1. vue做代理
2. <meta name="viewport" content="width=device-width, user-scalable=no,

initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0,viewport-fit:cover">

viewport-fit:cover /// 解决刘海屏问题

1. flex布局
2. justify-content: center; // 水平居中对齐
3. align-items: center; // 垂直居中对齐
4. 1px的物理像素问题，解决不同像素下的二倍图与三倍图的问题
   1. 解决方案在minix.scss中
   2. 背景图使用minix解决
5. minix --- css的复用机制
6. 项目的启动
   1. 1. 安装mogodb服务器
   2. 2. 启动mogodb.exe文件
   3. 3. npm start 启动成功
   4. 4. 启动client 即可访问
7. rating评分组件
8. 怎么看父子路由关系
   1. 1. 父组件中可以包含子组件的内容，才被称为父子路由的关系。
   2. 2. 如果是一个单独的页面则是一级路由。。。。布局关系就是路由关系
9. 写页面的时候要考虑到路由之间的关系
10. vue中使用sass
11. 这个项目中使用的是stylus
12. 安装stylus-loader
13. <style lang="stylus" rel="stylesheet/stylus" scope>
14. 即可
15. 引入reset.css和fastClick与阿里的iconfont在index.html中
16. 拆分组件并配置vue-router
17. .在router文件夹中配置路由。
    1. 1. 引入vue，引入vue-router，Vue.use(VueRouter)
    2. 2. （将其他的路由组件导入进来。） -- 为了第三步映射
    3. 3. export default new Vue( { routes : XXX } ) 即可
       1. 这里将routes进行拆分
    4. 4. 在main.js的vm中映射路由
18. 大体的架子配好之后，就可以向每个组件中填结构和样式了。
    * 1. ！！！开始拆分每个页面的静态组件

小知识点：

* 1. ---- vm.$router.push()
     1. 跳转到不同的url，会向histroy栈中添加一个记录。--点击后退会返回上一个页面
     2. 用法: vm.$router.push({ name: 'home' }) // 编程式路由导航
     3. vm.$router.push({ name: 'home' , params: { userId: 123 } }) // 带params的编程式路由导航
     4. vm.$router.push({ name: 'home' , query: { userId: 123 } })
        1. 带query的编程式路由导航
     5. 或者 --- 字符串的用法 这样写带不了参数
     6. vm.$router.push('/home')
     7. 或者 ---- 使用对象，但是使用path属性
     8. vm.$router.push({ path: '/home' })
  2. vm.$router.replace()
     1. replace的用法与push相同
     2. 跳转到不同的url，不会向histroy栈中添加一个记录。 --- 点击后退会返回上上一个页面

Pages中路由组件，component中是非路由组件

下面开始拆分静态组件。

1. 开始填充footer的结构和样式
2. 关于footer导航跳转的问题
   1. 绑定点击事件传入当前的path，使用编程式路由导航进行跳转
   2. 关于导航样式的改变，使用判断:class={ on: ‘/msite’==$route.path }
3. 填充外卖，搜索，订单，我的这四个的结构和样式
4. Header抽取成一个组件，这个组件大部分都相同，但是有一些部分是不同的，所以使用插槽。

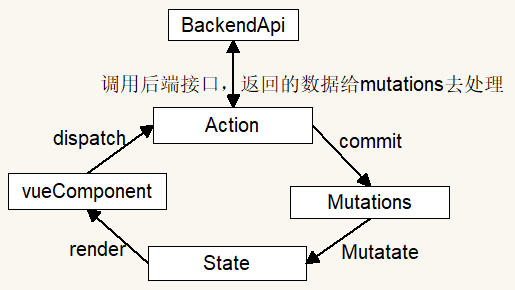
插槽就是将元素与数据一起传给子组件。，子组件定义插槽占位符进行接收，如果传入则替换掉占位符，如果不传入，则不显示插槽。

* 1. 插槽的使用：
     1. 在header组件中定义两个插槽，name叫left和right
     2. 首页需要将元素与数据传入到插槽中，则在传入的元素标签上分别写上属性名: slot=”left” slot=“right

1. 将header抽取成全局组件。在main,js中 使用 Vue.component()语法”
2. 将商家列表抽取成组件，不过不是全局组件，现在只有首页在使用
   1. 组件： 引入、映射组件、使用组件
3. 抽取login组件
   1. 当login组件显示的时候，不显示FooterGuide组件。
   2. 真正的判断是如果只有当路由是msite/order/search/profile时才会显示。
   3. 解决办法
      1. 将这个四个路由放到一个数组中，FooterGuide的v-show使用一个计算属性进行判断。
      2. 这里使用一个新的属性，路由中的meta属性。
         1. 给上述四个路由配置meta: { isShowFooter: true }
         2. 然后v-show：”$route.meta.isShowFooter”即可

