环境：

Node v5.1.1

Npm 3.3.12

当前我们company情况,使用jquery，没有使用mvc框架

1、需要提取公共js模块如common.js

2、提取公共css模块如cssreset.css，并且css作为单独文件

3、css采用scss编译

4、代码很多得不到继承公用

5、不是嵌入app的h5,大多数供浏览器解析；不需要一次性加载其他还没有用到的资源，否则会很大，启用多页面开发而非单页面开发，也是为了适应以前跳转页面刷新资源。

[**命令行生成vue2.0 -cli项目框架**](http://blog.csdn.net/zzwwjjdj1/article/details/52687488)

1. npm install -g vue-cli
2. vue init webpack vue2.0(默认安装2.0，安装这个)

vue init webpack#1.0 vue2.0(安装1.0)

1. npm install(安装依赖的东西)
2. npm run dev运行开发服务器环境
3. npm run build 生成生产环境

提醒

1、1个cli项目只有一个index.html,App.vue,main.js

2、所有组件放在：index.html中div#app里面

**文档：**

2.0官网文档：<https://vuejs.org/guide/>

2.0中文文档：<https://vuefe.cn/guide/>

**定义与使用组件步骤：**

1、定义组件，格式包括<template>,<script>,<style>

2、引入组件 import

3、注册组件 components

4、使用组件 <firstcomponent></firstcomponent>

app.vue（使用FirstComponent组件）

<template>  
 <div id="app">  
 <img src="./assets/logo.png">  
 <FirstComponent></FirstComponent>  
 </div>  
</template>  
<script>  
**import** FirstComponent **from** './components/FirstComponent'  
**export default** {  
 name: 'app',  
 components: {  
 FirstComponent  
 }  
}

或者另外一个注册组件方式

**import** homeHeader **from** '../../components/home/home-header.vue'

components: {  
 'home-header': homeHeader  
}

使用（不要与标签名字比如footer一样）

<home-header></home-header>

FirstComponent.vue（自己定义的）

<template>  
<h2>{{text}}</h2>  
</template>  
<script>  
 **export default**{  
 data () {  
 **return** {  
 text: '这是我的第一个组件'  
 }  
 }  
 }  
</script>  
<style scoped>  
 h2{  
 background: red;  
 }  
</style>

**项目使用SASS语法**

1. npm install node-sass –save-dev
2. npm install sass-loader –save-dev
3. webpack配置

{  
 test: /\.scss$/,  
 loader: "style!css!sass"  
},

1. 使用

<style lang='scss'>  
@import "./assets/scss/test.scss";//引入外部scss文件  
#app1 {  
 font-family: 'Avenir', Helvetica, Arial, sans-serif;  
 -webkit-font-smoothing: antialiased;  
 -moz-osx-font-smoothing: grayscale;  
 text-align: center;  
 color: #2c3e50;  
 margin-top: 60px;  
 & .red{  
 color:red;  
 }  
}  
</style>

1. autoprefixer的添加

webpack配置

vue: {  
 loaders: utils.cssLoaders(),  
 postcss: [  
 require('autoprefixer')({  
 browsers: ['last 2 versions','Android >= 4.0','iOS >= 7']  
 })  
 ]  
}

这个效果有用（下面代码，用的是css而不是scss）

<style scoped>  
 h2{  
 background: red;  
 display: flex;  
 }  
</style>

生成

<style type="text/css">  
 h2[data-v-34c8e960]{  
 background: red;  
 display: -webkit-box;  
 display: -webkit-flex;  
 display: -ms-flexbox;  
 display: flex;  
 }  
</style>

使用scss

<style lang='scss'>  
@import "./assets/scss/test.scss";  
#app1 {  
 font-family: 'Avenir', Helvetica, Arial, sans-serif;  
 text-align: center;  
 color: #2c3e50;  
 margin-top: 60px;  
 diplay:flex;  
 animation-name: dropAnim1;  
 & .red{  
 color:red;  
 }  
}  
</style>

