iphone6/7/8的DPR是2，表示iphone6/7/8屏幕的两个像素点描述了一个物理像素点。

而美工给的一般是iphone里面的像素值。所以使用的时候要除以2。

DPR和配合使用，如果DPR是2，那么meta里面设置成0.5，那么就可以直接使用美工给的px值了。

rem适配：<https://www.jianshu.com/p/ccbe9eb285e2> 没看 - -

淘宝rem适配github地址

<https://github.com/amfe/lib-flexible/blob/master/src/flexible.js>

使用方法， 在创建好的vue脚手架的终端里面直接npm install lib-flexible -S

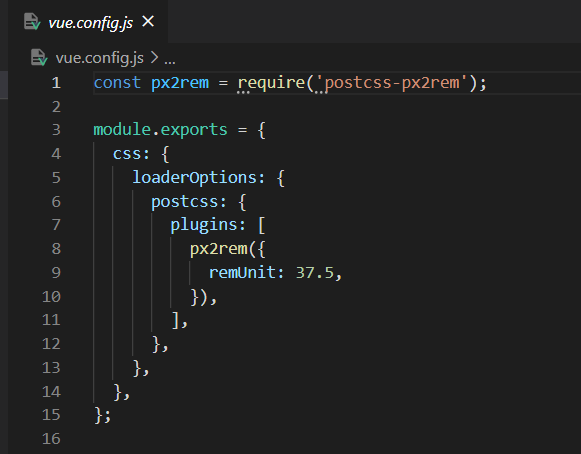
然后在main.js中import ‘lib-flexible’ 引入即可。这个插件会把1rem设置成视口宽度的十分之一。

然后把public文件夹下的index.html中的这个标签删除。

再利用postcss的一个插件把文档中的px自动转换成rem，npm install postcss-px2rem,

谐音(px to rem ) 表示把px转换成rem

然后配置vue.config.js文件，来指定postcss-px2rem这个插件的1rem表示多少像素。



vant组件库是根据37.5像素配置的，所以就需要把vue.config.js中的remUnit写成37.5

如何实现边框的0.5px，利用伪元素加缩放实现。

.name {

            width: 100px;

            height: 100px;

            background-color: red;

            position: relative;

        }

        .name::after {

            content: "";

            position: absolute;

            left: 0;

            top: 0;

            width: 200%;

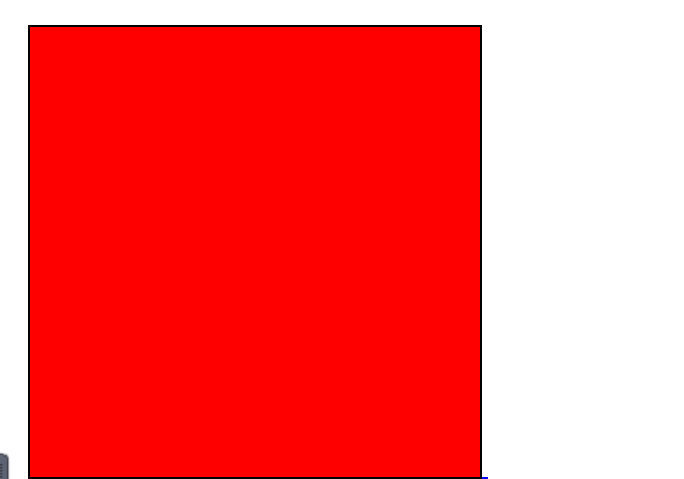
            height: 200%;

            border: 1px solid;

            transform-origin: 0 0;

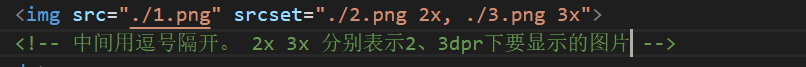
            transform: scale(0.5);