小知识点：

点击去登陆 订单页面页面可以去登陆，我的页面也可以去登陆，登陆页面左上角有一个返回的按钮，绑定点击事件，在其子路由中设置replace。点击返回的按钮触发 $router.go(-1) 即可

1. 项目亮点
   1. 获取经纬度
      1. 应用场景： 签到功能
2. 修改eslint的配置，在packjson.json中留了一个“rules”: { “no-console”: “off”}
3. 3的脚手架的配置是写在vue.config.js中的，与packjson同级
4. Vue配置跨域代理。
   1. 请求的接口统一加上/api (baseUrl: ‘/api’)，以方便统一请求。在vue.config.js中配置代理。拦截到/api的请求，对其进行跨域处理。然后将请求接口进行处理(将/api替换为 ‘’)
5. 总结： 1. 搭建项目结构，拆分静态组件(分析结构然后才能进行拆分)，一般有上下、上中下、左右结构。先整体在局部。然后看路由，分析一级路由，二级路由，谁是谁的子路由
6. 学习vuex
7. 
   1. Vue中的单向数据流。数据渲染模板，用户通过一些操作让数据变化。数据变化之后会响应到模板中。
      1. 这就形成了view –-> action 🡪 mutations(data)--> view。
   2. State包含n个状态数据的对象 基本数据
   3. Mutations 包含n个用来直接更新状态数据的方法 的对象 同步！
      1. 一个方法就是一个mutation，mutation只能更新state的同步代码，
      2. 不包含逻辑，mutation由action触发commit(‘ mutatios中的方法名调用 ’)
      3. 一般是action获取请求的数据commit(mutation)添加到state中。
   4. Actios
      1. 类似于一个装饰器，包裹mutations,使之可以异步，间接更新数据
      2. 一个方法就是一个action，action中可以有逻辑代码和异步代码
      3. action由组件来触发调用。
         1. this.$store.dispatch(‘actionName’)
   5. Getter 类似于vue中的计算属性，从基本数据计算出的属性。是一 个 对象。可以直接获取里面的值。功能类似于state，不过是计算出来的值而已。
8. Axios的配置
   1. 封装一个自己的axios方法，将axios export default 出去。
   2. 然后再同级的index.js中引用，引入自己封装的axios，定义中获取接口数据的方法。将每个方法export 出去。每个方法返回都是一个pending状态的promise。在需要的地方对数据进行处理。这里是将数据全部存放到vuex中
9. Vuex的使用
   1. 下载vuex
   2. 单独建立一个文件夹，vuex，然后创建store.js
      1. Vue.use(Vuex) // 使用vuex，并(导出) export default new Vuex.Store({ state, mutations, actions, getter })
      2. 在main.js中的那个实例中，导入store.js，在Vue的实例中添加store
      3. 将store挂载到vue的实例中之后，即可通过this.$store来访问store对象中的各种属性和方法。即可拿到store中的数据，就可以在组件中进行渲染。
      4. 不可在组件中修改store中的数据，因为这违反了vue的单向数据流原则。
      5. 真正的项目中需要去拆分模块。
      6. 如果在一个组件中需要获取属性，需要在computed属性中映射mapState,
10. Vuex在项目中使用
    1. 将上面所说的vuex的大体架子搭好。Main.js能够映射store。
    2. Vuex与axios的配合使用。Vuex做状态管理。管理axios返回来的数据。
    3. 将需要请求的数据定义在state中。方便actions的获取。
    4. 再actios中获取axios返回来的数据。对数据进行执行
11. 具体： 要搞清楚vuex的数据流。
    1. State.js中是页面中需要使用的一些数据的仓库。将数据export出去。
    2. Mutation\_type.js中存放n个mutation中所需的函数名的集合
    3. Mutation.js 中定义方法，该方法定义state参数用来存储数据，定义第二个参数当作占位符，用来获取actions传过来的数据存储到state中。
    4. Actions.js 需要对数据进行处理。进行ajax请求，导入state.js中的数据，
       1. 虽然用不到，但是发现了，action中的state是异步的数据。
       2. state数据是共用的。一个改变了，全部都会改变

Import 导入的数据如果是一个字符串，要将这个字符串作为方法名使用，则要将这个引入的数据用[] 包起来。就可以作为方法名使用了

1. 获取this.$store.state中的数据是一个异步操作，用computed接收就可以了，
   1. 问题：问什么要用computed去接收就好了。、
   2. 记住 ，组件中去接收$store.state中的值，使用computed计算属性去获取
2. 知识点：
   1. 对象展开运算符 { …obj } // 首层深拷贝此对象
3. 出了个bug ，msite组件中titile不显示，点击其他的路由在跳转回来就好了。