生成效果

<style type="text/css">  
 html, body {  
 background: #fff;  
 -webkit-animation-name: dropAnim1;  
 animation-name: dropAnim1;  
 display: -webkit-box;  
 display: -webkit-flex;  
 display: -ms-flexbox;  
 display: flex;  
 }  
 #app1 {  
 font-family: 'Avenir', Helvetica, Arial, sans-serif;  
 text-align: center;  
 color: #2c3e50;  
 margin-top: 60px;  
 display: -webkit-box;  
 display: -webkit-flex;  
 display: -ms-flexbox;  
 display: flex;  
 -webkit-animation-name: dropAnim1;  
 animation-name: dropAnim1;  
 }  
 #app1 .red {  
 color: blue;  
 }  
</style>

解决连接：

<https://github.com/vuejs/vue-loader/issues/227>

<https://github.com/vuejs/vue-loader/issues/234>

<https://github.com/vuejs/vue-loader/issues/424>

**如何把css作为一个单独的文件，而不是以style插入html里面**

1、

**路由vue-router**

1、安装

npm install vue-router --save-dev

2、使用

在main.js中引入

**import** Vue **from** 'vue'  
**import** App **from** './App'  
**import** VueRouter **from** 'vue-router'  
Vue.use(VueRouter)  
// 定义组件, 也可以像教程之前教的方法从别的文件引入  
**const** First = { template: '<div><h2>我是第 1 个子页面</h2></div>' }  
**const** Second = { template: '<div><h2>我是第 2 个子页面</h2></div>' }  
// 创建一个路由器实例  
// 并且配置路由规则  
**const** router = **new** VueRouter({  
 mode: 'history',  
 base: \_\_dirname,  
 routes: [  
 {  
 path: '/first',  
 component: First  
 },  
 {  
 path: '/second',  
 component: Second  
 }  
 ]  
})  
// 路由器会创建一个 App 实例，并且挂载到选择符 #app 匹配的元素上。  
**new** Vue({  
 router: router,  
 render: h => h(App)  
}).$mount('#app')

App.vue

<template>  
 <div id="app">  
 <img src="./assets/logo.png">  
 <FirstComponent></FirstComponent>  
 <ul>  
 <li><router-link to="/first ">点我跳转到第一页</router-link></li>  
 <li><router-link to="/second">点我跳转到第二页</router-link></li>  
 </ul>  
 <router-view></router-view>  
 </div>  
</template>

导入css

**import** './assets/css/test.css'

**导入jquery2.2.4**

npm install [jquery@2.2.4](mailto:jquery@2.2.4) –save-dev

在index.vue引入

**import** $ **from** 'jquery'

定义没有使用的话会报错

'$' is defined but never used

**导入依赖jquery的第三方库如jQuery.md5.js**

方法一：

1. webpack.base.conf.js

**var** webpack = require("webpack")//加入

resolve: {  
 extensions: ['', '.js', '.vue'],  
 fallback: [path.join(\_\_dirname, '../node\_modules')],  
 alias: {  
 'vue$': 'vue/dist/vue',  
 'src': path.resolve(\_\_dirname, '../src'),  
 'assets': path.resolve(\_\_dirname, '../src/assets'),  
 'components': path.resolve(\_\_dirname, '../src/components'),  
 'md5': path.resolve(\_\_dirname, '../src/assets/js/lib/jquery/jQuery.md5.js'), //使用别名，其他js文件就可以使用import md5 from ‘md5’ （这个‘md5’资源指向是这里）  
 'jquery':'jquery'//加入（如果是npm安装的话，可以省略）  
 }  
},

plugins: [  
 **new** webpack.ProvidePlugin({//全局暴露（否则jQuery is not defined报错），加入  
 $: "jquery",  
 jQuery: "jquery"  
 })  
],