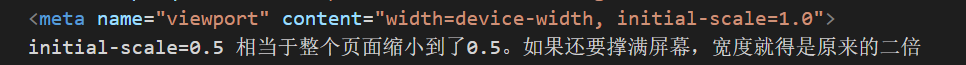
        }

其实是背面有个伪元素。

像素变大了，图片却模糊了。解决方法。

通过给img设置srcset属性来设置在不同dpr值下的不同图片。



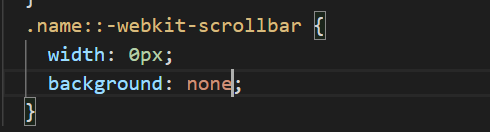


pc端最小字体是12px 移动端最小像素是8px

less使用：

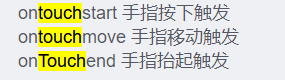


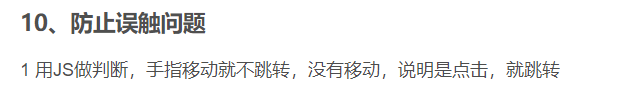
\*> 表示 #app 下面的所有元素。

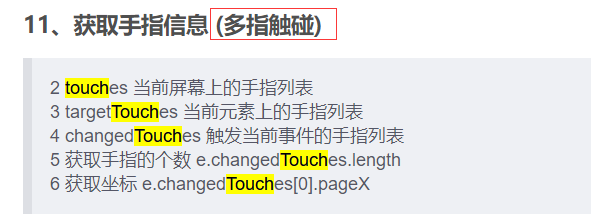


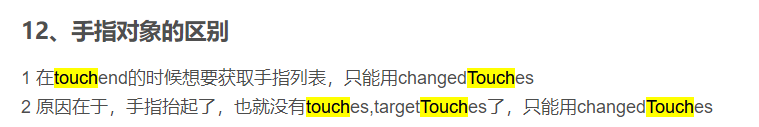
给某一个元素设置滚动条样式， 利用-webkit-scrollbal {} 来实现。

