

像素大厨----选择开发模式---- 选择2x图模式，就能看到图片是375px的

图片是属于软件上东西，图片的分辨率应该是逻辑的。它跟屏幕的物理分辨率有关系。

图片大于屏幕的物理分辨率，系统会有自己的算法去裁剪图片，让图片全部显示出来，肉眼看不出的。

图片小于屏幕的物理分辨率，系统会有自己的算法去增加图片数据，让图片全部显示出来，如果差得比较多，肉眼能看出来图片模糊，失真。

（5）布局

百分比布局，也叫做流式布局

特点：宽度自适应，高度是固定的（过去的年代常用方案，现在基本要宽高都要变化）

一个页面在适配移动端的时候哪一些要进行处理，哪一些不要。。视频说的是一切（图片，间距，字体等等）都要进行等比例的缩放

（6）移动适配

rem：目前多数企业在用的解决方案

vw、vh：未来的解决方案（大厂目前在使用了， b站，jd等等）

（7）rem单位

相对单位

rem单位是相对于**HTML标签的字号**计算结果

1rem = 1 html字号的大小（font-size）

手机屏幕大小不同，分辨率不同，如何设置不同的HTML标签字号？？？

答：媒体查询，它能检测视口的宽度

注意：笔记本屏幕的分辨率设置，它可能推荐了缩放120%，这样可能会导致media检测出现问题。一般都是写范围没什么问题，如精确width: 320px,有可能会检测不到。使用device-width就没问题

设备宽度不同，HTML标签字号设置多少合适？？？

答：目前rem布局方案中，将网页等分成10份，HTML标签的字号为视口宽度的1/10

如320px取32px，375px取37.5px，这样好计算，但是具体元素给多少rem？如何确定rem的值

实际书写的时候，要根据设计稿中的内容来做，设计稿的尺寸是px

|  |
| --- |
| 举例子：  假如设计稿适配375px的视口，有个盒子宽68px，高29px,那么68像素是多少个rem  1、先确定设计稿对应的设备的HTML标签字号  查看设计稿的宽度--确定参考设备的宽度（视口宽度）--确定基准根字号（1/10视口宽度）  2、rem单位的尺寸 = 元素px单位数值 / 基准根字号  68 / 37.5 |

Android的视口一般是360px，有很多不同的视口，我们不能每个都去media一下

结合flexible.js配合rem实现不同宽度的设备中，网页元素尺寸等比缩放效果

1、flexible.js是淘宝（阿里）开发出来的一个用来适配移动端的js框架

2、核心原理就是根据不同的视口宽度给网页中html根节点设置不同的font-size

（8）vw/vh（不能进行混用）

相对单位

相对于视口的尺寸的计算结果

vw：viewport width

1vw = 1/100视口宽度

vh：viewport-height

1vh = 1/100视口高度

它直接就能用，不用写其他的代码

5、input标签的属性

value属性：用户输入的内容，提交之后会返送给后端服务器

name属性：当前控件的含义，提交之后可以告诉后端发送过去的数据是什么含义

后端接收到的数据的格式是：name的属性值 = value的属性值

（1）vertical-align解决的问题

1、文本框和表单按钮无法对齐的问题

2、input和img无法对齐的问题

3、div中的文本框，文本框无法贴顶的问题

4、div由img撑开，img标签下边会存在额外间隙的问题

6、mounted生命周期函数

注意 mounted ****不会****保证所有的子组件也都一起被挂载。如果你希望等到整个视图都渲染完毕，可以在 mounted 内部使用 **[vm.$nextTick](https://cn.vuejs.org/v2/api/" \l "vm-nextTick)**：（感觉似乎没有，在数据变化后，再使用nextick就没问题）

HomePage.vue中在mouted中，想获取li的长度，它是用v-for渲染，它获取的时候的是为空的，但是看console.dir有数据，这个数据可能是后面动态补上的，可以看它获取的数据是空的，后面换到获取数据哪里，用来**修改数据之后，再使用this.nextTick()**，是有（12）这样的

在下次 DOM 更新循环结束之后执行延迟回调。在修改数据之后立即使用这个方法，获取更新后的 DOM。

为了在数据变化之后等待 Vue 完成更新 DOM，可以在数据变化之后立即使用 Vue.nextTick(callback)。这样回调函数将在 DOM 更新完成后被调用。

