前端利用Vue.js开发项目小结

——以互联网小贷风控系统为例

李鸿文2017/8/1

**培训内容：**

1.开发环境搭建

2.关于webpack模板的目录结构说明

3.npm 常用命令

4.webpack结合git用法

5.Vue.js介绍和基本用法

6.Vue-router路由用法

7.axios基本用法及解决ajax跨域请求配置

8.Element ui基本用法介绍

9.分享互联网小贷风控系统开发心得及遇到的坑

**开场白：聊聊我的前端经历**

*关键词：12年毕业，近五年前端工作经历，14年之前，以jquery为主，15年angular开发，SPA单页面，16年react火了，这正是 vue 跟 react PK，阿里weex 挑战 rn的时候，腾讯又跑出了个小程序~~~*

【前2两年都是*jquery*为主，切图/静态页面 == 俗称切图页面仔，

模仿建站，套模板php建站 == web前端；

后两年，基于java接口，spa单页面，前后端分离，使用AngularJS，vue，==前端开发】

三剑客没落了：

1、网页三剑客（Dreamweaver，Fireworks，Flash）

2、前端开发框架三剑客—AngularJS VS. Backone.js VS.Ember.js

三足鼎立之势：angular4,react,vue，

就全球使用率来说：angular；

就生态社区最活跃：react；

最轻量级，国产的个人开源，最容易入门：vue

前端的发展：

原始的刀耕火种 》》 前端构建工程化/自动化；

个人的单打独斗jq时代 》》 前端团队开发（基于MVVM模式前后端分离）

一句话：*前端入门简单难精通，搞懂原理不简单~*

前端已不止于前端: <http://mp.weixin.qq.com/s/V6wFBoFg8dq6kIorCWDj8g>

套用一句在前端届流传很广的“名言”来开启我们的篇章：

*任何能用javascript实现的软件及系统，最终都会拿来用javascript实现。 －－ 马云*

**1.开发环境搭建**

1.1、环境准备

1.1.1安装软件：

1)从node.js官网下载并安装node 输入 node -v 查看版本

2)npm包管理器，是集成在node中的，所以，直接输入 npm -v 查看版本

1.1.2安装构建工具和脚手架：

1) npm install webpack -g //安装webpack 输入 webpack -v 查看版本

2) npm install -g vue-cli //安装vue脚手架 输入 vue -V 查看版本

注意，此时V是大写

1.2构建项目:

安装完成后，即可运行命令：

$ vue init <template-name> <project-name>

<template-name>：模板名称，运行npm list会显示有什么模板可以选择；

<project-name>：需要命名的项目名称；

例如运行：

1.vue init webpack my-project //在工作目录下生成项目my-project

2.npm install //在项目目录初始化npm

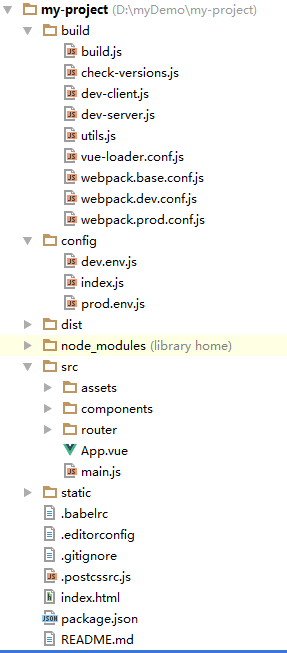
1.3 启动和打包:

1. npm run dev //启动开发

2. npm run build //打包发布

**2. 关于webpack模板的目录结构说明**

2.1生成的目录结构：



2.2目录结构说明：

|--build  //webpack相关代码文件夹  
|  |--build.js //生产环境结构代码  
|  |--check-version.js //检查node、npm、等版本  
|  |--dev-client.js  //热加载相关代码  
|  |--dev-server.js  //本地服务器  
|  |--utils.js  //构建工具  
|  |--webpack.base.conf.js //webpack基本配置  
|  |--webpack.dev.conf.js //webpack开发环境配置  
|  |--webpack.prod.conf.js  //webpack生产环境配置  
|--config  //项目开发环境配置  
|  |--dev.env.js  //开发环境变量  
|  |--index.js  //项目基本配置（proxyTable:{ //配置请求代理}）   
|  |--dev.env.js  //开发环境变量  
|  |--prod.env.js  //生产环境变量  
|--dist  //执行npm run build，生成打包发布的目录   
|--node\_modules  //初始化 npm install，生成的依赖包目录（注意，不要提交到svn！）  
|--src  //项目源代码目录  
|  |--components  //组件目录  
|  |--assets  //Vue默认logo目录  
|  |--router  //路由目录  
|  |--APP.vue  //默认组件，入口文件  
|  |--main.js  //程序入口文件，引用、加载各种组件  
|--static  //静态文件目录，比如：CSS、图片、等等静态文件  
|--index.html  //入口文件

说明参考：<https://my.oschina.net/hgwn/blog/1486220>

vue-cli的webpack模板项目配置文件分析

<http://blog.csdn.net/hongchh/article/details/55113751>

**3. npm 常用命令**

3.1、npm包安装模式

》》本地安装：package会被下载到当前所在目录，也只能在当前目录下使用。

》》全局安装：package会被下载到到特定的系统目录下，安装的package能够在所有目录下使用。

3.2、安装模块

1）本地安装，如grunt-cli

npm install grunt-cli：安装包grunt-cli，默认会安装最新的版本

npm install grunt-cli@"0.1.9"：安装0.1.9版本的grunt-cli

npm install grunt-cli --save 或 npm install grunt-cli -S：安装包grunt-cli并将信息将加入到package.json文件的dependencies（生产阶段的依赖）

npm install grunt-cli --save-dev 或 npm install grunt-cli -D:安装包grunt-cli并将包信息写入package.json文件的devDependencies（开发阶段的依赖）配置中，这样代码提交到github时，就不用提交node\_modules这个文件.

npm install grunt-cli --save-optional 或 npm install grunt-cli -O:安装包grunt-cli并将信息将加入到optionalDependencies（可选阶段的依赖）

模块的依赖都被写入了package.json文件后，他人打开项目的根目录（项目开源、内部团队合作），使用npm install命令可以根据dependencies配置安装所有的依赖包

2）全局安装，如npm

npm install -g npm :全局安装npm

npm install -g npm@2.14.14 ：安装指定的npm版本，同时也是降低和更新npm版本的方法；

全局安装模块的目录一般为：

c:\User\Administrator\AppData\Roaming\npm\node\_modules\下；

全局安装的模块，在代码中直接通过require()的方式是没有办法调用到的。全局的安装是供命令行使用的，就好像全局安装了vmarket后，就可以在命令行中直接运行vm命令。也就是可以直接在cmd的命令行中使用。