移动端事件：



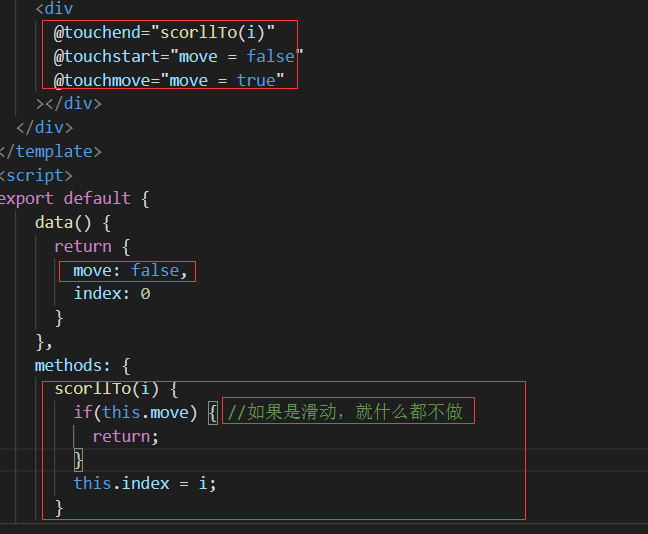


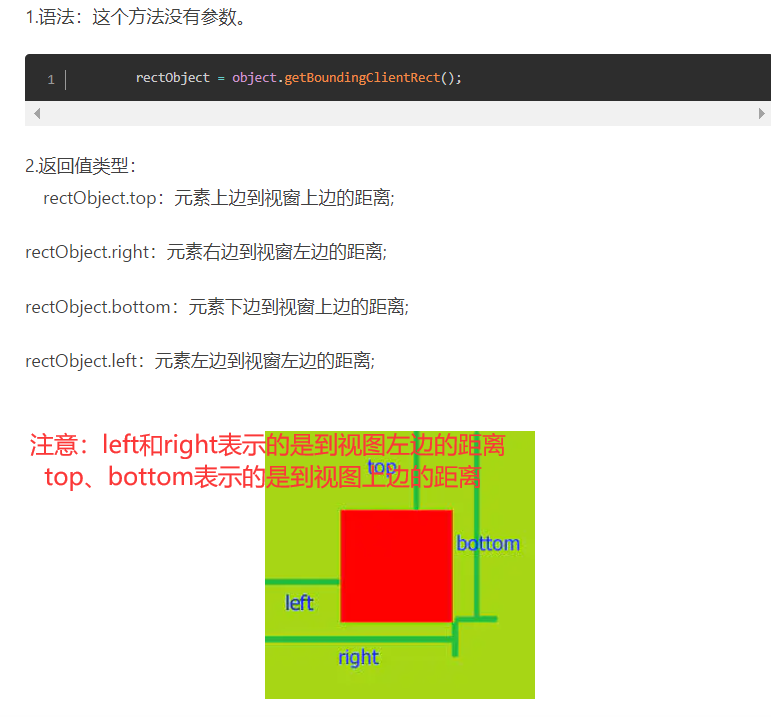


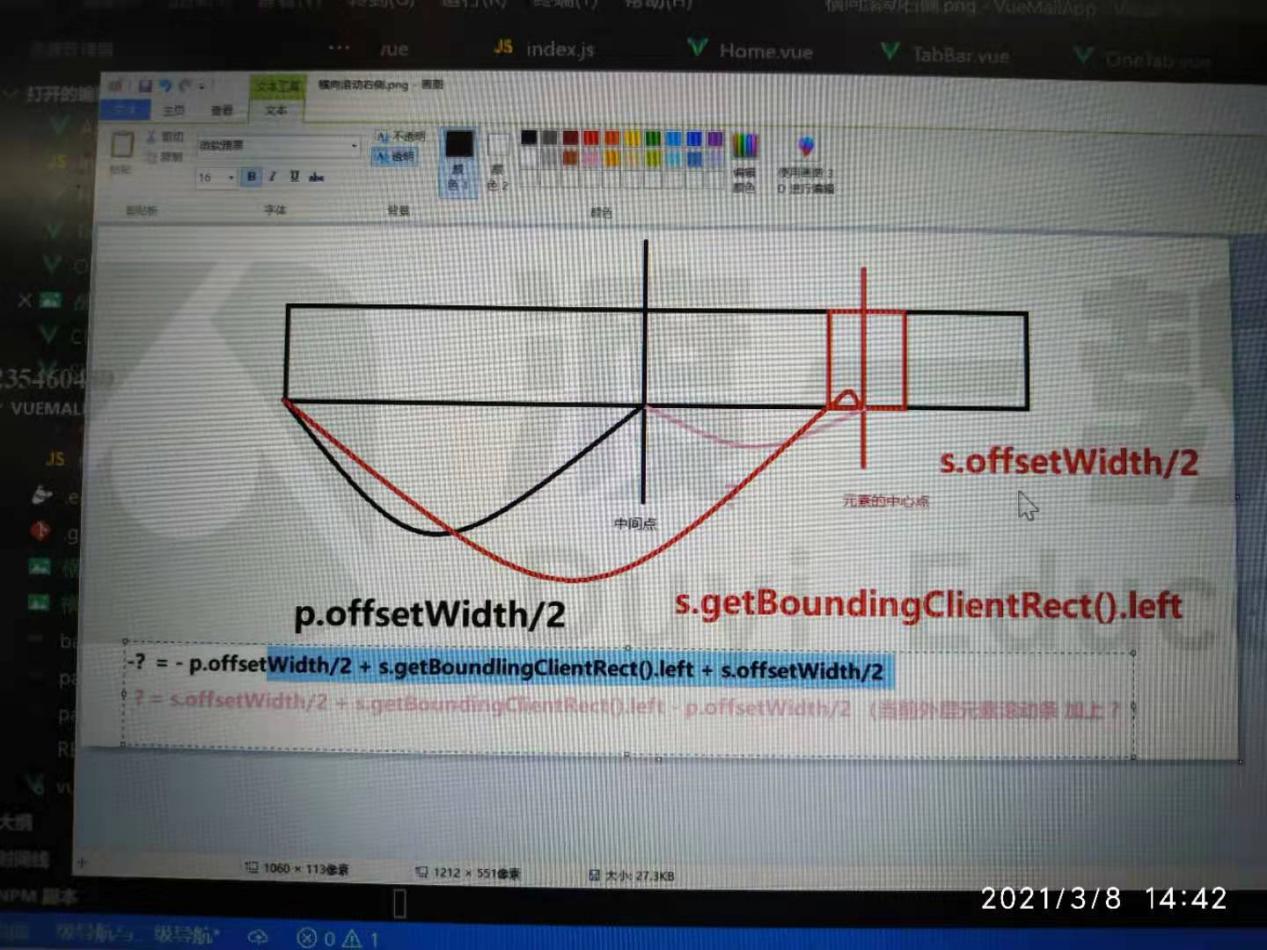




滑动的时候不触发触屏事件的解决办法：

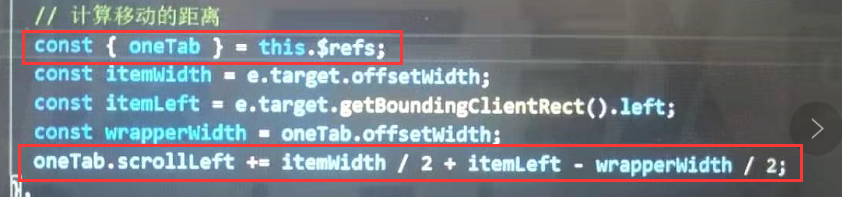