7、padStart()函数的使用

ES2017 引入了字符串补全长度的功能。如果某个字符串不够指定长度，会在头部或尾部补全。padStart()用于头部补全，padEnd()用于尾部补全。

|  |
| --- |
| 'x'.padStart(5, 'ab') // 'ababx'  'x'.padStart(4, 'ab') // 'abax'  'x'.padEnd(5, 'ab') // 'xabab'  'x'.padEnd(4, 'ab') // 'xaba' |

上面代码中，padStart()和padEnd()一共接受两个参数，第一个参数是字符串补全生效的最大长度，第二个参数是用来补全的字符串。

如果原字符串的长度，等于或大于最大长度，则字符串补全不生效，返回原字符串。

|  |
| --- |
| 'xxx'.padStart(2, 'ab') // 'xxx'  'xxx'.padEnd(2, 'ab') // 'xxx' |

8、created中异步请求数据，mounted的时候去获取数据，读取数据

created中的数据是异步去获取的，那么，在mounted的时候，挂载的数据就是最原始的，也就是我们设置的初始化，在mounted时候，数据并没有获取到，因为异步要等主线程页面全部完成后，在去获取数据，这个过程很短暂。

如果说是同步的，那么在mounted我们就能获取created中的数据。

注意：对一个存在的对象去使用 xxx.abc，虽然abc不存在，但是这个操作是没有任何问题的，abc的值是undefined。这也是解释在页面渲染后会报错undefined的问题，如果是三层xx.abc.d，那么d会报错，因为undefined不能去继续点了，后边它异步获取到数据之后，就能渲染出来了，这个时间很短暂，可以通过在created或者mounted中打印，如果是数组也是xxx[0].d，在页面创建，它获取xxx的时候，只是一个空数组，空数组是无法点的

可以通过条件渲染去做

或者在data中设置那么的数据结构，可以去点

9、高手代码里看到.slice(0)，查了下这样写的好处：

（1）对原数组进行深拷贝，这样进行一系列操作的时候就不影响原数组了

（2）将类数组对象转化为真正的数组对象：var anchorArray = [].slice.call(document.getElementsByTagName(‘a’), 0);

//用[]这原生数组提供的slice方法，将document.getElementsByTagName(‘a’)返回的类数组对象转化为真正的javascrpit数组对象。

10、路由传参的问题：首页中的分类点击是不需要监听的，它直接把数据传过去，直接用来用就可以了。

拓展：跳转过去后，数据过去了，但是监听那一块，是在参数接收到后之久才创建的。（初步这样理解）

它是什么时候开始监听的？？？

按照自己的想法，它应该是这样，首先query它是空的，没有值，然后我在跳转时候给了值，按道理它监测到变化，但是没有，它需要进行再一次变化才能监测。后续研究？？？

初步那个值可能是作为初始值了

11、js数组的遍历有多种方法

（1）findIndex

对于数组中的每个元素，findIndex 方法都会调用一次回调函数（采用升序索引顺序），直到有元素返回 true。

只要有一个元素返回 true，findIndex 立即返回该返回 true 的元素的索引值。

如果数组中没有任何元素返回 true，则 findIndex 返回 -1。

12、在有父子元素都有点击事件的时候，子元素要设置.stop，阻止冒泡行为，不然效果会有问题

13、在动画中不要去操作dom元素，el就是dom元素，操作el就可以，它就是那个切换显示的那个元素，它针对是某个元素，不是整体

14、ref=====================================

ref 被用来给元素或子组件注册引用信息。引用信息将会注册在父组件的 $refs 对象上。如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果**用在子组件上，引用就指向组件实例：**

当 v-for 用于元素或组件的时候，引用信息将是包含 DOM 节点或组件实例的数组。

关于 ref 注册时间的重要说明：因为 ref 本身是作为渲染结果被创建的，在初始渲染的时候你不能访问它们 - 它们还不存在！$refs 也不是响应式的，因此你不应该试图用它在模板中做数据绑定。

15、在 ES6 之前，社区制定了一些模块加载方案，最主要的有 CommonJS 和 AMD 两种。前者用于服务器，后者用于浏览器。ES6 在语言标准的层面上，实现了模块功能，而且实现得相当简单，完全可以取代 CommonJS 和 AMD 规范，成为浏览器和服务器通用的模块解决方案。