1. index.vue使用

<template>  
 <div class="container">  
 <home-header></home-header>  
 <home-banner></home-banner>  
 <button v-on:click="addTodo" class="btn">点击</button>  
 </div>  
</template>  
<script>  
 **import** $ **from** 'jquery'  
 **import** 'md5'  
 **import** homeHeader **from** '../../components/home/home-header.vue'  
 **import** HomeBanner **from** '../../components/home/home-banner.vue'  
 **export default**{  
 data () {  
 **return** {  
 msg: 'Welcome to Your Vue.js App'  
 }  
 },  
  
 components: {  
 'home-header': homeHeader,  
 'home-banner': HomeBanner  
 },  
  
 methods: {  
 addTodo: **function** () {  
 console.log($)  
 window.alert($('.btn').get(0).tagName)  
 console.log($.md5('a'))  
 }  
  
 }  
 }  
</script>

方法二：

1、webpack.base.conf.js，loaders加入

{  
 test: require.resolve('jquery'),//添加变量  
 loader: 'expose?$!expose?jQuery!jquery'  
},

2、index.vue

**import** $ **from** 'expose?$!expose?jQuery!jquery'  
**import** 'md5'

但是报错了（因为这里用了npm安装jquery）

Module build failed: Error: jQuery requires a window with a document

解决方法：你可以在alias修改jquery指向路径

'jquery': path.resolve(\_\_dirname, '../src/assets/js/lib/jquery/jquery'),//如果是本地的话

参考地址：

<https://github.com/yiifaa/yii-template>

[http://webpack.github.io/docs/shimming-modules.html#exporting](http://webpack.github.io/docs/shimming-modules.html%23exporting)

方法三：

希望引用CDN资源，步骤如下：

1、webpack.base.conf.js

externals: {  
 wx:'jWeixin',  
 'jquery':'jQuery',  
},

2、index.vue

**import** $ **from** 'jquery'  
**import** 'md5'

**3、html加上**

<script src="//cdn.bootcss.com/jquery/2.2.4/jquery.min.js"></script>

**疑问，**貌似上面的配置在每个组件上都要import或者require jquery，在想有没有一种方法可以在main.js引入，其他组件就不用引入呢？

注意：不要对lib文件下进行语法检查，否则一大堆错误，npm run dev不通过，方法参照下面

**md5，base64，jquery.cookies合并成一个文件**

1、

entry: {  
 app: './src/main.js',  
 vendor:['md5','base64']  
},

2、

alias: {,  
 'base64': path.resolve(\_\_dirname, '../src/assets/js/lib/base64'),  
 'md5': path.resolve(\_\_dirname, '../src/assets/js/lib/jquery/jQuery.md5.js')  
}

之后会自动插入到html中

多个合并成一个文件导致的问题：（尚未解决）

Uncaught ReferenceError: base64 is not defined

参考：

<https://github.com/webpack/webpack/tree/master/examples/multiple-commons-chunks>

**导入第三方库flexible.js（放在项目本地），这是不依赖任何框架的。**由于ES6语法检查严格，导致整个flexible很多报错，先关了找到webpack.base.conf.js（**不建议**）

preLoaders: [  
 {  
 test: /\.vue$/,  
 loader: 'eslint',  
 include: projectRoot,  
 exclude: /node\_modules/  
 },  
 {  
 test: /\.js$/,  
 loader: 'eslint',  
 include: projectRoot,  
 exclude: /node\_modules/  
 }  
],

重启，然后在main.js加上就可以运行了

require('./assets/js/lib/flexible/flexible.js')