注意： 无论是从左边减还是从右边的加其实都是一个公式。

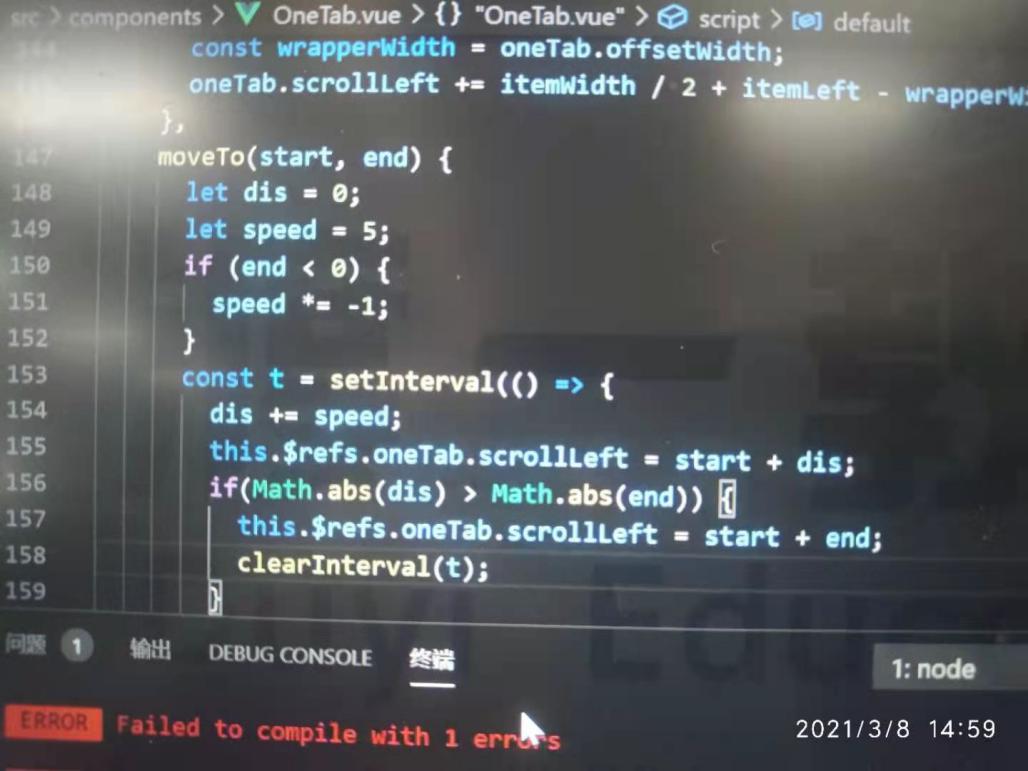
所以只用一个公式就可以解决两边的问题。注意下图：



methods中可以使用ref， 注意结构的写法，用一个公式解决两边移动的问题。

做一个过渡动画的函数：

参数start表示滚动条起始位置，end表示要滚动的距离



注意：

这里，要写成start+dis 的形式，每次定时器执行的时候叠加dis的形式。 而不要写成this.scrollLeft+dis的形式。