因为dispatch异步的问题

开始： 分类列表异步轮播

1．使用swipper的时机，必须在列表数据显示出来之后new Swipper才有效。

2. 在组件中，什么才是数据被渲染出来之后呢？watch+nextTick

3. 在mouted中引用，即元素都被渲染出来，但是数据不一定被渲染。

再强调一边数据流，后台数据在接口中，要将后台的数据存放在vuex中进行状态管理，组件中需要获取vuex中的数据进行展示。

具体：(前提是已经做好了vuex的配置) vuex通过action进行异步获取请求数据，将数据commit到mutation中，mutation将数据存放到state中。再需要的组件中将数据分发出去。然后再任意组件内即可获取到stroe中的数据。(注意:在组件中computed中获取)

4. 问题： 如果将一个数组中每八个分成一个数组，将分成的数组放到新数组中。

获取到新数组的长度，然后循环新数组的长度，对数据进行splice截取。(0,size)

5. 动态绑定src属性很简单，但是一个静态的数据拼接一个动态的数据成一个src怎么做？

:src="'https://fuss10.elemecdn.com'+food.image\_url"

Swipper组件创建的时期： 轮播图不能正常显示的问题

1. 创建的时机过早，必须在列表显示之后。意味着必须视图与数据全部显示才可以。因为vue的DOM是异步更新的
   1. 解决方案，
      1. 使用watch+nextTick，
      2. await this.$store.state.dispatch(‘receive\_food\_category’)
         1. 这个方法返回的是一个promise，将其异步执行。
         2. 但是还是不行。
2. 我试了这几种办法还是watch+nextTick无bug
3. Rating的解决方式，放在笔记中。
4. 商家列表使用骨架屏，骨架屏是三张svg图片。
5. 轮播图使用骨架屏，骨架屏是一张图片
6. 组件中的样式，如果组件中的样式放在父组件中，父组件的style属性如果加上了scope属性，则其样式不会渗透到子组件中。
   1. 将组件中的样式与结构图片等资源全部提取到该组件中。
7. 如果存在float计算的话，可能会存在误差，将第一次计算得到的float的属性值\*10 然后将比较的值\*10 进行比较。这样就可以减少误差

其实所有的路由都可以用div的 显示与隐藏做。

登陆:

1. 两种登录方式的切换与隐藏
2. 手机号格式验证
   1. 手机号格式验证正确才可以发送验证码
3. 倒计时的功能
4. 密码显示与隐藏
5. 前台表单校验
6. 两种登录方式的切换与隐藏
   1. 手机号登陆与密码登陆使用子路由的方式做。

定义pwd和tel组件。为登陆路由的子路由组件。定义router-link做切换，router-view做占位符。这两个路由时相互独立的。所以给每个组件最外层的盒子加上on类。

* 1. 也可以定义一个标记用来切换与隐藏。点击a标记置为true，点击b标记置为false 界面的显示与隐藏V-show:=’flag’ v-show=‘!flag’

1. 手机号格式验证
   1. 只有手机号格式正确，才可以发送验证码(将获取验证码的按钮不置为disabled)，而且验证码的颜色变为black。
2. 倒计时的功能
   1. 使用watch监听，当数据正确的时候，将按钮的disabled置为false，将验证码的颜色置为黑色(添加类)
   2. 点击发送验证码开始计时。获取验证码的文本变成倒计时的数字。
   3. 点击验证码的函数中，使用类节流阀，如果定时器id存在则直接return。

组件化开发的核心：

1. 界面的改变是根据数据改变的。数据改变了，界面就改变了。

在做项目的过程中，发现每个组件中的数据的设计很重要，虽然可以完成功能，但是我每次设计数据都太冗余了。要精简数据，通过计算的方式来对data中的数据进行精简

图形验证码

1. 点击图形验证码，局部刷新当前图形验证码
   1. 当点击图形验证码时，获取图片，改变图片的src属性
   2. 但是获取图片路径的地址是唯一的，就是如果给src赋的值相同的话，则不会重复修改数据。(因为浏览器缓存的原因)，所以重新赋值，不会重新发送请求，而是使用了上一次的缓存。这个时候需要设计一个时间戳。将时间戳放在url的参数上。
      1. This.$refs.captcha.src=‘http://localhost:4000/captcha?a=’+Date.now()

关于请求数据接口在哪里处理的问题：

如果需要将数据存放到vuex中，则在actions中进行交互。如果不需要将数据存放到vuex中，则不需要使用acitons中进行交互，将api/index在main.js中放到vue的原型链上即 后面需要使用到交互时，直接this.XXX即可调用接口数据