细节问题

1. 样式忘了加分号，报错没怎么看，只是单单报了语法错误

2、如果设置样式出问题，一定先检测样式，看是否有哪里冲突了，或者误设置了，如img，最好指定是谁的，不然会影响很多的

3、移动端的图片与文字对齐，可以采用flex布局是最简单

4、注意：vue/cli编译的端口一般是8080，有时候会变化，注意了，有时候是8081。。。。。。

5、border会增加行高的，字体加行高，要根据具体情况，看加与不加，加了会引起一些对不齐的情况，让字体去撑开就行。

1. 如果有错误先看看网页的提示，vscode输出的可能是自动添加了什么代码。例如func，这个老是会添加一些东西

7、直接使用它里边提供的函数能直接进行跳转。。。。自己无法写，因为能设置top后，没有办法回来，之前是自己控制整个scroll的。

一定要去官网看，看看有有没有对应的方法，插件已经帮我们处理好了。。还是挺方便的。

8、要看组件上整体所有设置的条件，都可能会影响到。看v-if就影响了，但是它是在父元素上的，一开始没有注意到

9、在DetailProduct <!-- 最初始化的数据是个空数组，数据还在异步获取当中，所以等数据来了再创建，初始化创建的时候会报错 -->

一定要先解决报错的顺序，先报错先解决，它会影响后边的执行。不能从后面开始解决

10、在vue ui中出错了，在命令行没有出错。。。。。。有可能项目打开错了

11、a标签出现红色，后续处理，初步判断说visited引起，把它color设置成透明的

12、搞错样式中的文件。。。

13、注意数组获取的时候，要看你想要获取什么，最好是检查一遍

this.address = this.userAddress[this.index]

写成下边的就无法实现效果

**this.address = this.userAddress**

14、在一些需要对齐的布局的时候，可以找左边left进行基准，右边可能是对不齐的

15、居中的方法，如果有定位，需要居中的话，left 0 right 0 ;margin: 0 auto可以尝试

16、sessionStoreage得到的都是字符串

17、Boolean类型转字符串false变成true了

18、路由导航一定要有个next（），不带参数的来结束钩子，否则会一直跳转，导致栈溢出

19、有时候数目算出来有点bug，会显示很长，可以用.toFixed（x）来取小数后几位

20、不要过于依赖vscode的补全，会有问题的

方法思路：

1、分类页面的制作思路

先获取相关的高度，ul，li，wrarper,索引

（1）怎么获取索引

直接在click中传递索引。

通过e来获取属性值index，自己设置的（采用这种方式）。

li是由span撑起来的，点击事件要给span才行。

li，sapn类样式的话，使用类名，跟三元表达式结合。

有个坑要注意了，就是通过Attribute获得的索引，已经变成字符串了，不是数字。这点要注意。不要太在意第一次的结果，因为初始化时候给的是0，巧合的跟index为0搭载一起。

2、滚回顶部

返回顶部的话，用window做，只有监测window才可以

返回的时候，要慢慢回去，不然一下就跳到最前

设置scrollTop的值

页面路由跳转后，页面没有在顶部，是因为前个页面滚动了，所以可以在小火箭那里处理一下，重置一下。第1、3、5个页面是超出视口，刚好在1，5有小火箭，所以如果3滚动的话，跳到1，它在mountd置回顶部的，设置scrolltop，其他没有超出视口高度

3、切换分类的时候，下拉菜单需要进行折叠问题：

（1）没有下拉，切换到需要下拉的情况

这个它的流程是这样的，监听到数据的变化，在下一次dom刷新时候，判断菜单是否超出可视区，超过的话给false，没超出就给true，显示下拉菜单选项。在切换分类的时候，由于初始化的第一个分类它是不需要下拉的，然后切换下一个时候，直接去读取ref，是读不到（报错了，后续无法测试），需要等到改为true后的，下一次nexttick，回调再去读取ref，把菜单叠起来。这样保证没问题。

（2）下拉，切换到没有的下拉情况

这个直接转为false的情况后，直接return，并不会再去读取ref，所以没问题。

4、vue瀑布流可以懒加载

忘了给组件props传值，会出现以下错误

Invalid default value for prop "data": Props with type Object/Array must use a factory function to return the default value.