vue中事件函数传e， 在元素上绑定函数的里面传$event。

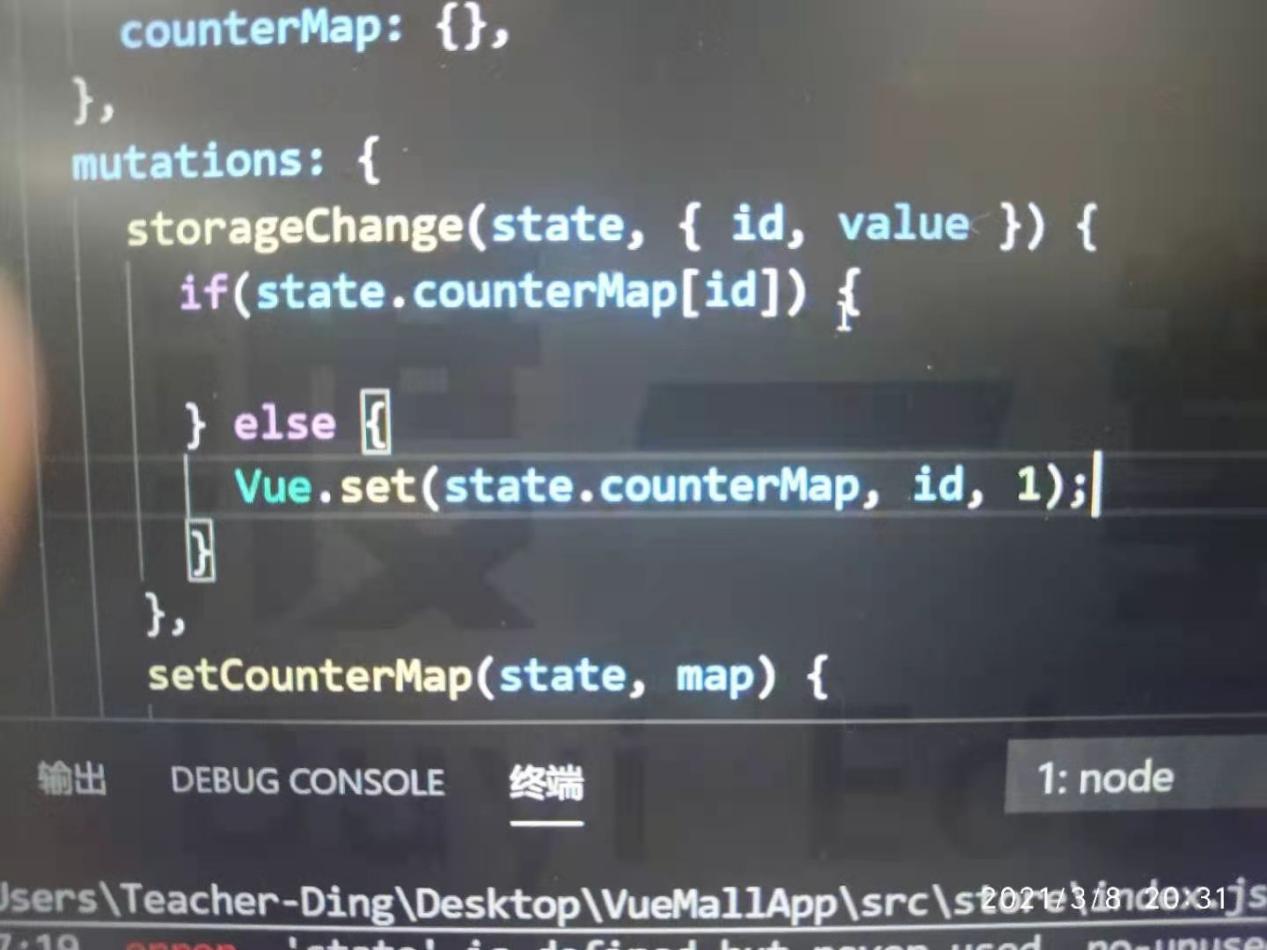
vue中，多个组件使用相同的方法，就提取到util中，创建一个tool.js的文件。放到里面使用。

js文件开头 注释的用法。



当vuex中的state中的counterMap还没有值时，需要先设置一个，所以需要使用

Vue.set()

