！Vue

移动端：

VUE YDUI 使用文档： <http://vue.ydui.org/docs/#/keyboard>

Vue的各种插件库： <http://vue.awesometiny.com/?tdsourcetag=s_pcqq_aiomsg>

有数据的接口（测试用的）

<https://jsonplaceholder.typicode.com/posts>

一．安装：

**Cli2.0**

npm install --global vue-cli （安装全局脚手架）

vue init webpack my-project （新建项目）

cd my-project （进入项目）

npm install （初始化项目）

npm run dev （运行项目）

安装使用http：

npm install vue-resource --save

在main.js引用import VueResource from‘vue-resource’,并：Vue.use(VueResiource)

安装路由

npm install vue-router --save

main.js引用 使用Vue.use(VueRouter)

**设置路由并将router放在new实例里面**

const router = new VueRouter({

mode:"history",

base:\_\_dirname,

routes:[

{path:"/",component:Customers},

{path:"/about",component:About}

]

})

**Cli3.0**

<https://www.jianshu.com/p/3fc59a4bba90>

**一．Js单独封装**

**Js文件：**

const Jsons={};

Jsons.a=””;

Json.b=function(){}

export default Jsons;

**Main.js文件**

import Jsons from 'src/util.js';

Vue.prototype.Json=Jsons;

**使用**

this.jsons.a;

this.jsons.b();

二．组件，路由传值

**子传父**：子：this.$emit("formSubmit",this.textToTranslate,this.langular);//传两个值

// textToTranslate和(langular)是子组件要传的值，formSubmit父组件中绑定的事件

父：<translateForm @formSubmit="translateText"></translateForm>

// translateText是执行的函数名，formSubmit与子组件对应

translateText（text，langular）{ //text和langular就是传过来的值 }

**父传子（默认传值没有点击事件直接传值）**：

父：<translateOutput :message ="hellow"></translateOutput>

// message父组件要传给子组件的值，定义在data里面的变量。写在组件中

子：props:[ "message" ], //message就是父组件传的值，直接用

**父组件调用子组件方法并传值2：**

父组件：<v-footer ref='footer'></v-footer>

<button @click='emitChild1'>ref与on触发</button>

方法：emitChild1(){

this.$refs.footer.childAction(val)//直接调用子组件方法

//footer是组建的ref，bridge是子组件监听的变化的值

// childAction是子组件的方法

},

子组件：

方法：childAction(val){

console.log(val)

}

**路由传给父组件：**

路由：this.$router.push({path:"/",query:{alert:"用户信息添加成功！"}});

//跳转路由，将信息赋值给alert去传递出去

父级this.$route.query.alert //获取传递过来的值alert；//注意这里是route不是router

**路由通过main.js的路由路径传递：**

main.js里面{path:"/customer/:id",component:CustomerDetails}

父级页面点击 <router-link v-bind:to="'/customer/'+customer.id"></router-link>

子页面：在初始化的时候获取created(){ this.$route.params.id }

this.$router.push({ name:"detail",params:{ code:11}})//params用name来传值，接收this.$route.params.code//不显示

this.$router.push({path:"/",query:{alert:"用户信息"}});// query用path来传值，接收this.$route.query.alert,显示浏览器上

三，建立本地jsonserver

1.npm install -g json-server

2.建立一个文件夹jsonserver

3.cd jsonserver

4.npm init (会有个提示，将文件名写成小写)，结果是，建成了一个package.json的文件

5.npm install json-server –save (建立项目node\_modules)

6.打开package.json文件，将script里面的key:value换成"json:server": "json-server --watch db.json"

7.在jsonserver文件夹下建一个db.json文件。

8.这样就可以在db里面写一些自己想要的json数据，然后启动json文件，

npm run json:server

1. 将jsonplaceholder里面的数据拿到本地来，在script里面添加

//"json:server:remot":"json-server <http://jsonplaceholder.typicode.com/db>"

然后保存并运行npm run json:server:remot,这样就能获取到很多远程下载到本地的数据

**三-二 建立本地服务器http-server**

<https://www.cnblogs.com/jingangel/p/7477930.html>

**三-三mock，前后端分离，模拟后端数据**

<https://www.cnblogs.com/jasonwang2y60/p/7302449.html>

<https://blog.csdn.net/bocongbo/article/details/81700843>

四．钩子/事件

created:function(){ } 和methods平级，初始化完成的时候加载

computed: {reversedMessage: function () {}}//数据计算/传递. reversedMessage是一个数据

watch:{'name':function(newValue,oldValue){}}//检测属性变化 属性变化的时候执行

**1.watch监听**

1). watch: {

cityName(newName, oldName) {

// ...