3.3、卸载模块

比如grunt-cli

npm uninstall grunt-cli: 卸载包grunt-cli

npm uninstall grunt-cli@"0.1.9": 卸载0.1.9版本的grunt-cli

3.4、更新模块

比如grunt-cli,全局npm

npm update #升级当前目录下的项目的所有模块

npm update grunt-cli 更新

npm update -g npm@3.14.14，安装指定的npm版本，同时也是更新npm版本的方法；

3.5、查看模块

比如 grunt-cli

npm ls grunt-cli：查看特定包grunt-cli的信息

npm info grunt-cli：查看详细的输出信息（包括作者、版本、依赖等）。

npm list #列出已安装模块

npm show grunt-cli #显示模块详情

3.6实例：

引入element ui <http://element.eleme.io/1.3/#/zh-CN/component/installation>

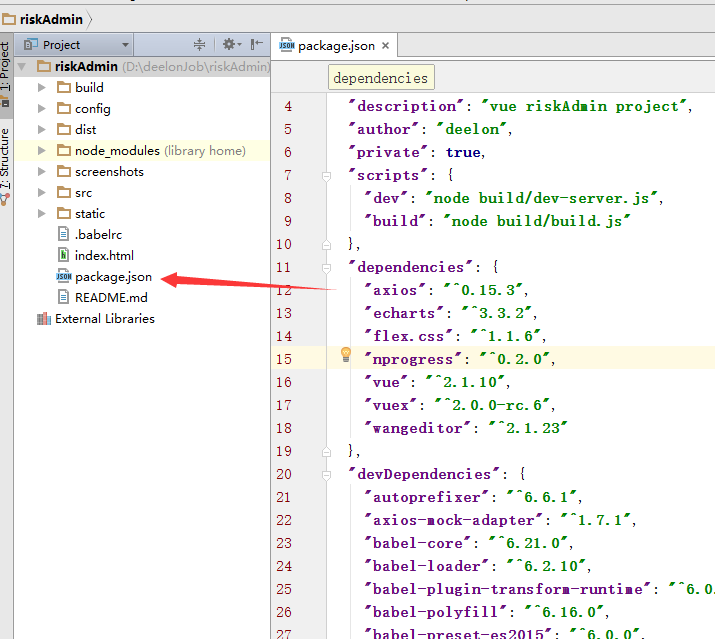
npm安装，命令如下：

npm i element-ui -S //简写

同样，引入axios，——http ajax 请求组件，命令如下：

npm install axios –save

安装包并将信息保持到项目的package.json文件中，见下图：



【延伸】：

-S, --save 安装包信息将加入到dependencies（生产阶段的依赖），如：

npm install gulp --save 或 npm install gulp -S

package.json 文件的 dependencies 字段：

"dependencies": {

"gulp": "^3.9.1"

}

-D, --save-dev 安装包信息将加入到devDependencies（开发阶段的依赖），所以开发阶段一般使用它，如：

npm install gulp --save-dev 或 npm install gulp -D

package.json 文件的 devDependencies字段：

"devDependencies": {

"gulp": "^3.9.1"

}

npm详解：<http://www.cnblogs.com/wdlhao/p/5900360.html>

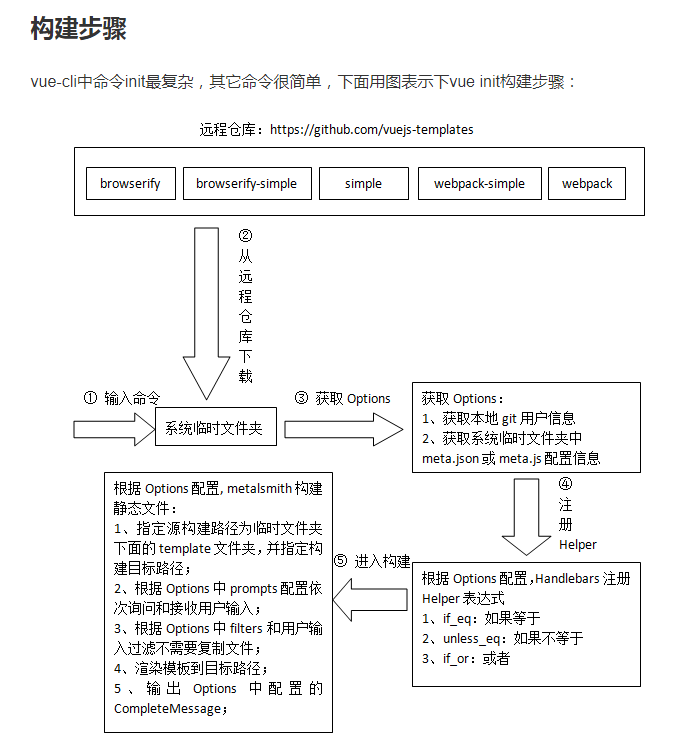
**4. webpack 结合 git用法**

Why?