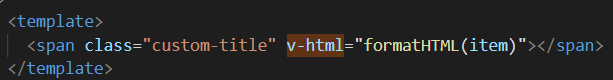




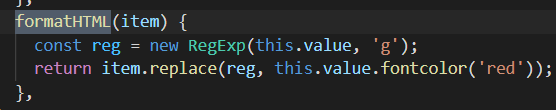
使用jquery获取元素的data-xx属性的属性值， 上面表示获取元素的data-fruit值。

如果要获取data-id 那么就是$(dom).data(‘id’);

v-html的使用方法：



item为上一个结构v-for的值。 此处当做是一个字符串即可。

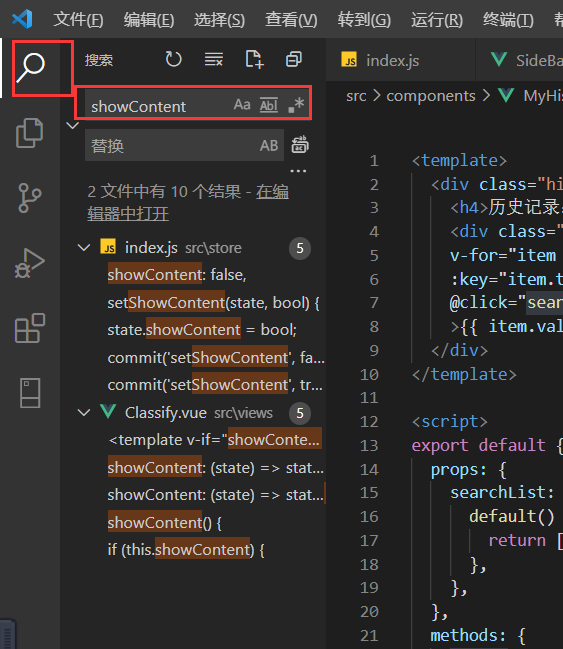


这是在methods中定义的v-html函数。

this.value表示搜索框的值，利用正则匹配出输入的值，然后传入的字符串匹配到的值替换成添加的颜色的值。 这里的fontcolor是js自带的方法，会把匹配到的字加上传入的颜色。

可以vscode打开的整个文件夹里面搜索和内容匹配的文件及位置。

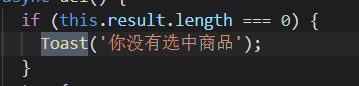
点击即可跳转。 找到变量按F12即可直接跳到对应的定义的位置。



vant局部引入组件。

如需要引入：toast组件时，可以直接在组件中：

用结构的方式。



使用这样的方式。

得到文字提示的效果：

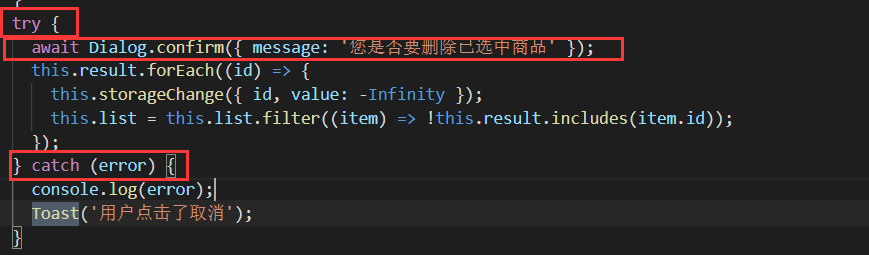


vant中的组件：

得到的是一个异步的弹窗，如果点击确认就到了then中，点击取消就到了catch中。

但是，也可以使用这样的方法来使用：如下：



如果点击确认，就直接删除localstorage，如果点击取消就谈消息’取消’

v-if在从true到false时，会触发destroy钩子函数。



一个组件中触发的事件，发送请求，得到数据，应用到另一个组件。

这种情况下，就可以把请求到的数据放到vuex中管理。

1. vuex中的state放的是请求回来的数据，或者某个组件的状态。

辅助函数...mapState({})