短信验证码

1. 使用容联云进行短信验证码服务，注册。然后将账户对应的sid，id，appi放到后台数据里面。还需要配置测试号码。然后即可在前台调用接口，即可完成短信的发送。

如果不分发(dispatch) 则不会将数据存到vuex中。

问题：

子组件需要向父组件中发送数据，但是触发的函数在父组件中。

根本就没办法发布事件。因为没有触发的时机。

组件设计错误，需要重新设计组件

引入mint-ui

登陆的操作：

1. 点击登陆，子组件中登陆。
2. 发送ajax请求，请求登陆结果，将登陆结果发送给父组件，父组件判断是否登陆成功。(这样做的原因是有两个子路由，如果分别去判断的话，会造成代码的冗余)。将登陆结果与刷新验证码的数据发送给父组件。
   1. 2.1 登陆成功，在跳转到登陆页面之前（该组件被销毁之前）。将定时器清除。
      1. 跳转路由后，该路由组件中的数据被销毁。但是还有一些东西没有被销毁，比如说定时器。
      2. 登陆成功之后，要将用户信息存放到vuex中(使用dispatch的时候，第二个参数可以将数据传过去)，跳转到profile路由，获取用户信息，根据不同的登陆的方式(判断user.phone是否存在)，显示不同的元素。

this.$store.dispatch('receive\_user',result.data)

* 1. 2.2 登陆失败，将刷新验证码的方法，传送给子组件，让子组件刷新验证码
     1. 用户名密码进行登陆失败后，验证码不会刷新，需要重新调用一次更新验证码的操作(验证码错或者密码错)，更新验证码

1. 实现登陆功能的自动登陆
   1. Token: token是一些包含有效信息的字符串，后台程序可以通过解码token取得用户的有效信息和token的过期时间，使用token对数据进行拦截。
      1. 如何生成token，登陆成功，使用当前用户的id与密钥和token过期时间使用jwt生成token。
      2. 由服务器端生成，由浏览器保存。
      3. 登陆成功，将生成的token与user信息放到响应体中一起返回，token也可以放在cookie中，但是token不太安全，而且浏览器端可以禁用
      4. 将token保存在localStorage中，为了下次的自动登陆，将token和user保存在vuex中，是为了在内存中反复使用
      5. 携带请求头token访问服务器，服务器进行token的解析，获取到token中的id信息，与服务器中资源进行比对，如果存在，则发送请求。不存在则提示token无效，拦截掉请求
   2. Token的操作
      1. 再请求拦截器中，判断token是否存在，如果存在，将token放在请求体中，发送请求。如果不存在，则对请求进行拦截。并抛出异常。抛出后的异常会在响应拦截器中进行错误的请求中处理。(这样的话连请求都不会发出去。)
         * 1. 请求拦截器抛出的异常会在响应拦截器中进行处理。
         1. 在响应拦截器错误请求中，提示用户无token的信息，并返回一个promise。
         2. error中存在哪些数据。？？？？？？
            1. Request，response，config，code，
      2. 登陆拦截器的具体操作
         1. 如果localStorage中的token 存在，则将token携带在请求头中向后台发送数据，
         2. 如果不存在，则抛出异常error。异常中保存一个数据status为401.
         3. 请求中出现的错误则在响应拦截器中的异常处理中进行处理。
            1. 请求出现错误有几种：

1. localStorage中的token未存在，则error中没有response。说明没有token，对其进行处理。

提示用户信息过期，跳转到登陆页

2. Token存在，error中有response但是token过期/别人伪造token。会将请求发出去而且有error中有一个status是401。则对其进行处理。

退出登陆

将localStorage中的token删除，然后将state中的user数据与令牌数据清空。

提示用户信息过期跳转到登录页。

3. 404错误

提示用户此资源不存在。转向404页面。

* + - 1. 不允许重复点击当前路由。
         1. 因为跳转到登录页之后，仍然还有请求因为没有token要跳转到登陆路由。

Ajax数据什么时候使用vuex获取，什么时候使用当前组件自己获取。

初始化数据，使用vuex获取，

如果是操作(需要当前页面数据的话)的话当前组件自己获取。

<router-view @fun="fun"></router-view>

这个才是子组件

登陆成功之后，需要清除定时器，否则会造成内存泄漏，那么怎么才算将定时器清空了呢？ 每次跳转路由之后，是不是都将本路由的数据全部清空了。

跳转路由之后，该路由组件被销毁。里面的数据被清空，但是手机号验证码定时器没有被清空，所以使用

  // 监听beforeDestory，当组件即将被销毁之前，清空定时器

        this.$once('hook:beforeDestroy',() => {

          console.log("触发了hook，beforeDestory")

          clearInterval(intevalId)

        })

进行销毁定时器。

在组件中使用vuex的strore将数据进行分发， 即可将数据存放到vuex中。

获取vuex中state中的数据。在组件中有一种很简单，很固定的格式。

Import { mapState } from ‘vuex’

computed : {

…mapState([‘user’])

……………….