}

}

2). immediate表示在watch中首次绑定的时候，是否执行handler，值为true则表示在watch中声明的时候，就立即执行handler方法，值为false，则和一般使用watch一样，在数据发生变化的时候才执行handler.

deep: true 则可以监听到cityName.name的变化，此时会给cityName的所有属性都加上这个监听器，当对象属性较多时，每个属性值的变化都会执行handler

watch: {

cityName: {

　　handler(newName, oldName) {

　　// ...

　　},

deep: true,

　　immediate: true

}

}

3). 对单个属性进行监听

'cityName.name': {

handler(newName, oldName) {

// ...

},

deep: true,

immediate: true

}

五．基础。.class style绑定

1.class

<div class=”static” :class= “{ active : true/false, ‘text-danger’: hasError }” ></div>

= ==<div class=”static active text-danger” ></div>

2.class

<div :class="classObject"></div>

data: {

classObject: {

active: true,

'text-danger': false

}

}

3.class绑定 :class=”{displayNone:displayNoneAsync}”

// displayNone是css属性，displayNoneAsync是变化值

4. style

v-bind:style = “{color:activeColor,fontSize:fontSize+’px’ }”

4.事件处理器

@click

v-on:click.stop阻止冒泡

v-on:click.stop.prevent阻止默认事件

v-on:click.self 只给这个dome元素绑定事件，对其子元素点击不产生效果

v-on:click.once 对该事件只绑定一次

v-on:keyup.enter/tab/esc/left/right/down/delete…

六.路由

<router-link to='/about'>关于</router-link>

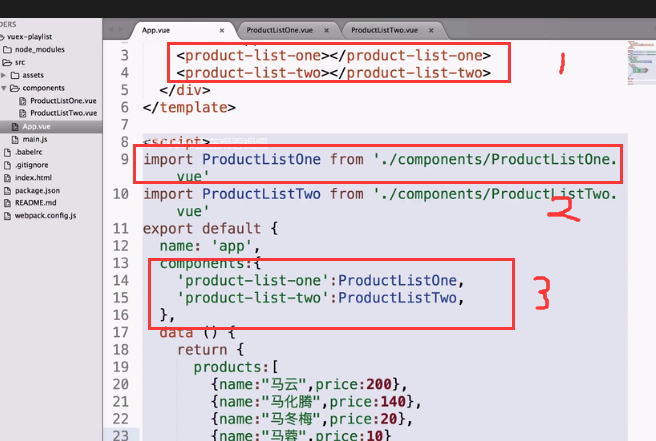
<router-view></router-view>

跳转到某个路由

this.$router.push({ name:"detail",params:{ code:11}})//params用name来传值，接收this.$route.params.code//不显示

this.$router.push({path:"/",query:{alert:"用户信息"}});// query用path来传值，接收this.$route.query.Alert,显示浏览器上

七．组件（组件内随便写）



八．Vuex（https://vuex.vuejs.org/zh-cn/）install vuex –save

1.vuex安装到使用

（1）.在src内创建store文件夹，在里面创建store.js文件



export default store

（2）.在main.js里面



///////////////////////////////////////////////////////////////////////////////////