有时候刷新了还是原来的图片，在没有加上nexttick的情况下

watch: {

foodListData: function () {

this.$nextTick(() => {

this.$waterfall.forceUpdate()

})

}

},

数据的变化，在dom上的数据并没有更新上去

5、滑动那里获取数据时候，它的offsetwidth是整数，但是实际可能会超出一点，所以采取加1px

6、编辑地址的思路

添加地址后，把数据直接存放vuex中的一个数组中，然后在我的地址页面，会去判断vuex地址数组有数组，有就全部显示出来。

在修改地址时候，通过hash路由传索引，然后在修改页面，用vuex中的地址数组对应索引的值，作为地址编辑的初始值。

如果用户保存的话，再把vuex地址数组中的索引的对象，直接替换掉，因为不知用户修改了哪些，简单粗暴直接替换就ok

7、购物车的处理逻辑

（1）先处理加减号处理,加的话，主要操作vuex中的数量信息，自增

input数量的处理，一开始直接跟input绑定起来，input它输入的字符，会影响挺多的

（2）加入购物车，就是在vuex中的数组添加的元素

（3）右上角删除选中的商品，他们是位于不同的组件，是属于父子组件的关系，涉及了传值，其他也不用传值，通过在商品信息给个标志位checked。

（1）选中的就删除，点一次取反一次就行。还会去判断是不是全选了

（2）全选的话，有api全选，不能用点击事件了，全选不会触发点击，设置vuex中的状态都为true。

（3）全不选的话，跟全选一样的处理就行。

（4）价格的计算是根据checked，然后计算它有多少件，它的数量有多少来进行计算

8、显示我的地址，先添加地址，把地址信息添加到数组当中，然后用vant组件进行渲染，这样也知道修改哪个地址信息的索引

9、图标同步的问题，分类图标跳往分类，去逛逛跳往首页。。。。computed计算属性，还能进行监控，还有用￥store.state这样也能监控，这样最原始的方法。

去逛逛的话，它原本是个路由的，把它改成按钮来跳转了，主要是修改vue的state

首页的跳转，它是路由，会携带参数过去，在分类页面创建的时候，会判断你是图标切换过来的，还是在首页图标切换的，在判断中来使得图标同步。修改vuex

在切换tabbar图标的时候也要去修改vuex，不然他的值一直不会变化的，这样在首页到分类，分类到首页，一直是1会造第一次有效，后边跳转就无效了

难点：

1、better-Scroll滑动卡顿的问题，左右手势引起的

禁止左右滑动手势，它会影响到betterScroll滑动的使用，只能在这里禁用它，不可能让用户去设置浏览器不能左右滑屏。

在PC端模拟器无法检测这个问题。

禁用浏览器默认的滑动切换页面事件

|  |
| --- |
| css方法：  html{  touch-action:none;  touch-action:pan-y;  } |

处理思路：一开始使用这个betterScroll插件的时候，它在真机上现象有时能滑动，感觉很卡顿，不正常，在PC端测试又是正常的，这个就让我比较疑惑，初步判断是配置出了问题，一直从修改配置入手，改它回调函数速度，验证移动的距离，都没有效果。

后边在测试的时候，感觉一滑动，手指一用力，页面的切换了，比较难测试。通过css把滑动切换给禁止，测试了一下发现就正常了，后来确认是会影响的，有点偶然解决。PC端跟移动端还是有比较多区别的