*一位JAVA大神杰哥说过：原理是解决问题的法宝！！！*

*一位全栈开发大神涛哥说过：光是看文档是没用的，关键是实操，用到项目去！！！！*

基于webpack+vue-cli构建项目的原理图，不难看出：



git教程：

<https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/>

**5. Vue.js介绍和基本用法**

学习一个新技术，必须要清楚两个W，”What && Why”。

“XX 是什么？”，

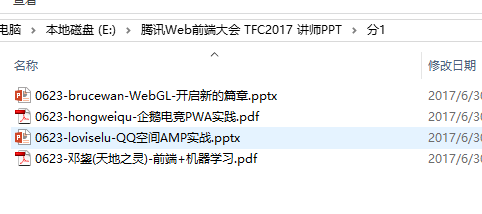
”为什么要使用 XX ，或者说 XX 有什么好处”，

最后才是”XX 怎么使用”。

前端技术发展很快，来势汹汹~已经跟不上了~

框架 ：jquery > backbone / knockout> angular/vue/react 。。。PWA? （未来）

下图贴今年腾讯举办2017前端大会，前端新技术：

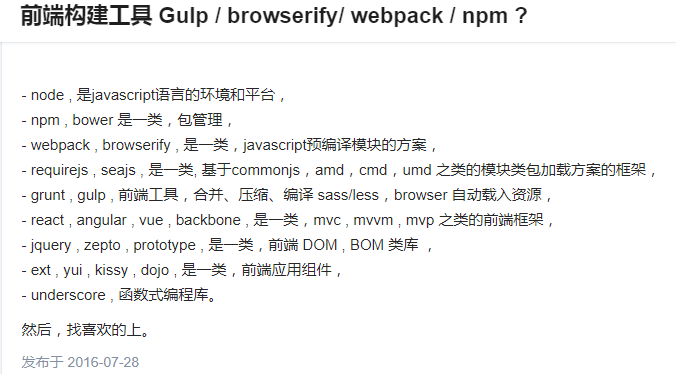


前端打包构建工具：grup < gulp /百度FIS < webpack < rollup

问题：前端构建工具 Gulp / browserify/ webpack / npm ? 啥区别呢？

截图知乎整理的分类，仅供参考：

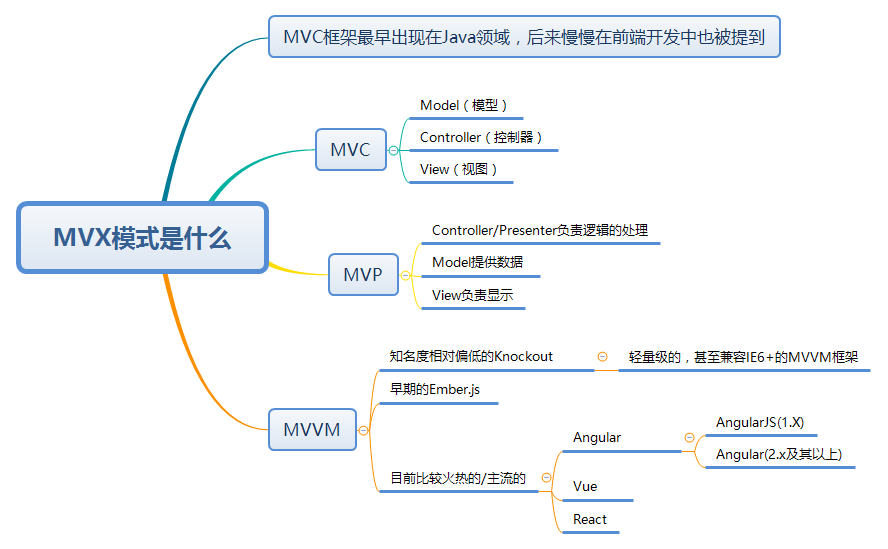
<https://www.zhihu.com/question/37694275>



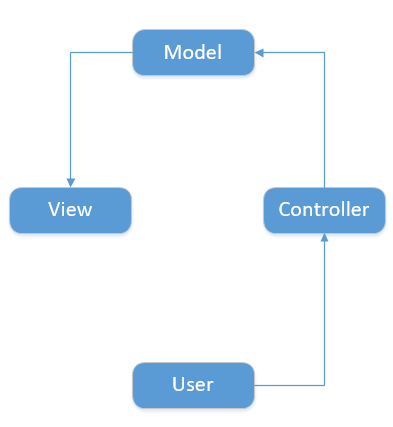
*上图：知乎网友的整理，仅供参考。*

5.1 MVX模式是什么?

下图是我参考《VUE.JS权威指南》一书的理解，画的思维导图：

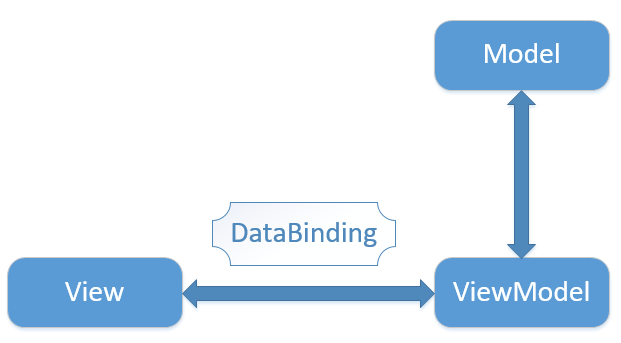


》MVC模式通讯方式：用户（User）通过Controller来操作Model已达到View的变化。

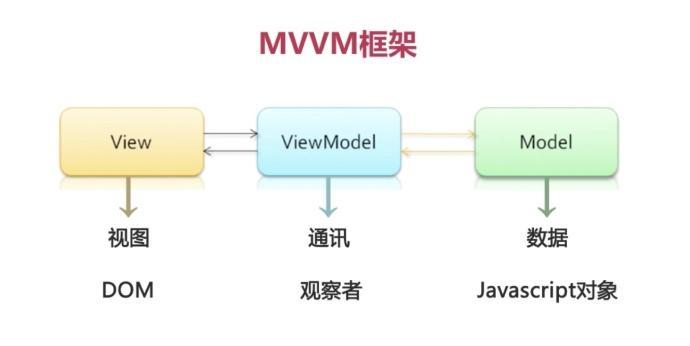


》MVP模式：MVP是从经典的MVC模式演变而来，它们的基本思路有想通的地方：Controller/Presenter负责逻辑的处理，Model提供数据，View负责显示。

》MVVM模式：相比前面两种模式，MVVM只是把MVC的Controller和MVP的Presenter改成了ViewModel。View的变化会自动更新到ViewModel，ViewModel的变化也会自动同步到View上显示。



MVVM 全称是 Model - ViewModel - View 的简称



Model 对应的是数据，JavaScript 对象。

View 对应的是视图，也就是 DOM。

ViewModel 是连接 View 和 Model 的中间件，在 MVVM 下，View 和 Model 是不能直接通讯的，当数据变化（用户操作视图），ViewModel 都能监听到变化，实现了双向绑定。

附录：

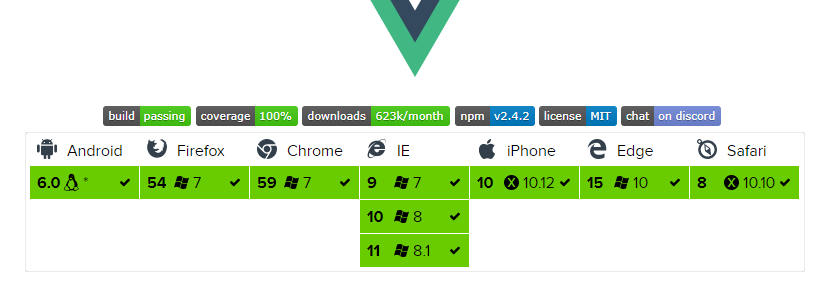
前端 MVC 已死？【译】：

<https://blog.yongyuan.us/articles/2016-10-15-mvc-dead/>

5.2 Vue.js是什么

官网：<https://cn.vuejs.org/v2/guide/>

Vue的兼容性，见下图：



5.3. vue / angular /react区别

对比主流框架：<https://cn.vuejs.org/v2/guide/comparison.html>

Vue 跟react 使用Virtural DOM，组件化的视图组件，将注意力集中保持在核心库，而将路由和全局状态管理分离出来（交给相关的库）

Vue和AngularJS（1.x版本），语法很相似（如 v-if 和 ng-if），借鉴了AngularJS

5.4 基本用法

一、基础语法

1、v-model(绑定数据)