（3）调用，使用

1.在组件内 this. $store.state.products  
 2.在公共js内调用：

// a.js

import store from './store.js' //先引入文件

console.log(store.state.count) //调用

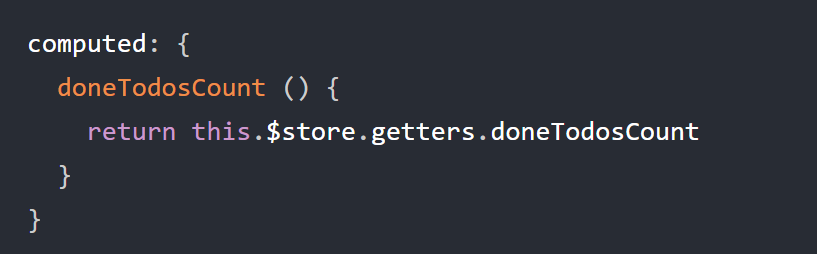
store.commit('increment') //调用

2.getters用法

定义：



调用：

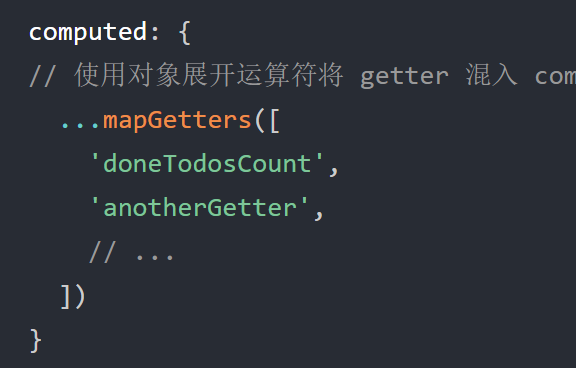


3. **mapGetters** ： 辅助函数，将store内的getter映射到局部计算属性（）

使用方法  
 1）.定义如上

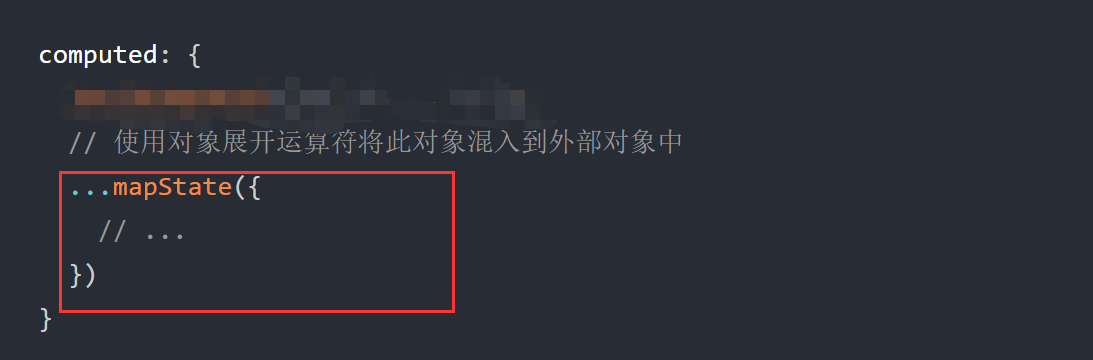
2）.组件内调用：  
 1.引入文件 import { mapGetters } from 'vuex'

2.配置：将getter内方法放进computed的mapGetters内



3.调用 this. doneTodosCount（相当于把doneTodosCount映射到vue的实例上）

**mapState**同上用法



# Mutation

更改 Vuex 的 store 中的状态的唯一方法……..必须是同步函数

展示写法：

mutations:{

addCount(state){

state.count++;

},

addCountWithParams(state, params){

state.count += params;

},

pushDataToArrat(state){

state.array.push({

name:"sunq",

age:100

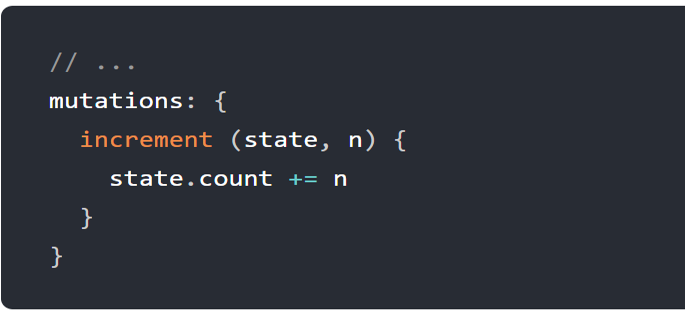
})