**这里是附加说明：**

你可以针对某个文件夹不对他进行eslint语法检查，可以在项目中找到.eslintignore后面添加这句话：

build/\*.js  
config/\*.js  
src/assets/js/lib/\*\*/\*.js

我的目录结构是这样的：



│ .babelrc

│ .editorconfig

│ .eslintignore

│ .eslintrc.js

│ .gitignore

│ index.html

│ package.json

│ README.md

├─build

│ build.js

│ dev-client.js

│ dev-server.js

│ utils.js

│ webpack.base.conf.js

│ webpack.dev.conf.js

│ webpack.prod.conf.js

│

├─config

│ dev.env.js

│ index.js

│ prod.env.js

│ test.env.js

│

├─node\_modules

│

├─note

│ byluchanan.docx

│

├─src

│ │ App.vue

│ │ filters.js

│ │ main.js

│ │ routers.js

│ │

│ ├─assets

│ │ │ logo.png

│ │ │

│ │ ├─css

│ │ │ test.css

│ │ │

│ │ ├─font

│ │ │ test.css

│ │ │

│ │ ├─images

│ │ │ └─icon

│ │ │ dot.png

│ │ │ dot\_on.png

│ │ │ search.png

│ │ │ service.png

│ │ │

│ │ ├─js

│ │ │ │ test.css

│ │ │ │

│ │ │ └─lib

│ │ │ │ base64.js

│ │ │ │

│ │ │ └─jquery

│ │ │ jquery.cookie.js

│ │ │ jquery.js

│ │ │ jQuery.md5.js

│ │ │

│ │ └─scss

│ │ │ common.scss

│ │ │ index.scss

│ │ │

│ │ └─base

│ │ \_common.scss

│ │ \_cssreset.scss

│ │ \_extend.scss

│ │ \_footer.scss

│ │ \_function.scss

│ │ \_header.scss

│ │ \_mixin.scss

│ │ \_necessary.scss

│ │ \_variable.scss

│ │

│ ├─components

│ │ │ Hello.vue

│ │ │

│ │ ├─common

│ │ │ search-icon.vue

│ │ │ service-icon.vue

│ │ │

│ │ └─home

│ │ home-banner.vue

│ │ home-header.vue

│ │ home-topic.vue

│ │

│ ├─views

│ │ └─home

│ │ index.vue

│ │

│ └─vuex

│ test.css

│

├─static

│ │ .gitkeep

│ │

│ ├─api

│ │ │ .gitkeep

│ │ │

│ │ └─Home

│ │ │ homepageV3.json

│ │ │

│ │ └─images

│ │

│ └─js

│ └─lib

│ │ base64.js

│ │

│ └─flexible

│ flexible.js

│

└─test

├─e2e

│ │ nightwatch.conf.js

│ │ runner.js

│ │

│ ├─custom-assertions

│ │ elementCount.js

│ │

│ └─specs

│ test.js

│

└─unit

│ .eslintrc

│ index.js

│ karma.conf.js

│

└─specs

Hello.spec.js

据说也可以在build/webpack.base.conf.js修改exclude: /node\_modules/改为

exclude:[/node\_modules/,/src/]貌似只在**根目录**可以（还配置alias指向'libs': path.resolve(\_\_dirname, '../libs')使用：require('libs/js/flexible.js')或者import ' libs/js/flexible.js '），在**子目录不可以（~~也许有解决方法，暂时没头绪~~，现在有解决方法了）**

exclude:[/node\_modules/,/src\/assets\/js\/lib/]，还是检查语法了，有了解决方法（**最新**）：excluede要用的是具体路径，里面不是包含正则的

**var** libs=path.resolve(\_\_dirname, '../src/assets/js/lib/')

使用

exclude: [/node\_modules/,libs]

虽然说关闭（webpack.base.conf.js注释）可以解决引入成功，但是也导致了关闭其他用ES6语法检测错误，不建议这样使用，目前**暂时（下面的方法可以关闭语法检查）**我的解决方法是把它们放到**static**文件夹下，然后在index.html引入，

<script src="static/js/lib/flexible/flexible.js"></script>

如果放在src文件下只能通过require或者import的方式来引入，如果用下面的方法，是引入不了资源的（找不到路径）！

<script src="./src/assets/js/lib/flexible/flexible.js"></script>

另外**注意**，如果不是放在static，比如远程地址的话（index.html写上下面代码），而又想在src下引入微信jssdk的话，就要这样配置，即在.vue文件中引入第三方非NPM模块。