2、v-for（循环）

3、v-on（绑定事件）

4、data（数据）

5、methods（方法）

6、v-if（不满足条件的话则不会出现在dom中）

7、v-show（不满足条件，则样式会设置成隐藏 display:none;）

二、数据绑定语

1、文本插值：

{{msg}}

2、单次绑定：

v-once {{msg}}

3、纯HTML：

v-html

4、绑定表达式：

{{number+1}}

5、过滤器：

{{ message | filterA | filterB }}、{{ message | filterA ‘arg1‘ arg2 }}

6、指令缩写：

<a v-bind:href="url"></a> -->>

<a :href="url"></a>

<button v-bind:disabled="XXX">Button</button> -->>

<button :disabled=" XXX ">Button</button>

<a v-on:click="doSomething"></a> -->>

<a @click="doSomething"></a>

5.5 理解生命周期图

[http://cn.vuejs.org/v2/guide/instance.html#](http://cn.vuejs.org/v2/guide/instance.html)生命周期图示

理解，参考《Vue.js权威指南》一书，

》》 init：在实例开始初始化时同步调用。此时数据观察、事件和Watcher都尚未初始化。

》》 created：实例创建之后同步调用。此时已建立：数据绑定、计算属性、方法、Watcher/事件回调。但是还没有开始DOM编译，$el还不存在。

》》 beforeCompile：在编译开始前调用。

》》 compiled：在编译结束后调用。此时所有的指令已经生效，因而数据的变化将触发DOM更新。但是不担保$el已插入文档。

》》 ready：在编译结束和$el第一次插入文档之后调用，如在第一次attached钩子之后调用。注意，必须是由Vue插入（如vm.$appendTo()等方法或指令更新）才触发ready钩子的。

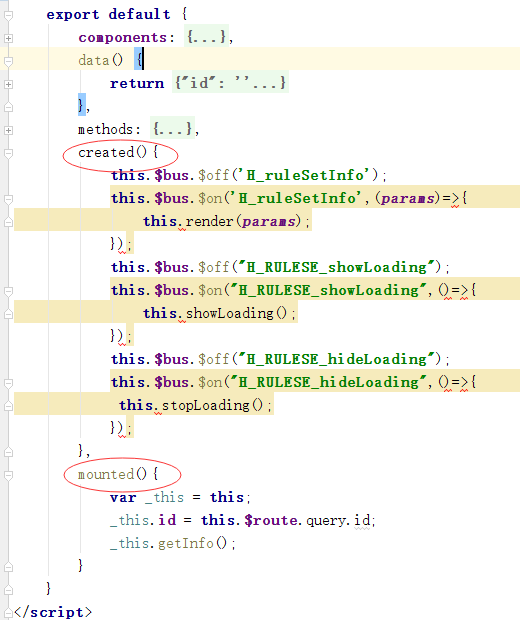
》》 attached：vm.$el插入调用。必须是由指令或实例方法（如$appendTo()）插入，直接操作vm.$el不会触发这个钩子。

》》 detached：在vm.$el从DOM中删除时调用。

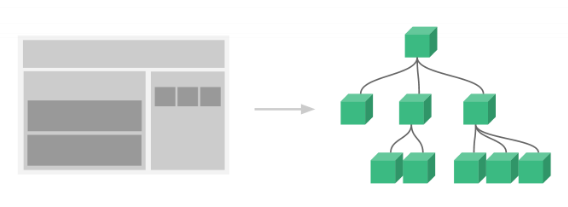
》》 beforeDestroy：在开始销毁实例时调用。

》》 destroyed：在实例被销毁之后调用。

目前，在风控系统项目开发中，只用上了created()和mounted()，实例代码：



5.6如何理解组件？模块化？



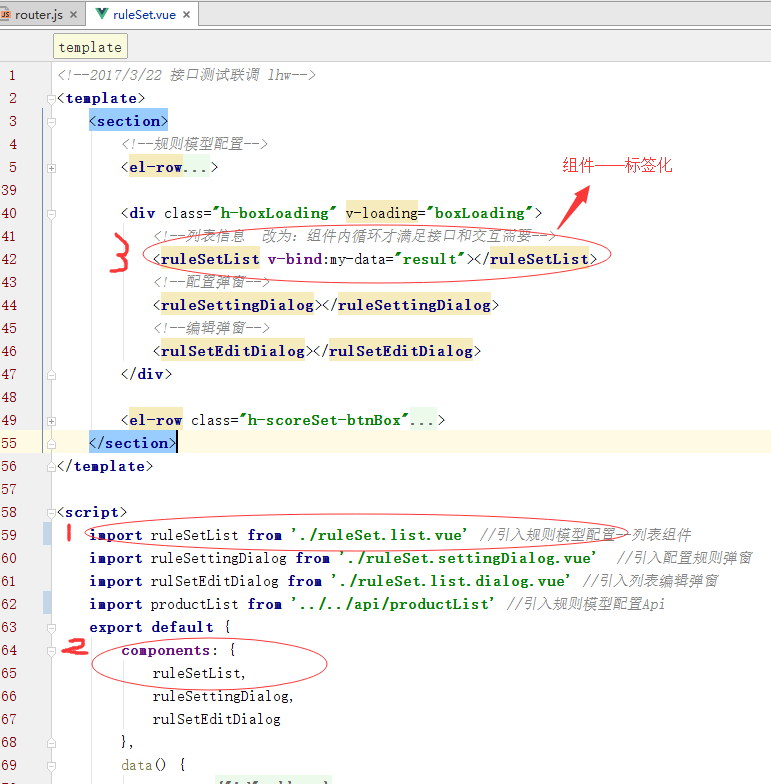
下图是vue官网对组件的理解：



---------Talk is cheap,show me the code-------------------

组件化——前端中“组件化”这个词，在UI这一层通常指“标签化”，也就是把大块的业务界面，拆分成若干小块，然后进行组装。

代码示例：



<**ruleSetList v-bind**:my-data=**"result"**></**ruleSetList**>

**import** ruleSetList **from './ruleSet.list.vue'** *//引入列表组件*

看下jsp做静态页面生成的时候，就可以利用jsp的<jsp:include page=”xxx.jsp”/>

指令进行引入公共组件。

再或者，跟thinkpphp 模板中include引入其他模板文件，如：

<include file="Public/header" /> // 包含头部模版header

模块化——所谓的模块化开发就是封装细节，提供使用接口，彼此之间互不影响，每个模块都是实现某一特定的功能。如：CommonJS和AMD。

两者比较理解：

》》》模块化：主要针对Javascript，模块化开发的基础就是函数。

》》》组件化：侧重UI层面，由html、css、js构成的独立文件。

**6. Vue-router路由的使用**

官网API：<https://router.vuejs.org/zh-cn/installation.html>

6.1 NPM安装：npm install vue-router

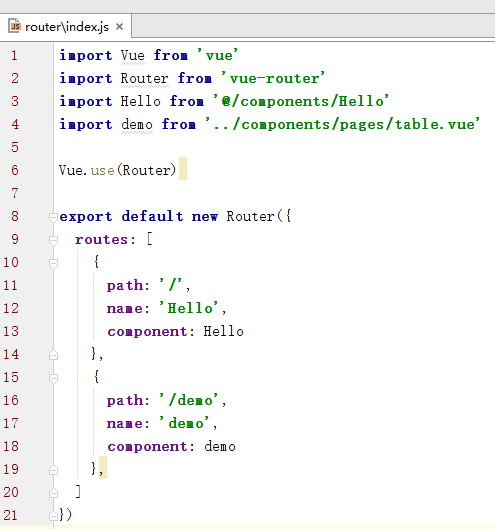
在如果在一个模块化工程中使用它，必须要通过Vue.use() 明确地安装路由功能：

import Vue from ‘vue’

import VueRouter from ‘vue-router’