}

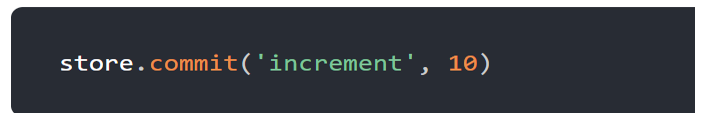
},

使用过程：

1. 定义方法：第二个参数是传递的值，可写可不写

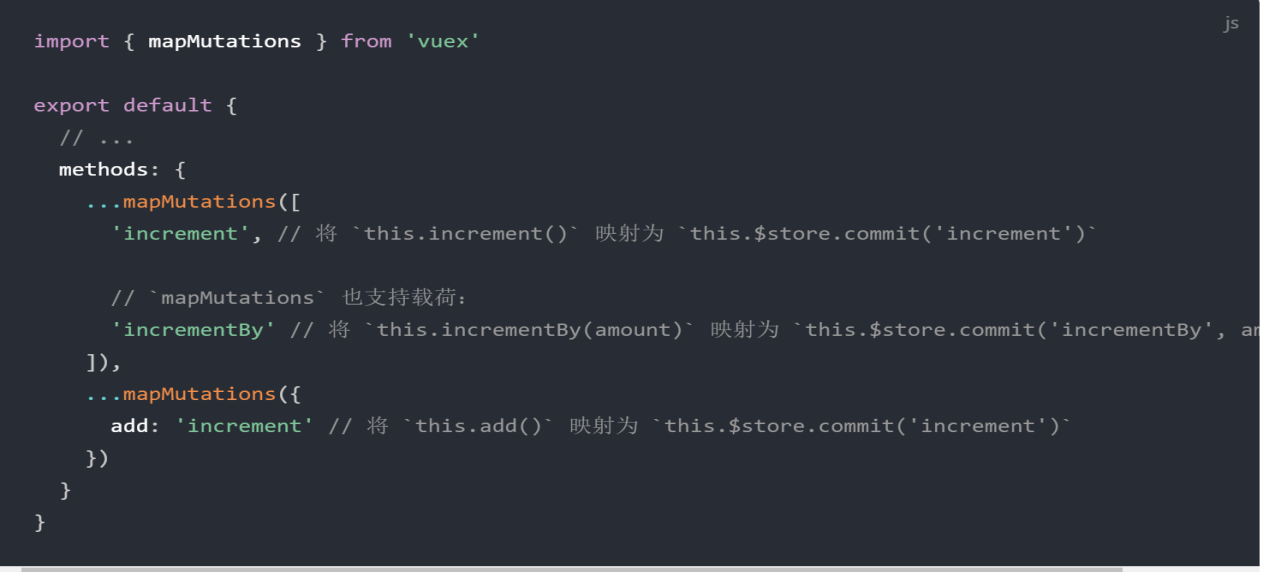


1. 调用

action内调用

组件内调用：this.$store.commit("increment");