}

出现了一个未知bug，登陆之后，reset.css引入不进去，刷新一下就好了。

将引入css文件的位置放在main.js中。

有些请求需要携带token，有些请求不需要携带token。在请求时，携带一个请求头needToken=true即可。不需要携带token时，则请求头不携带needToken。

这样只适用于一些情况。还有一种解决办法就是所有的请求除了登陆都携带token过去。

自动登陆的功能：

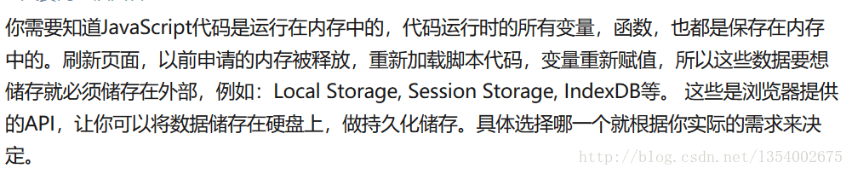
1. 问题: 将用户信息存在了vuex中，我一刷新数据就没了。Vuex中的数据就没了。

有一次我在做项目的时候使用vuex做状态管理，当时做首页和登录页，发现了一个很诡异的现象，vuex保存了首页的导航栏数据和一些列表信息，首页渲染将这些数据存放到vuex中，然后我做登录的时候，将数据也放在了vuex中，当我把登陆数据放在vuex中的时候，首页的信息就没了。当我重新加载首页的时候用户信息就没了。并且我点击浏览器的刷新按钮刷新页面的时候，用户信息就没了。

问题的根源：

在vue的单页面应用中，如果在某一个具体路由的具体页面下点击刷新，那么刷新后，页面的状态信息可能会丢失

问题1 ： 刷新页面代表了什么 为什么会丢失数据



所以需要实现用户自动登陆的功能

点击登陆将数据发送到login组件中，login组件分发actions。将数据存到vuex中。再profile组件中获取vuex中的user信息，展示到页面上。一旦刷新的话，vuex中的数据丢失了。所以获取到的user信息是undefined。

所以在profile组件中，通过分发自动登陆接口(自动登陆的接口虽然没有传参数，但是携带了带有token 的请求 ) 在拦截器中通过localStorage中的token数据获取用户的信息，然后将数据放到vuex中。然后在获取vuex中的信息，展示到页面上。

小知识点： dispatch是一个异步的函数，所以在dispatch的时候加上await，然后在获取stroe中的数据

问题: 如何解决dispatch的异步问题 ，除了async和await

如果使用了router-link，在router-link标签上写上replace的话，则意味这个路由不会向histroy栈中添加记录。可以做返回上一个级就是返回返回上一个页面。

在router-link 上添加了exact则代表，此路由只能被精确匹配.

在router-link标签上写上exact的话, 代表此路由只能精确匹配

拆分商家详情页面的组件。

子路由中path: ‘‘ 这个子路由是默认显示的路由。可配合redirect，可以默认显示子路由组件。

Vue-router之填坑指南1 一级路由中有子路由，子路由中的path有’’ 或 /

就是默认指向父路由的意思。这个时候父路由中不能有name。具体：

该路由需要一个默认的路由指向子路由的时候，该路由且不能定义name。

    {

        path: '/b',

        component: B,

        children: [

            {path: '/b/b1',component: B1},

            {path: '/b/b2',component: B2},

            {path: '/',redirect: '/b/b1'},

        ]

    }

上述完成的功能是： 父路由初始显示默认指向一个子路由

设计json数据

使用mock.js进行接口设计。

Mock的数据与真实的数据有什么不一样？

数据结构必须一样，数据值可以不同。但是数据值的类型 (字符串必须是字符串，数值必须是数值) 必须相同。

Import data from ‘./data.json’

这个data现在指的是js对象，因为webpack内置了json-loader，自动将json字符串转化成json对象

通过ajax发送请求，浏览器还没接收到，被mockjs拦截掉了

Mockjs使用教程：

1. 安装mockjs yarn add mockjs
2. 建立一个新的文件叫mockServer，引入mockjs
   1. Import Mock from ‘mockjs’
   2. 引入模拟的data.json中的数据
      1. Import ‘./data.json’
3. 使用Mock开始生成接口与数据
   1. Mock.mock(‘ /goods ’, {code : 0, data: data.goods})
   2. 即生成了接口/goods ，调用goods接口，即可获取goods数据。
      1. myaxios.get(‘/goods’) 获取的promise中存储了goods数据。
4. mock数据完成，无需导出，在main.js中运行一遍即可。
5. 拆分商家信息的静态组件，拆分样式。
   1. 头部组件 shopHeader
   2. 导航（不是组件）