Vue.use(VueRouter)

下面贴上demo代码：



6.2 路由的跳转方式和路由传参：

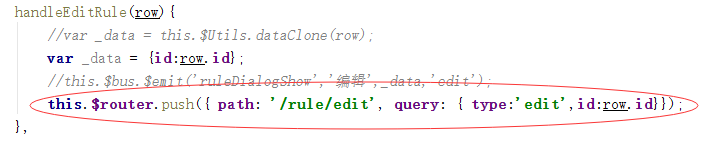
》使用<router-link>标签，相当于<a>标签，其中to属性等于href

<**router-link** to=**"/main"** class=**"project\_name"**>{{ProjectName}}</**router-link**>

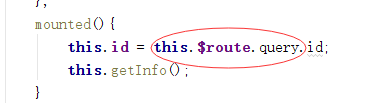
》使用js跳转：

**this**.**$router**.push({ **path**: **'/productList'**});

》路由传参数跳转：



》》目标页面接受参数方法：



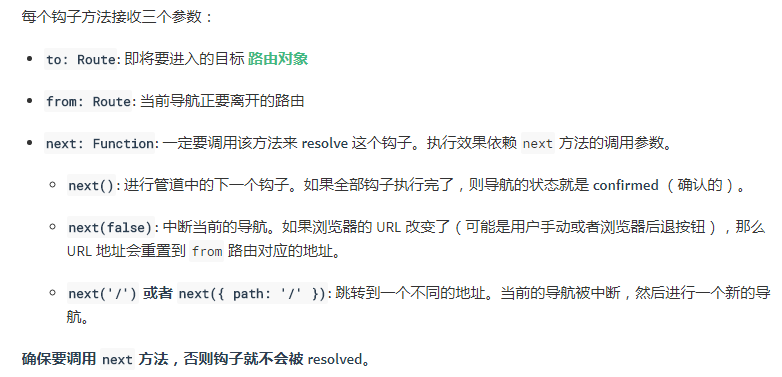
导航钩子：vue-router 提供的导航钩子主要用来拦截导航，让它完成跳转或取消。

const router = new VueRouter({ ... })

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

// ...

})



使用场景：如判断用户是否登录跳转和权限页面跳转控制。参考main.js

**7. axios基本用法及解决ajax跨越请求配置**

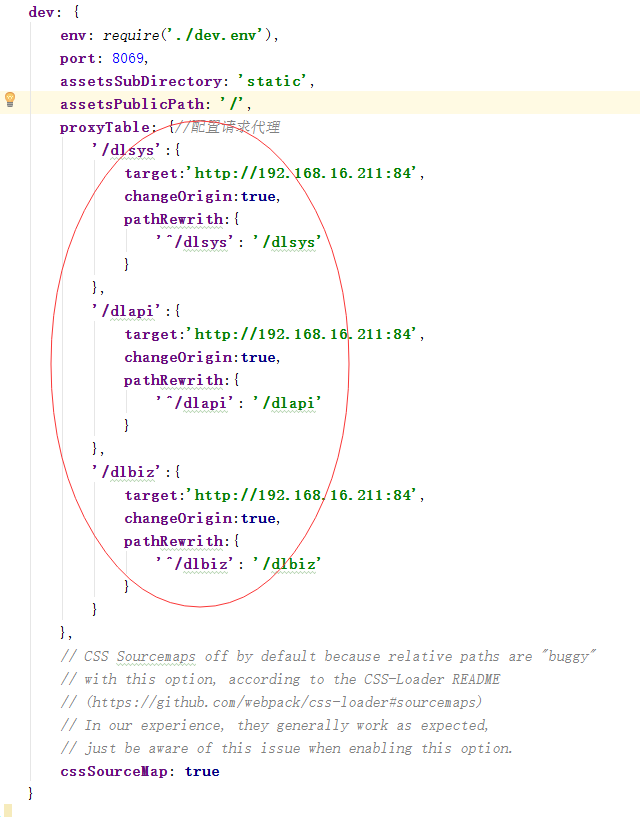
Github：<https://github.com/mzabriskie/axios>

7.1 用法简单，看文档~



7.2 解决ajax跨域问题

在config目录下的index.js配置proxyTable参数



上面的代理，其实Apache反向代理差不多



others：

1、vue-resources 有jsonp的方法

//1.jsonp不能发post请求，不管是否跨域，只要用jsonp方式就只能是get，因为本质是script方式加载的。

2、后端 response header设置，

Access-Control-Allow-Origin: http://xxx.com，允许来自xxx的跨域

3、现代的浏览器，跨域优先考虑Cross-Origin-Resource-Sharing. IE 11就全面支持了。

**8.Element UI基本用法介绍**

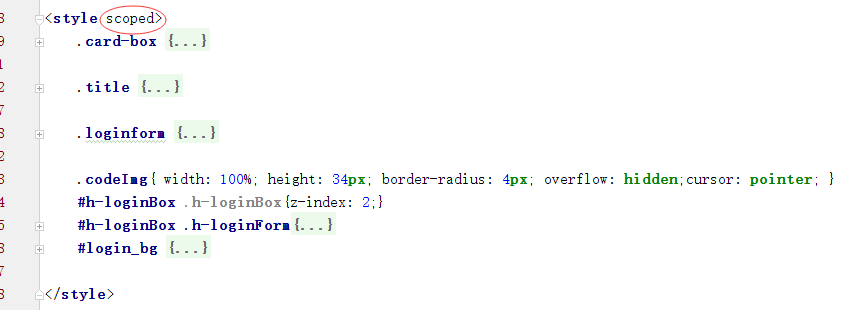
官网API: <http://element.eleme.io/1.3/#/zh-CN/component/installation>

8.1升级注意事项

见：<https://my.oschina.net/hgwn/blog/896780>

8.2组件内样式

通常，组件中标签的样式是全局的，在使用第三方UI库（如element），全局样式很可能影响UI库的样式。可通过添加scoped属性来是style中的样式只作用于当前组件，如下图：