<script src="http://res.wx.qq.com/open/js/jweixin-1.0.0.js"></script>

1、webpack.base.conf.js加上（修改这里面的东西要重新npm run dev才生效）

externals: {  
 wx:'jWeixin'  
},

jWeixin为http://res.wx.qq.com/open/js/jweixin-1.0.0.js暴露的名字，wx为import名字

2、index.html加上

<script src="http://res.wx.qq.com/open/js/jweixin-1.0.0.js"></script>

3、需要的地方使用如main.js

**import** wx **from** 'wx'  
console.log(wx)

**怎么请求后台数据并且嵌入html中**

**1、npm install vue-resource --save-dev**

**2、main.js**

**var** vueResource = require('vue-resource')  
Vue.*use*(vueResource)

**3、index.vue请求数据**

注意：模拟数据json文件要放在static并且要用get方式（POST会出现404）才能访问

**export default**{  
 data () {  
 **return** {  
 bannerList: []  
 }  
 },  
 mounted: **function** () {  
 **this**.$http.post('static/api/Home/homepageV3.json

').then((response) => {  
 // 为bannerList赋值数据，然后html就可以遍历  
 **this**.bannerList = response.data.banner\_list  
 }, (response) => {  
 console.log(response)  
 })  
 }  
}

**4、插入html**

<swipe-item class="slide" v-for="banner in bannerList"><img v-bind:src=banner.image /></swipe-item>

**vue插入style为background的代码**

v-bind:style="{ 'background-image': 'url(' + banner.image + ')' }"

使用第三方组件的参数

<swipe class="my-swipe" :auto="5000">

Class使用if进行多条件判断

之前使用artTemplate中class判断是这样的

<div class="item {{if value.data\_type==2}}destination{{/if}}{{if value.display\_type==3&&value.data\_type==1}}topic{{/if}}{{if value.display\_type==3&&value.data\_type==0}}activity{{/if}}{{if value.display\_type==2&&value.data\_type==1}}product{{/if}}">

现在使用VUE

<div v-bind:class="{'item': list.data\_type==2,  
 'item surprise':list.data\_type==1&&list.display\_type==3,  
 'item surprise activity':list.data\_type==0&&list.display\_type==3,  
 'item surprise list':list.data\_type==1&&list.display\_type==2}" v-for="list in indexData.data\_list">

参考

<http://www.blogs8.cn/posts/WcqJb2e>

项目使用vue lazyload

1、main.js

**import** VueLazyload **from** 'vue-lazyload'  
Vue.*use*(VueLazyload)

2、index.vue(line.image遍历数据)

<div class="bg cover bg\_lazyload" v-lazy:background-image="line.image"></div>

**参考资料：**

大致了解vue.js -cli

<http://www.larabase.com/collection/3/post/128>

[Vue2.0 新手完全填坑攻略——从环境搭建到发布](http://www.jianshu.com/p/5ba253651c3b)

[Vue.js 2.0 基本入门](http://www.thatyou.cn/vue-js-2-0-%E5%9F%BA%E6%9C%AC%E5%85%A5%E9%97%A8/)

[国内有哪些公司在用Vue.js，有什么心得?](https://www.zhihu.com/question/38213423)

[Vue 组件化开发实践](http://gold.xitu.io/entry/55f77eb460b28e6a6f0f4f86)

[Vue + webpack 项目实践](http://jiongks.name/blog/just-vue/)

[webpack入坑之旅（六）配合vue-router实现SPA](http://www.tuicool.com/articles/yI322iq)

[vue-router 快速入门](http://www.cnblogs.com/axl234/p/5899952.html)

[一小时包教会 —— webpack 入门指南](http://www.tuicool.com/articles/fiyQvq7)

IDE工具

<http://www.jianshu.com/p/142dae4f8b51>

e6相关

[VUE.JS组件化开发实践](http://www.cnblogs.com/rock-roll/p/5692891.html)

[Laravel 5.1 中使用 Vue2.0 组件化开发（配置）](http://www.open-open.com/lib/view/open1471589613718.html)

[vue.js2.0路由文档](http://router.vuejs.org/)

[Vue.js 开发实践：实现精巧的无限加载与分页功能](https://segmentfault.com/a/1190000005351971)

[用vue+webpack+es6+sass实现Cnode网站(一)](https://segmentfault.com/a/1190000006000118)