显示导航路由

* 1. 购物车组件
     1. shopCart组件

1. mock数据
2. 将数据存储到vuex中。

问题：

如果ajax获取的数据需要子组件中显示，是在父组件中获取数据呢？还是在子组件中获取数据？

尚硅谷的老师是在父组件中将数据存到vuex中的。

目前我只能在当前组件中分发数据，因为必须async await才能获取state中数据。也不会用computed。

使用定时器测试过了，vuex中的数据获取不到就是因为异步的原因。

使用computed的原因就是computed是计算属性，如果异步的数据回来了，会自动改变当前数据

点击，让当前点击的元素加上current class，其他的清除current class

footGuilde 用的是class = “{ current: router.path===’当前路由’ }”

商家详情的折扣，优惠怎么做呢？

定义一个变量currentName，存放当前点击的good（就是折扣、优惠）的名字。

在li上判断 :class="{current: currentName === good.name}

给li添加点击事件

@click="currentClass(good.name)"

在事件处理函数中让this.currentName = good.name

赋值的时候直接写window.location相当于直接写window.location.href

解决多次点击路由报错的问题

 goto(path) {

        if(this.$route.path !== path) {

            this.$router.push(path)

        }else {

          // window.location = path

        }

如果重新点击当前页面需要重新刷新当前页面的话，

则else里面写window.location = path。这种方式在mode是history时有效。

也就是说路径中没有#。

但是如果路径中有#的话就不会重新刷新了。解决方案：

在else中写window.location.reload强制刷新即可解决

在其他的情况需要多次跳转到一个路由的话，只要在路由守卫或在请求中判断router.currentRoute.name === 当前路由的名字，如果是在进行跳转或者放行

解决所有的路由点击多次的出现NavigationDuplicated的问题。

在main.js中引入

// 解决了NavigationDuplicated的问题

const originalPush = Router.prototype.push

Router.prototype.push = function push(location) {

  return originalPush.call(this, location).catch(err => err)

}

1028号

问题解决了： 要在vuex中的state文件中将需要写的数据写上。这样就不会报错了。Vuex中异步的错误。

当渲染模板的时候，从vuex中取数据，如果只有一层则不会报undefined错误，如果有两层的话，则会报undefined错误。如何解决？

在最外层的div上加v-if=” info.support ” 即可因为如果info.support为假的话，根本就不会渲染页面。但是v-show的话不行。V-show还是会报错。

通过这次的错误明白了一个道理：

1. Vue中的数据响应是实时的，如果渲染模板需要的数据是异步的时候，需要给这个模板一个初始值。如果需要链式点这个对象两次的话，需要给这整个模板加上v-if。因为请求是很快的，人们的肉眼根本无法捕捉到请求发送之前模板上渲染的undefined。如果在网速慢的情况或者高并发的情况。可以做一块静态模板暂时展示，当请求的数据回来之后在展示数据。

问题： 遮罩层是怎么做的？

全部，吐槽，推荐

三个路由之间的切换。切换的样式使用class={‘active’: currentName====’quanbu/tucao/tuijian’}。点击的时候将当前的一个名字标识传给currentName即可。

切换路由达到数据的切换，使用过滤数据的功能。定义一个变量存储全部数据（专门用来过滤数据），定义一个变量专门用来展示数据（将过滤的数据赋值给需要此要展示的数据）。

列表满减、折扣、优惠的样式前三名的样式不同，后面的样式都一样，可以使用数组进行存储样式的类名，:class=”activeClasses[support.type]” 即可。

这又是一种解决方案，nice

betterScroll

new BetterScroll的时候，需要在列表显示之后。那么怎么才能知道列表显示了呢？

Watch goods + nextTick。监听goods的变化，如果goods改变了，则new betterScroll。 但是如果在别的路由点过来的时候，goods已经存在了，所以watch不到goods的变化，这个时候需要在mounted中判断如果goods.length>0则new BetterScroll,

但是我dispatch的地方就是在本页面。所以不用在mounted中new BetterScroll。如果是在shop组件dispatch的话，就需要在munted中new BetterScroll了。

联级滑动

滑动右侧列表，左侧当前分类自动变化，

滑动左侧分类项，右侧列表自动滑动到对应的位置

定义数据

存放每个li的高度的数组foodHeightArr。并且是叠加存放。

监听滚动事件，查找foodHeightArr数组，如果当前滚动的位置小于foodHeightArr数组中的某一位而且小于其下一位的值。返回foodHeightArr值的下标。赋值给currentIndex。