看官方的例子使用，不要直接看用法，有可能是不齐

2、购物车动画的处理

（1）图片一直无法显示

图片一直出不来的原因是，因为那个图片是在ul中的，ul外层给了overflow:hidden，导致溢出部分没了

一开始考虑的是跟着图片一起移动的，后边发现绝对定位，设置了z-index也没法跳出，overflow:hidden的控制

（2）实现图片的单次动画的思路，这个思路无法实现多次点击

移动 距离一直不对，对transform:translate的用法理解有误，其实只要知道小车的坐标就ok，

一开始想错了，就复杂化了。

一开始把样式是用js控制style，不能控制rem的大小，后边添加类，去除类，这样也简单，也能适应rem

换成类之后，测试不行。。。后续再研究，应该是可以的？？？

移动后又出问题，做动画的图片不要放在滑动里边，容易出问题，没有滑动是正常的，滑动后有问题，把它拉出来解决

动画必须做一个节流，不然点击多次会出问题------

还是会出现点击多次，然后卡住的问题。。。。。。多次点击，快速点击

（3）实现多次点击的思路

加入购物车动画的分析：（图片经过加入购物车按钮时，会挡着按钮了）

1、用户可能会点击很多次，设置一个大小20的数组a来触发图片移动的动画，都初始化为false（这个数组的作用类似于队列，你处理完就出去，不要在里边占位置，这样的话，达到循环利用的效果，支持最多一次性20次点击，一般来说，一个动画是1.5s，在1.5s内不会点到20次），再准备一个数组b，用来存储正在移动的图片。

2、点击购物车按钮时，先遍历数组a，找个的false的位置，加上图片的地址，设置为true，并把这个元素值（对象）加入数组b

3、beforeEnter钩子函数，它处理的是图片的初始化位置，它的参数el就是false变成true的那个dom对象，初始化位置，跟图片大小。这里不需要自己去获取dom对象，也不用去关心是谁变了。它是谁从false变true，这个el就是它。

4、enter：加上过渡效果，和最终的位置，移动到小购物车的位置，并且透明就可以,这里不加done（）,我们下边就不用监听完成事件了，让系统帮我们去处理

5、afterEnter: 动画完成后，通过数组b，shift（shift的作用把数组的第一个元素从其中删除,并返回第一个元素的值。）得到第一个元素，把它重置为false,也就是重置数组a，有了空位。

（4）实现购物车图标的弹动效果

1、在home页面准备动画类（这个类不要写到别的组件去了），正常---缩小---正常---放大---正常，这样的动画效果，类似弹动。还需要给组件一个id，不能用ref，因为它是组件来着。

2、在homePage中，在图片动画结束的钩子那里，给dom元素添加类，500ms后移除类。

3、在PC端调试没有问题，手机端空白问题，突然出现？？？build之后也是访问不了，明天再处理了，答：没有处理null值引起的，现象比较巧妙

登录怎么有问题，那段回调显示的需要看下，答：是因为页面刷新后，vuex的数据没了，所以去匹配sessionStoreage后，名字为空。

要考虑空值的处理，页面一开始没有sessionStorage的，读取了null，没有处理，传了一个空值给vuex中store，然后去读取null的属性就报错了，不然会报错，有值的情况下不会有问题。

隔天发现的

疑问

1、css改其他组件样式，有的需要加deep，有的不用加也能改

|  |
| --- |
| 1、全局修改-直接当前组件页面修改，创建一个没有scope的标签内部直接修改（弊端可能会影响整个项目的样式，如果保证自己类名起名独一无二，然后从左外层自己可以选择到要修改的选择器。）（不推荐）  <style></style>  2、/deep/ 加scoped  <style scoped>  .a {  ::v-deep .b{  样式  }  }  </style>  3、深度作用选择器  如果你想修改组件的样式就可以使用 >>> 操作符：  <style scoped>  .a >>> .b { /\* ... \*/ }  </style>  上述代码将会编译成：  .a[data-v-f3f3eg9] .b { /\* ... \*/ }  有些像 sass、less 之类的预处理器无法正确解析 >>>。这种情况下你可以使用 /deep/ 或 ::v-deep 操作符取而代之——两者都是 >>> 的别名，同样可以正常工作。 |

2、vuex中mutations能带多少的参数

|  |
| --- |
| 提交载荷（Payload）  你可以向 store.commit 传入额外的参数，即 mutation 的 载荷（payload）  在大多数情况下，载荷应该是一个**对象**，这样可以包含多个字段并且记录的 mutation 会更易读： |