**mapMutations** 用法：引入文件，配置在组建的methods内，直接调用



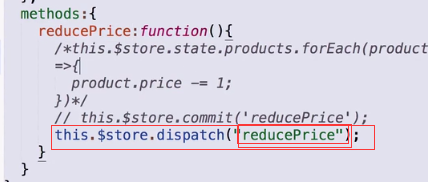
**Action**

Action 提交的是 mutation，而不是直接变更状态

Store内部定义和调用

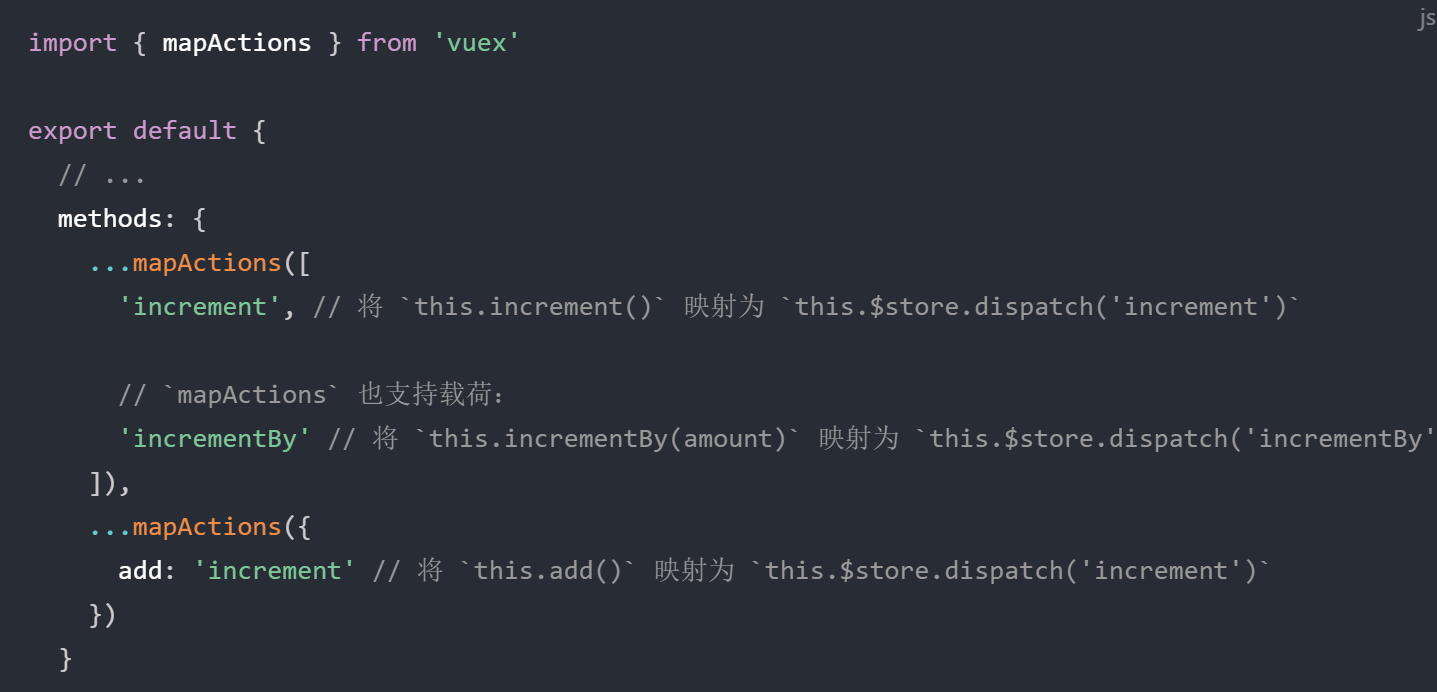


组件中的调用reducePrice是上面的action内的方法名



**mapActions**

使用方法：引用文件，配置在methods内，调用this.



1. **懒加载**

npm install vue-lazyload //安装

import VueLazyload from “vue-lazyload”

Vue.use(VueLazyload);





**十．生命钩子**

**beforeCreate**：在实例初始化之后，数据观测(data observer) 和 event/watcher 事件配置之前被调用。

**Created**：实例已经创建完成之后被调用。在这一步，实例已完成以下的配置：数据观测(data observer)，属性和方法的运算， watch/event 事件回调。然而，挂载阶段还没开始，$el 属性目前不可见。

**beforeMount** 在挂载开始之前被调用：相关的 render 函数首次被调用。

**Mounted** el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子。

**beforeUpdate** 数据更新时调用，发生在虚拟 DOM 重新渲染和打补丁之前。 你可以在这个钩子中进一步地更改状态，这不会触发附加的重渲染过程。

**Updated** 由于数据更改导致的虚拟 DOM 重新渲染和打补丁，在这之后会调用该钩子。该钩子在服务器端渲染期间不被调用。

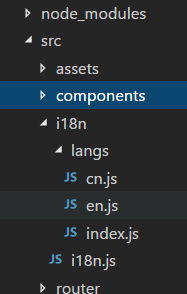
**beforeDestroy** 实例销毁之前调用。在这一步，实例仍然完全可用。