也就是 0, 750 ,1060,1506,1627,1843…… 获取当前滚动的距离，查找，在哪两个之间就将其前一个的角标赋值给currentIndex。

以上就实现了滑动右侧列表，左侧列表分类自动变化。

点击左侧分类。获取index。使用better-scroll中的一个方法。

scrollTo(x , y , time , ’easing’ )

其中x即为 0

y为foodHeightArr[currentIndex]

probeType: 3 手指滑动屏幕即使是惯性的滑动也可以进行触发滚动

probeType: 1 非实时的，只会在触摸滑动分发事件。

probeType: 2 实时的，只会触摸滑动才分发

滑动操作通过probeType: 1 + scrollEnd事件会有更好的性能。

功能： 左侧列表随着右侧列表撑在左侧的最上面

在右侧列表滑动滚动中监听。

// 将meun的li滚动到当前的 当前currntIndex的位置。

          this.menuScroll.scrollToElement(this.$refs.menu[this.currentIndex], 300)

ps: 就是多传了一个参数’easing’ ，找了很久的错误，可能是插件内部做了兼容性处理，也不报错。使用插件的方法时，要看清楚传递的参数。

问题： 创建了多个BScroll对象来管理一个dom元素

解决： better-scroll点击一次，触发多次的问题：  
 将new BScroll() 变为单例模式。

If(!this.bscroll) {  
 this.bscroll = new BScroll(‘ xxx’,{ })

}

使用better-scroll时 dom改变时，要刷新dom元素节点。

this.bscroll.refesh()

点击食品，显示此食品的弹出框，做成一个组件。

让div变得不可点击

pointer-events:none

主要是用来做div动画的时候，因为动画需要时间，如果用户手速很快的话，会在动画未结束之前，点击div继续触发事件。所以在在div上判断，当判断结果成立的时候，加上此样式。

发现这种样式会让此元素阻止冒泡的功能也没了。所以要让div变得不可点击。还是通过js来进行判断。在remove事件的开始处进行判断。如果count为0的话，则直接return出去。这样就即使点击了，但是也没效果。

购物车大概的思路。

数据结构： 一个总的订单数组。数组中每一项都是一个订单项。

订单项数据结构为： name,price,count

在vuex中的mutation中将数据push进去。因为要保证一个商品只有一个订单项所以要在mutation中对数据进行判断，如果order中存在了orderItem，(判断的标准是遍历order中的orderItem的将要被push进去的orderItem的name相等则将order中的orderItem的count替换为要被push进去的orderItem的count。)

shopCart组件：

显示的数据： 总价格、每个订单项(列表数据)----要显示的有(商品的count、名字、价格)----这是push之前自己组装orderItem的数据结构。

总价格在vuex中的getter进行计算。商品的名字与价格使用vuex中order的数据。

CartControl组件：

需要prop一个数据food，(shaopCart组件与orderdingFood组件都用到了cartControl组件)。如果是shopCart传过来的food有count，如果是orderdingFood传过来的food就是原始的该食品数据。

因为有n个CartControl组件，并且每个CartControl中的data数据私有。但是展示的数量需要共享，所以CartControl中的的添加、减少、数量都要操作vuex或者获取vuex中的数据。展示的数据为

遍历vuex中order的数据，如果该组件接收到的prop的name与某一个order中的数据相同，则将order数组中的那一条数据的count赋值到cartControl组件的数据上。

小知识点; 最外层的div有个点击事件，如果不想让内层div点击触发。给需要的内层div绑定事件且.stop即可

// 给shoplist添加点击事件，是为了阻止mask的点击事件

    stopThis() {

      this.a = 1

}

@click.stop=”stopThis”

处理bug

登陆之后仍然能点击。已解决。

cartControl组件点击一次报错，但是不影响正常的使用。

未解决

一个页面中定义了多个相同的组件，如果使用事件让其中一个组件中的computed属性或者watch属性触发了。这些组件都会被触发。

已解决： 问题的原因在于多了一个组件。在弹出商品详情的模态框里面多出来一个组件。这个组件并没有传值。打印出来所有的this发现的。

路由守卫

全局路由守卫：

拦截所有的路由。对其进行处理(拦截/强行跳转/放行)。

将 需要检查的数组定义为一个数组。

应用场景： 要想进入a路由和b路由，必须登陆。如果没有登陆，则强制跳转到登录页

/\*

功能1: 进入a/b必须登陆, 如果没有登陆自动跳转到登陆

\*/

const paths = ['/a', '/b'] // 需要进行登陆检查的path的数组

// 定义全局前置守卫

router.beforeEach((to, from, next) => {

  // console.log('全局前置', to, from)

  // 如果目标path在paths中, 但用户没有登陆, 自动跳转到login

  const path = to.path

  if (paths.indexOf(path)!==-1 && !store.state.user.token) {

    next('/login')