注意：在有scope属性的style标签内导入其他样式（@import ‘xxxx.css’），同样会受限于作用域，变为组件内样式。 //复用程度较高的样式不建议这样使用

另外：在组件内样式中应避免使用元素选择器，性能大大降低。

反过来：若是想覆盖掉element默认样式，则在组件去掉scoped属性 + important，当然建议在全局样式吧。

8.3缺点：vs ivew

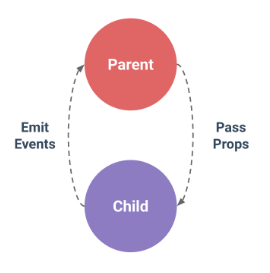
ivew <https://www.iviewui.com/>

**9. Vue.js项目开发心得和遇到坑**

9.1如何解决组件之间通信问题

9.1.1父组件和子组件之间通信

父子组件的关系可以总结为 props down, events up。父组件通过 props 向下传递数据给子组件，子组件通过 events 给父组件发送消息。



1）使用Prop传递数据，见官网demo：

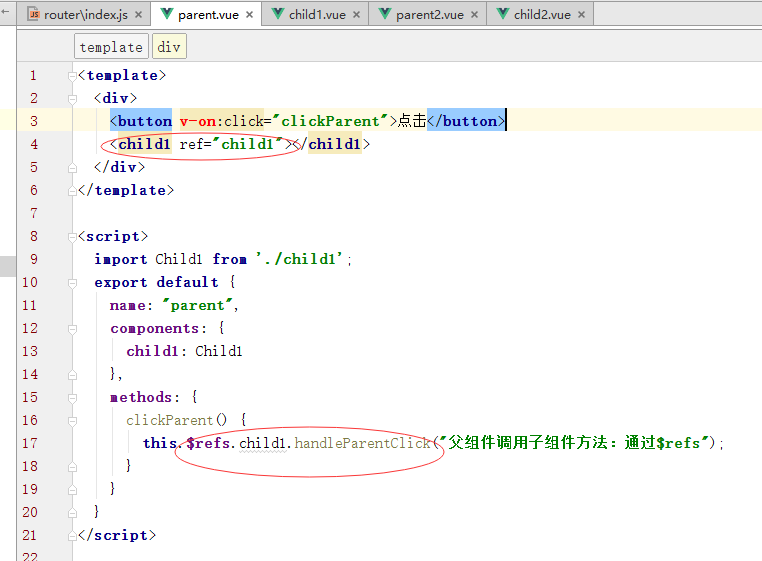


2）父组件调用子组件方法：通过$refs

实例，见demo 效果：



parent.vue源码：



child1.vue组件源码：



3）子组件向父组件通信

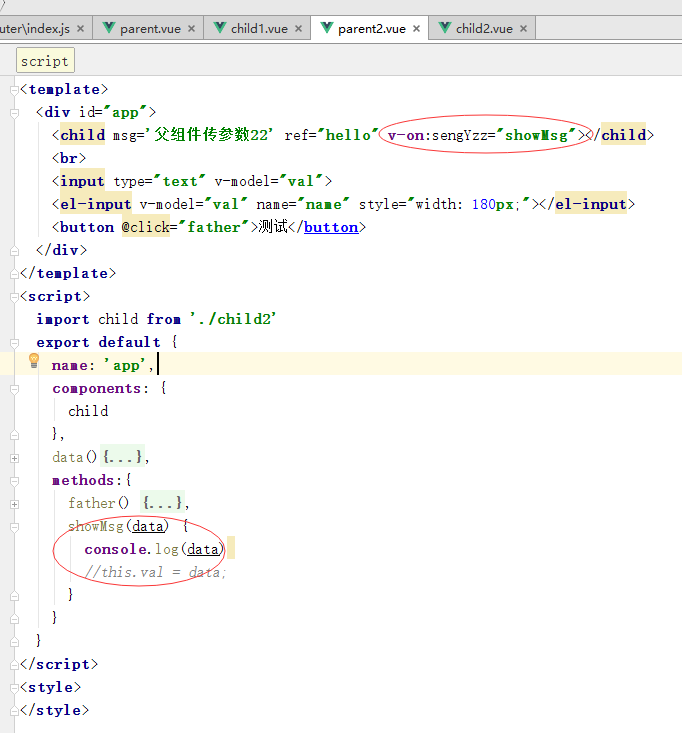
使用 $on(eventName) 监听事件

使用 $emit(eventName) 触发事件

*官网 https://cn.vuejs.org/v2/guide/components.html#自定义事件*

实例demo源码：

*parent.vue父组件*



*child2.vue子组件*



9.1.2非父子关系组件又该如何通信呢？

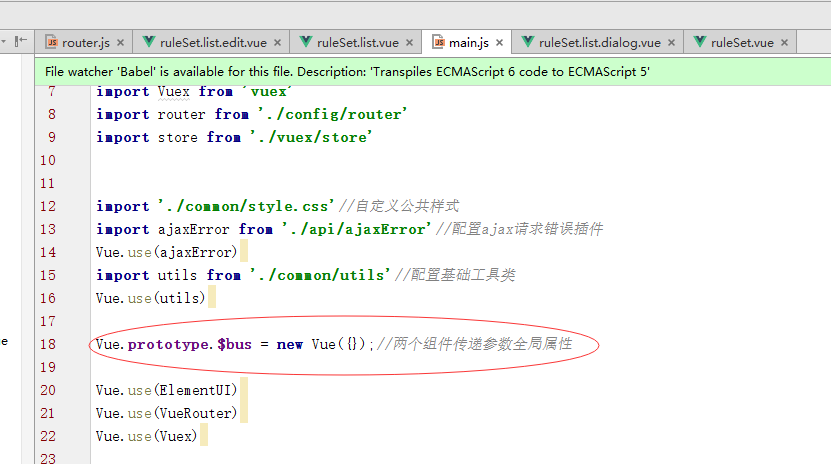
方法一、事件总线模式

可以理解为：订阅发布模式

场景：数据列表 list.vue组件和编辑弹窗 model.vue组件，单击列表某条数据，然后在弹窗显示编辑。

看代码吧：

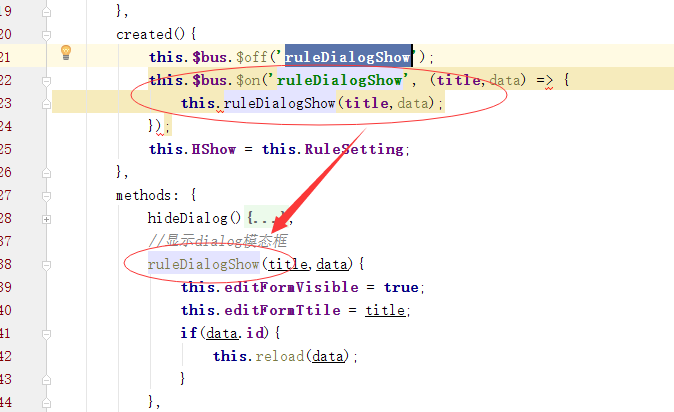
1)在main.js中，Vue对象添加prototype原型属性$bus