**Destroyed** Vue 实例销毁后调用。调用后，Vue 实例指示的所有东西都会解绑定，所有的事件监听器会被移除，所有的子实例也会被销毁。 该钩子在服务器端渲染期间不被调用。

**十一．国际化多语言i18n**

安装：npm install vue-i18n

目录：



各个文件的内容和用法

**main.js**  
import i18n from './i18n/i18n';

new Vue({

el: '#app',

i18n, //引用

components: { App },

template: '<App/>'

})

**i18n.js**

import Vue from 'vue'

import locale from 'element-ui/lib/locale';

import VueI18n from 'vue-i18n'

import messages from './langs'

Vue.use(VueI18n)

//从localStorage中拿到用户的语言选择，如果没有，那默认中文。

const i18n = new VueI18n({

locale: localStorage.getItem(“language”) || 'cn',

messages,

})

locale.i18n((key, value) => i18n.t(key, value)) //为了实现element插件的多语言切换

export default i18n;

**index.js**

import en from './en';

import cn from './cn';

export default {

en: en,

cn: cn

}

**Cn.js**

import zhLocale from 'element-ui/lib/locale/lang/zh-CN'

const cn = {

message: {

'hello': '你好，世界',

},

...zhLocale

}

export default cn;

**en.js**

import enLocale from 'element-ui/lib/locale/lang/en'

const en = {

message: {

'hello': 'hello, world',

},

...enLocale

}

export default en;

使用点击事件：

changeEng(){

this.$i18n.locale = 'en';

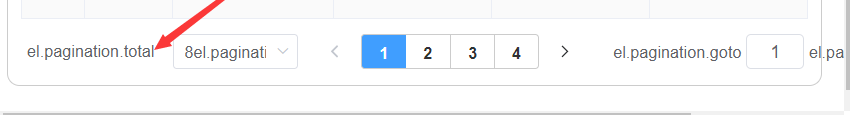
localStorage.setItem(“language”,”en”)

}

实例：

<p>{{$t('message.hello')}}</p>

**出现的问题bug**：表格的页码的地方出现错误，



需要在最开始初始化一下

（放在index.vue初始化一次即可）

this.$i18n.locale = localStorage.getItem("langular")||"cn";

**十二．Vue的滚动条easyScroll**

// <https://blog.csdn.net/qq_35844177/article/details/78042284>

1>.安装

1. npm install babel-plugin-syntax-jsx babel-plugin-transform-vue-jsx babel-helper-vue-jsx-merge-props babel-preset-es2015 --save-dev

2. npm install --save easyscroll

2>.引入main.js

import EasyScroll from "easyscroll";

Vue.use(EasyScroll);

3>.使用

在父级div上面加id="wrapper"

<EasyScrollbar>

<div id="wrapper" style="height: 300px;">

<div style="height: 500px;background-color: greenyellow;text-align: center;">

最简单的应用

</div>

</div>

</EasyScrollbar>

**十三. Vue内动态生二维码**

<https://www.cnblogs.com/zhengweijie/p/6911834.html>

1、引入qrcode--------npm install qrcode

2、在main.js中引入

import QRCode from 'qrcode' //定义生成二维码组件

3、在需要使用到生成二维码的组件中引入

import QRCode from 'qrcode' //引入生成二维码组件

4、在HTML中定义生成的位置，注意添加样式

<template>

<div id="query">

<h1>二维码：</h1>

<canvas id="canvas"></canvas>

</div>

</template>

#canvas{

width: 80%!important;

height: auto!important;

}

5、在js中定义生成二维码的方法并调用

//动态生成二维码

useqrcode(){   
 //生成的二维码内容，可以添加变量  
　　　　this.QueryDetail=”需要生成二维码的内容”；

var canvas = document.getElementById('canvas')

QRCode.toCanvas(canvas, this.QueryDetail, function (error) {

if (error) console.error(error)

console.log('success!');

})