  } else {

    // 其它情况, 放行

    next()

  }

})

组件路由守卫，比如有这样一个需求：

1. 如果未登陆，点击div进行登录，如果登陆成功的话，点击此div要跳转到个人中心，就可以在此组件中使用路由守卫，当store中的token存在的话，就使用路由器跳转路由。

beforeRouteEnter (to, from, next) {

    next(component => {

      // 如果tokenu存在，则直接跳转。

      if(component.$store.state.token) {

        component.$router.replace({name: 'a'})

      }

    })

},

组件路由守卫与data，methods的层级一致。

场景：

组件中开了定时器，切换组件时，组件被销毁，但是定时器没有被销毁，可以在beforeDestory生命周期钩子函数中销毁该定时器。也可以在定时器循环过程中监听该钩子函数，销毁定时器，后者更符合vue的思想。

但是如果这个组件被keep-alive包起来的话，那么上述的解决办法便无效了。

新的应用场景，

A组件中有一个定时器，当不在此组件时，定时器停止(保留计时的数据)，当重新进入A组件时，定时器又从以前的位置重新开始计时。

解决方案：

使用keep-alive将组件包起来。定时器的开启在组件路由守卫，beforeRouterEnter内。定时器的结束在beforeRouterLeave内。

‘

使用vue-lazyload进行图片懒加载

1. 安装 yarn add vue-lazyload
2. main .js中使用插件。
   1. import VueLazyLoad from 'vue-lazyload'

Vue.use(VueLazyLoad,{

error:'./static/error.png',

loading:'./static/loading.png'

})

1. <img class="item-pic" v-lazy="newItem.picUrl"/>
   1. 将需要懒加载的图片的src属性改成v-lazy即可。

如果一个组件被keep-alive包裹起来，那么这个组件的beforeDestroy与Destroy两个生命周期钩子就不会被执行。但是后置守卫仍然会执行。Mounted钩子只会执行一次。第二次进去不会再次执行。

场景： 打开组件时开始创建定时器，销毁组件时清空定时器。重新打开组件时继续创建定时器……………

在mouted中开启定时器，beforeDestory中关闭定时器。可以实现。

如果这个组件是 keep-alive的，那么就mouted只在第一次进去的时候执行，而且组件的beforeDestroy与Destroy两个钩子无法执行。这个时候使用beforeRouterEnter和BeforeRouterLeave。

一些tab栏标签的切换通常用keep-alive包裹起来。

路由懒加载 : 什么时候使用这个路由什么时候才开始加载这个路由的js

const Msite = () => import(‘pages/Msite/Msite.vue’)

映射路由组件该怎么写就怎么写！

Import() : 动态导入模块，他会对被导入的模块进行单独打包(使用到了webpack中 code split)

定义的组件为一个返回的promise的函数，第一次请求路由对应路径时会请求后台获取对应的路由打包文件。 加载就是请求后台的路由js文件。

好处： 首屏得到了优化，第一次加载的js文件的量变得少了。

mode的 hash模式与history模式

如果是hash模式，真正的请求地址后面又#。

问题： 不美观

好处：仅hash符号之前的内容会被包含在请求中。 即使后端没有对路由的全覆盖，也不会返回404错误。

History模式：

好处： 地址栏没有#，而且符合浏览器的一些特性，能更好的应用这些特性。

缺点: 不怕前进不怕后退，就怕刷新，刷新是实实在在的去请求服务器的。

哈希模式：

Window中可以使用onhashchange事件，当url中的地址发生了变化，无需发送请求，window可以监听到这种变化，然后按需加载所需要的资源。

历史模式：

是正常的模式，但是如果要做到把url替换的同时就不刷新页面这里使用了pushState和replaceState两个H5的API。但是需要后端人员去配合，如果会断人员不配合的话，会造成访问二级页面，刷新操作报404的错误。

动态组件：

<component :is=”变量(对应组件名)”></component>

动态的改变变量的值，可以动态的切换组件。不过现在都用路由做了。

缓存组件

Kepp-alive将组件包裹起来。

Keep-alive能够解决这种场景。 比如一份列表数据，用户筛选出来之后，由数据列表进入该数据的详情，而当从数据的详情返回之后，该用户仍然能看到筛选后的状态。就使用keep-alive。并且可以避免组件的重复渲染，提高性能。

Keep-alive的实现原理

重写了钩子函数，创建唯一的缓存key，存入vue实例中。当渲染的时候，如果找到了这个key，则直接拿到这个key中的数据渲染到页面上。否则直接返回该组件的实例。

生产环境打包

yarn bulid

history的问题： 在某个路由下刷新会出现404的问题。跨域的问题。

Github issiue 解决方案

问题的解决方案