2)被使用组件，在created()钩子使用$on添加自定义事件回调函数，在methods:{

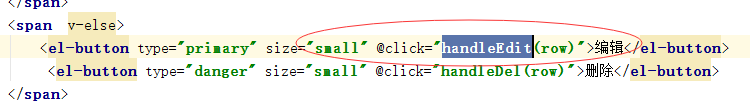
//接受参数，自行处理函数

}，见项目截图：

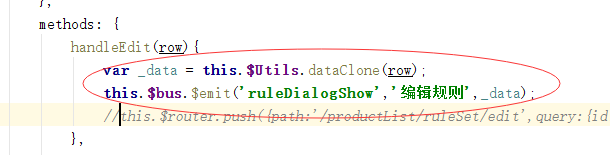


3）使用组件，用$emit(‘自定义函数名称’,参数)，见截图：

Html:



JS：



方法二、使用vuex

应用场景：规则配置自定义是否显示或隐藏---在数字字典里面进行配置。

简单使用，不废话，看代码——talk is cheap,show me the code!

1）在新建vuex目录下，创建store.js，引入vue和vuex，定义应用初始状态/所需的mutations，最后export store的实例，见截图：



2）在main.js引入上面创建的store.js组件，见截图：



3）更新state状态

如在Home.vue公共组件，通过ajax 调用API接口获取返回的用户自定义的信息，并更新到store实例的状态。

Home.vue组件，部分源码：

1、**import** { mapState } **from 'vuex'**

2、**this**.**$store**.commit(**'UPDATE\_RULESETTING'**, \_val);

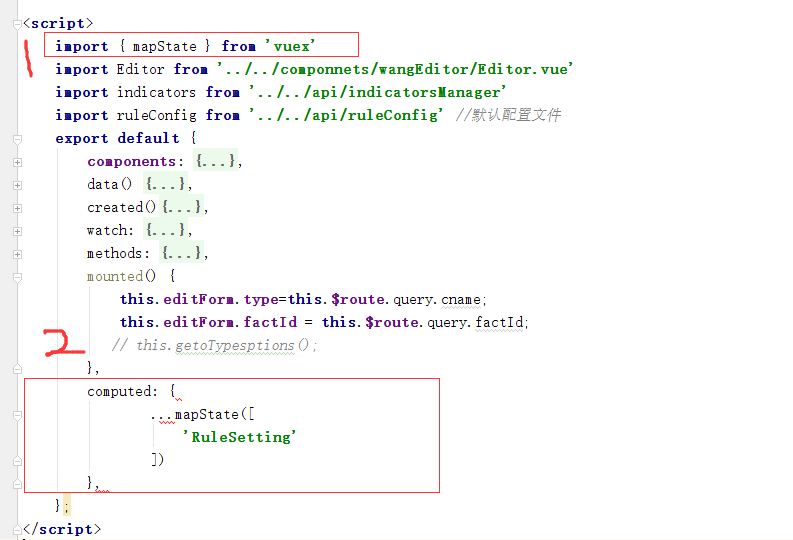
代码截图：

 4)使用状态，见部分源码：

Html部分：

<**el-row** v-show=**"RuleSetting"**>  
 <**el-col** :span=**"24"**>  
 <**el-form-item** label=**"指标逻辑"** prop=**"expression"**>  
 <**v-editor** :input-content=**"inputContent"** :upload-url=**"uploadUrl"** v-model=**"editForm.expression"** name=**"expression"**></**v-editor**>  
 </**el-form-item**>  
 </**el-col**>  
</**el-row**>