}

**十四. Vue不能在ie下运行问题（不同问题不同处理）**

1. 安装 npm install --save babel-polyfill
2. 在package.json中添加依赖，

devDependencies：{

…

"babel-polyfill": "^6.22.0"

}

1. 在app.vue中添加引用“import 'babel-polyfill'”
2. 在webpack.config.js文件中使用

module.exports = {  
     entry: {  
     app: ["babel-polyfill", "./src/main.js"]  
     }  
};

**十五. 人工智能验证**

[**https://help.aliyun.com/document\_detail/66323.html?spm=5176.2020520162.0.0.51f25fb0uqKc5K**](https://help.aliyun.com/document_detail/66323.html?spm=5176.2020520162.0.0.51f25fb0uqKc5K)

**十六.vue点击复制文本**

vue点击复制文本

<https://my.oschina.net/benlv/blog/1613147>

**import Vue**

**import VueClipboard from 'vue-clipboard2' Vue.use(VueClipboard)**

**<script src="vue.min.js"></script>**

**单独引用**

**<!-- 必须在vue.js之后放置这一行 -->**

**<script src="dist/vue-clipboard.min.js"></script>**

<template id="demo">

<div class="container">

<input type="text" v-model="message">

<button type="button"

v-clipboard:copy="message"

v-clipboard:success="onCopy"

v-clipboard:error="onError">复制</button>

</div>

</template>

onCopy: function (e) {

console.log('你刚刚复制: ' + e.text)

},

onError: function (e) {

console.log('无法复制文本！')

}

**十七. Sourcetree**

//不登录 <https://www.cnblogs.com/lucio110/p/8192792.html>

<https://blog.csdn.net/qq_34975710/article/details/74469068>

eg1.有分支：自己的分支提交推送，(提pull request)，master分支拉取，切换到自己的分支，右击master向本分支合并，解决冲突，再提交。

eg2.如果直接用主分支，没有自己的分支，先提交，拉去，（解决冲突），推送

**十八.vux**

<https://vuex.vuejs.org/zh-cn/>

**十九.vue引入cookie**

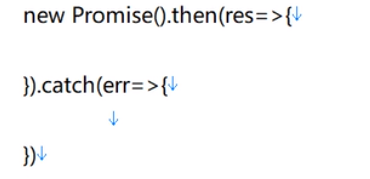
<https://blog.csdn.net/M_wolf/article/details/78967233>