computed: {

    ...mapState({

      goodsList: (state) => state.goodsList,

      counterMap: (state) => state.counterMap,

    }),

  },

放到组件的computed中，当成组件的计算属性。

1. vuex中的mutations中放的是请求回数据以后通过数据改变state中的值的函数，或者改变某个组件的状态的函数。

...mapMutations({})

 methods: {

    ...mapMutations(['resetGoodsList']),

放到组件的methods中，当成组件的函数可以通过this访问到resetGoodsList

mutations的另一种使用方法 this.$store.commit('setCounterMap', counterMap);直接通过$stroe.commit访问mutations函数

3、通过actions里面定义的async函数进行发送请求，然后配合mutations中的函数改变状态。

 methods: {

      ...mapActions(['getGoodsList']),

放到组件的methods中，当成组件的函数可以通过this访问到getGoodsList

如果只需要执行actions中的函数的话，引入后执行this.getGoodsList(xxx)即可，

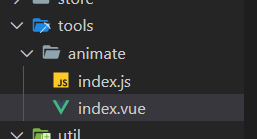
如果要获取actions中函数的返回值，就需要加个await了。

var result = await this.getGoodsList(xxx)；

vue中的extends的用法：

如果需要自己添加全局组件，new zujianming() 那么就需要用到extends了。

1. 先定义一个模板 index.vue，把要实现的功能都写在模板里面。并且没有静态数据，所用到的数据全部需要外部传递。





注意：上面的script标签里面不写data，看下面就知道了

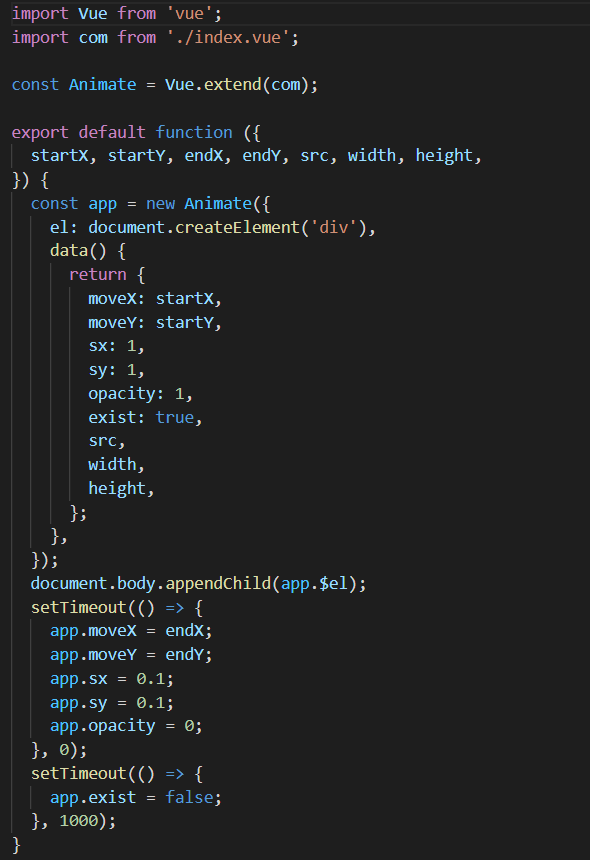
1. 在index.js中 通过extends把组件变成一个构造函数，可以new的构造函数，每次使用时，只需要new一个组件就可以了。如下：com表示上面的index.vue，导出一个函数供外部的vue使用，函数里面new Animtate({})其实就是对上面的script中代码的补充。

这样直接使用导出的函数，往里面传参就可以了。

其中：

el:document.createElement(‘div ’)表示，因为这个组件是创建一个元素，而不是挂载到已知元素上，所以需要创建元素。

document.body.appendchild(app.$el) 就是表示这个创建的元素，插入到body中。



3、使用extends，可以自己创建一个元素，把实现的效果的组件放到这个元素中，然后在组件中直接使用。如：

引入：tool/animate/index.js，也就是上面导出的函数，然后在组件中调用函数