JS: 见下图



*关于vuex具体问题见9.2讨论*

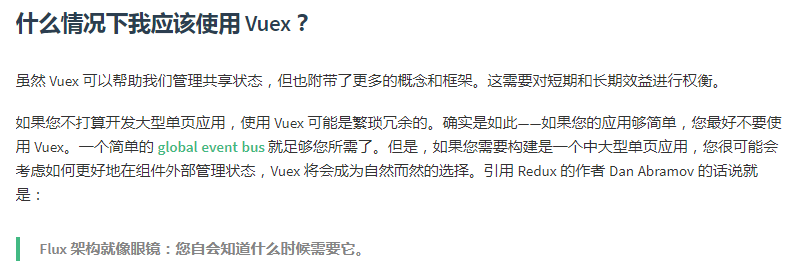
方法三、VUE2.4.0版本提供了一种新的解决方案

大概思路：首先组件支持inheritAttrs的选项，其次需要用到实例属性$attrs。

9.2 vuex是什么，解决什么问题？适用我们的项目开发吗？？

vuex是什么<https://vuex.vuejs.org/zh-cn/intro.html>

官网：Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。



9.3 vue开发调试

一、github下载安装vue-devtools

1.github下载地址：https://github.com/vuejs/vue-devtools

有Git的同学直接 git clone https://github.com/vuejs/vue-devtools

2.下载完成之后，打开cmd进入vue-devtools文件夹，把依赖装好npm install 之后再进行npm run build

3.然后打开shells>chrome>src>manifest.json

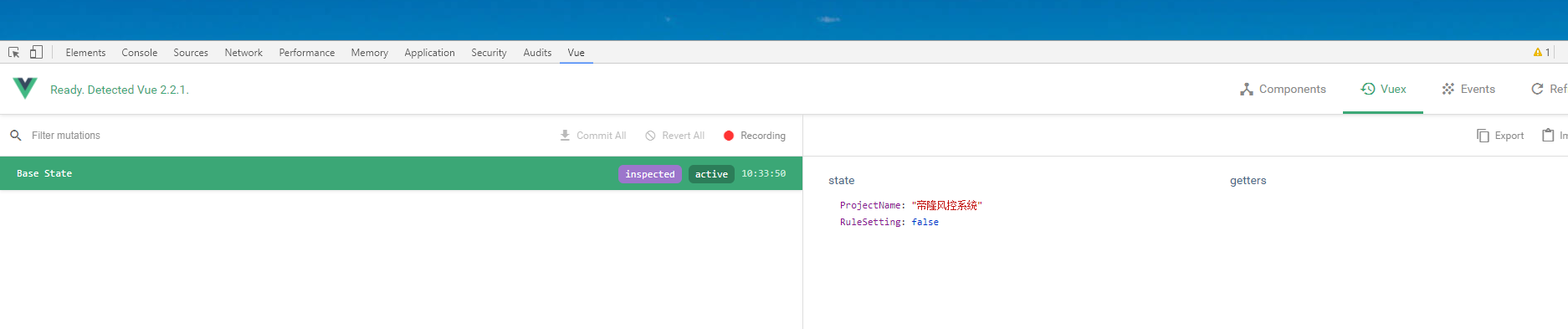
把里面的"persistent": false改为true

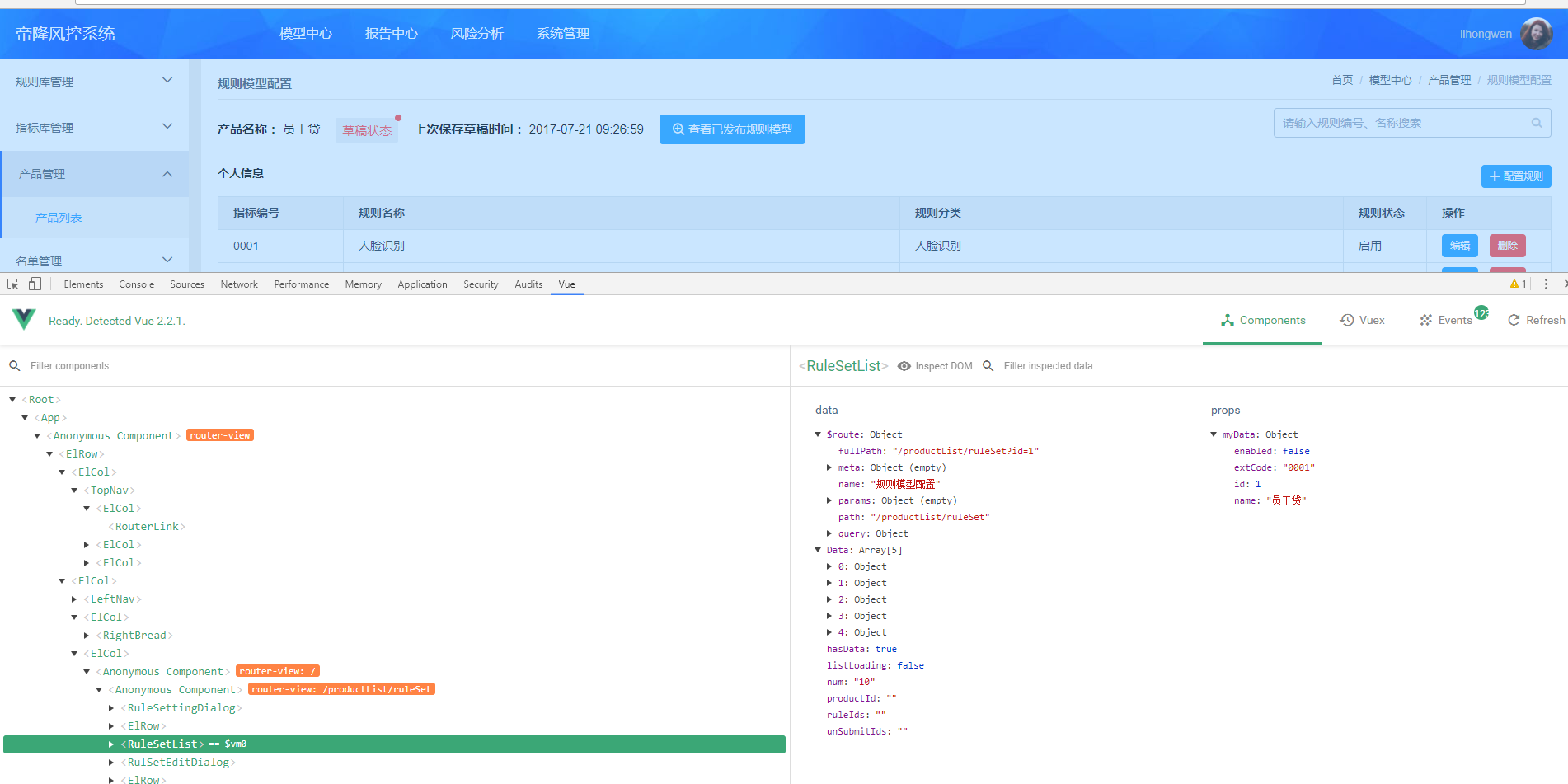
二、配置chrome浏览器

打开chrome

1.打开里面的设置 > 点击扩展程序 > 点击开发者模式

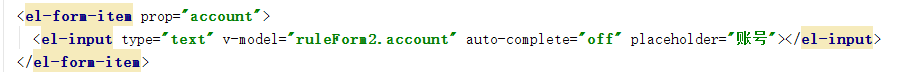
2.再点击加载已解压的扩展程序，然后把shells>chrome这个文件夹放入就ok了





9.4 Element UI在IE9遇到的坑~

9.4.1、input值变化



对比：

<**div** class=**"el-input"**>  
 <**input** type=**"password"** name=**"password"** v-model=**"ruleForm2.checkPass"** auto-complete=**"off"** placeholder=**"密码"** class=**"el-input\_\_inner" v-on**:keyup.enter=**"handleSubmit2"**>  
</**div**>

9.4.2、Element UI 无法兼容E9上传问题

问题遗留：<https://github.com/ElemeFE/element/issues/3643>

其实最近很火iview UI也是没有解决ie9上传问题~

<https://www.iviewui.com/components/upload>

9.5项目路径配置

当我们把代码放到服务器上时，有时候会遇到静态资源引用错误，导致界面一片空白的问题。

这是由于 vue-cli 默认配置的 webpack 是以站点根目录引用的文件，然而有时候我们可能需要把项目部署到子目录中。

我们也可通过 config/index.js 来修改文件引用的相对路径：



可看出来，导出对象中 build 与 dev 均有 assetsSubDirectory、assetsPublicPath 这两个属性。

其中 assetsSubDirectory 指静态资源文件夹，也就是打包后的　js、css、图片等文件所放置的文件夹，这个默认一般不会有问题。

assetsPublicPath 指静态资源的引用路径，默认配置为 /，即网站根目录，与 assetsSubDirectory 组合起来就是完整的静态资源引用路径 /static。

所以，解决办法是：只要把根目录改为相对目录就好了。

将build对象的属性

assetsPublicPath: **'/'**,

修改为：

assetsPublicPath: **'./'**,

没看错！就是一个.的问题，O(∩\_∩)O~

~~~

如果没有条件翻墙的话，又想谷歌的话，建议用下面这个搜索：

<https://g.sov5.cn/>



**Q & A**

1. 前端代码规范问题？css命名/组件命名/代码注解等等~~
2. 引入css模块化，sass在开发环境（没有build） ie9下怎么调试呢？
3. 当调用组件API返回的数据不符合前端组件的时候，一般都是在客户端浏览器这边做转换，能否在服务端增加一个中间层/控制器，处理复杂的数据转换工作呢？
4. vue需要在服务器渲染呢? How do ?
5. ~~~头脑风暴，开始讨论~~~