**二十promise**

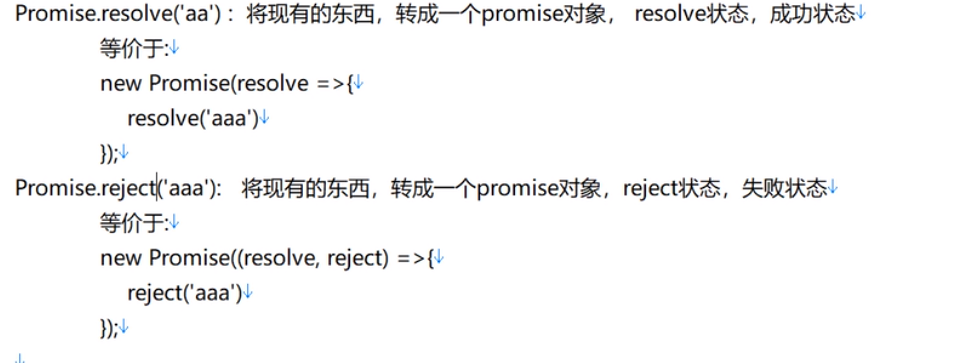
**1.。。。。**



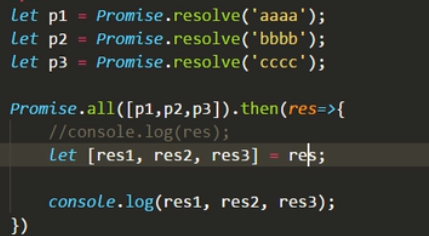
**2.相当于**



**3.Promise.resolve(“”); Promise.reject(“”);**



1. **Promise.all([p1,p2,p3]).then(res=>{});**

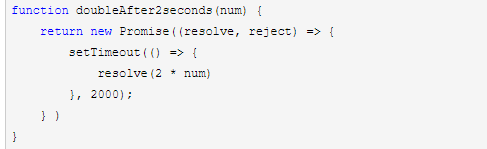


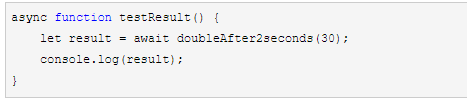
**Eg:例如**





**Aysnc**  //await是等待执行完成之后才能继续往下执行。





**二十一.** 饥人谷 浏览器直接编辑代码/一些插件

<http://js.jirengu.com/pusufipoxu/1/edit?html,console,output>

axios

<https://www.kancloud.cn/yunye/axios/234845>

<https://www.cnblogs.com/libin-1/p/6607945.html>

加载图片

<http://xyxiao.cn/vue-cropper/example/>

移动端滚动插件BetterScroll

<https://juejin.im/post/59b777015188257e764c716f>

兼容各个浏览器的滚动轴样式：  
<https://garveyzuo.github.io/easyscroll.github.io/#/>

Decimal 精度计算

<https://github.com/MikeMcl/decimal.js>

vue 的移动端的框架mintui

<http://mint-ui.github.io/docs/#/zh-cn2/quickstart>

**二十二 vue 打印**

<https://www.jianshu.com/p/c5363ddf7d43>

**二十三 Vue-echarts**

<https://v-charts.js.org/#/>

( npm i v-charts echarts -S )

安装有问题的情况下用淘宝镜像安装

**二十四 less**

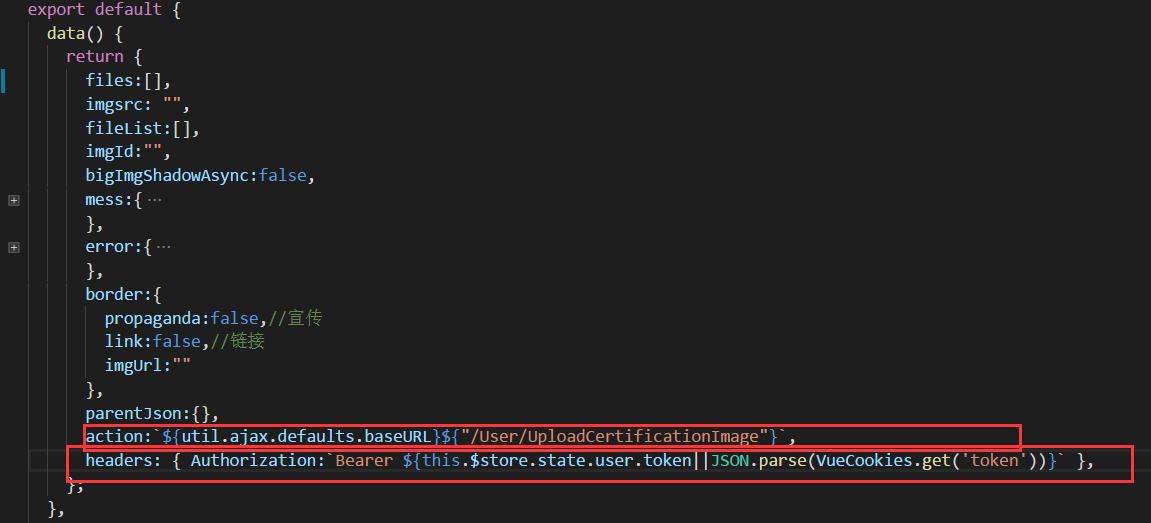
npm install less less-loader --save //安装less和less依赖

写法：

<style scoped lang="less">

**写法问题**

1.data里面用store和公共js里面的内容



2.html里面属性绑定

<img :src="`../../../static/img/country/dist/az-country-`+`${item.shortName}.png`" /><div class="picBoxTopBg" v-if="packetDetail.imgUrlAsync"

:style="{backgroundImage:'url('+packetDetail.imgUrl+')'}">