3、v-for,如果我把其中的一个数据删除，那index会不会变化

|  |
| --- |
| <template>  <div v-for="(item, index) in list" :key="index" >{{item.name}}</div>  </template>  const list = [  {  id: 1,  name: "Person1"  },  {  id: 2,  name: "Person2"  },  {  id: 3,  name: "Person3"  },  {  id:4,  name:"Person4"  }  ];  v-for  此时，删除 “Person4” 是正常的，但是如果我删除 “Person2” 就会出现问题。  这个时候，除了 Person1 之外，剩下的 Person3、Person4，因为被发现与相应 key 的绑定关系有变化，所以被重新渲染，这会影响性能。  如果此时 list 的 item 是 **select** 的选项，其中 Person3 是选中的，这个时候 Person2 被删除了，用 index 作为 key 就会变成是 Person4 选中的了，这就产生了bug。  如果使用唯一id作为key，删除 Person2 后，剩下的元素因为与 key 的关系没有发生变化，都不会被重新渲染，从而达到提升性能的目的。此时，list 的 item 作为 select 的选项，也不会出现上面所描述的bug。 |

\*4、使用vant中的粘性布局，自己使用fixed去布局，会导致滑动出问题，暂时无解

5、用常量替代Mutations事件类型，怎么做

|  |
| --- |
| 使用常量替代 mutation 事件类型在各种 Flux 实现中是很常见的模式。这样可以使 linter 之类的工具发挥作用，同时把这些常量放在单独的文件中可以让你的代码合作者对整个 app 包含的 mutation 一目了然：  // mutation-types.js  export const SOME\_MUTATION = 'SOME\_MUTATION'  // store.js  import Vuex from 'vuex'  import { SOME\_MUTATION } from './mutation-types'  const store = new Vuex.Store({  state: { ... },  mutations: {  // 我们可以使用 ES2015 风格的计算属性命名功能来使用一个常量作为函数名  [**SOME\_MUTATION**] (state) {  // mutate state  }  }  })  用不用常量取决于你——在需要多人协作的大型项目中，这会很有帮助。但如果你不喜欢，你完全可以不这样做。 |

6、o2o是什么意思

|  |
| --- |
| o2o模式是指将线下的商务机会与互联网结合，让互联网成为线下交易的平台，英文是Online To Offline。 |

\*7、在动画中不要去操作dom元素，el就是dom元素，操作el就可以，它就是那个切换显示的那个元素，它针对是某个元素，不是整体

// this.$refs.imgRef[pic.index].style.transform = `translate(${imgPosition.x}px, ${imgPosition.y}px)`

el.style.transform = `translate(${imgPosition.x}px, ${imgPosition.y}px)`

为什么上一句话会有问题？？？

8、购物车弹动的，后边再处理？？？

1. 为什么有的图片要require呢？？？？

|  |
| --- |
| 1、在img标签里面直接写上路径：  <img src="../assets/a1.png" class="" width="100%"/>   1. v-for用:src的方式渲染图片，此时就要require，或者是直接是网络地址   <el-carousel-item v-for="item in carouselData" :key="item.id">  <img :src="item.url" class="carouselImg"/>  <span class="carouselSpan">{{ item.title }}</span>  </el-carousel-item>  data: () => ({  carouselData:[  {url:require('../assets/a1.png'),title:'你看我叼吗1',id:1},  {url:require('../assets/a3.png'),title:'你看我叼吗2',id:2},  {url:require('../assets/a4.png'),title:'你看我叼吗3',id:3}  ]  }), |

10、箭头函数，普通函数 ES6中的函数分别怎么写？？？

|  |
| --- |
| ES6 之 对象的简写方式  简写有两条基本原则：  1、同名的属性可以省略不写  2、对象中的方法中的 : function 可以省略不写(允许我们省略对象方法的function关键字及之后的冒号)  来看下下面这个例子，我分别用ES5 和 ES6 的语法分别定义并声明了一个简单的学生对象：  ES5:  var studentES5 = {  name: '小方哥',  age: 20,  sex: '男',  getName: function () {  return this.name;  }  }  console.log('ES5', studentES5);  console.log('ES5', studentES5.getName());  ES6:  const name = 'Jack';  const age = 25;  const sex = '女';  const studentES6 = {  name,// 同名的属性可以省略不写  age,  sex,  getName() {// 可以省略方法中的 : function  return this.name;  }  };  console.log('ES6', studentES6);  console.log('ES6', studentES6.getName()); |

\*11、组件的生命周期是怎么样的，切换就销毁了吗，进入就挂载？？？不会一直在内存中的，路由，在别的页面进行刷新，home一直是存在的Locale.use('en-US', enUS);

加上\*号等换电脑时候在解决下