[在VUE2.0中使用PostCSS](http://gold.xitu.io/post/581bfc368ac247004fe174af)

[jQuery转Vue项目实践总结](http://blog.csdn.net/violetjack0808/article/details/51451672)

github源码参考

<https://github.com/825618507/vue-app>（入门适合）

<https://github.com/zerqu/qingcheng>

[webpack+vue.js重构的cnode社区的单页面(spa)应用](https://github.com/cwsjoker/Cnode-vue-spa)

[知乎日报](https://github.com/luchanan/vue-zhihu-daily)

<https://github.com/bluefox1688/vue-cli-multi-page>

<https://github.com/jiananle/vue-multi-page>

[微信高保真vuejs1.0](https://vue-wechat.github.io/)

技巧：

1、为当前目录生成tree，并且保存为tree.txt，tree命令生成文件夹，tree /f生成文件夹和文件下所有文件（tree只生成目录）

F:\person\_project\github\vuejs\_learn>tree /f >F:\person\_project\github\vuejs\_lea  
rn\tree.txt

2、npm run build之后如何运行项目dist

2.1、先安装

npm install -g http-server

2.2、进入打包后的项目运行，然后输入地址就好了

http-server

3、在index.vue中<style>写html,body的样式貌似没有用(不知为何)，只能写container（包含）里面的样式

<template>  
 <div class="container padding\_b\_120">  
 <home-header></home-header>

</div>

</temlate>

<style lang='scss' scoped>  
 @import "../../assets/scss/base/necessary.scss";  
 html,body{  
 background: $white;  
 }

</style>

多页面应用开发

1. webpack使用ssi include方式：使用html-loader来部分代替SSI（在.html文件中使用include的方式暂时没有找到很好的办法）

**import** c **from 'components/common/notice.html'**;

//上面代码会解析为字符串

<**template**>  
 <**div class="home-container"**>  
 <**h1**>{{ msg }}</**h1**>  
 </**div**>  
</**template**>  
  
<**script**> **import** c **from 'components/common/notice.html'**;  
  
 **export default** {  
 data **()** {  
 **return** {  
 **msg**: c  
 }  
 }  
 }  
</**script**>

1. 获取多页面入口路径utils.js，把所有html文件放在views文件夹下
2. exports.getEntries = **function (**globPath,type**)** {  
    **var** entries = {}  
    **var** ishtml=type!==**undefined**?**true**:**false**;  
    glob.**sync(**globPath**)**.forEach**(function (**entry**)** {  
    **if(**ishtml**)**{  
    //html以模块文件作为输出  
    //以locahost:8080/views/index.html这样形式访问  
    **var** tmp = entry.split**('/')**.splice**(**-3**)  
    var** moduleName = tmp.splice**(**0,2**)**.join**("/")**;  
    entries**[**moduleName**]** = entry  
     
    }  
    **else**{  
    //js以模块文件作为输出,比如indx.js  
    **var** basename = path.**basename(**entry, path.extname**(**entry**))**;  
    tmp = entry.split**('/')**.splice**(**-3**)**;  
    **pathname** = tmp.splice**(**1, 1**)**;  
    entries**[pathname]** = entry;  
    }  
    }**)**;  
    **return** entries;  
   }

3、<script>插入多页面html，如果htmlWebpackPlugin没有指定chunks的话，那么多所有js会插入多页面html问题

**var** pages = utils.getEntries**('./src/views/\*\*/\*.html'**,1**)**;  
**for (var** pathname **in** pages**)** {  
 // 生成html相关配置  
 **var** conf = {  
 **filename**: pathname + **'.html'**, // html文件输出路径  
 **template**: pages**[**pathname**]**, // 模板路径  
 **inject**: **true**, // js插入位置  
 **minify**: {  
 //压缩设置  
 //removeComments: true,  
 //collapseWhitespace: true,  
 //removeAttributeQuotes: true  
 // more options:  
 // https://github.com/kangax/html-minifier#options-quick-reference  
 },  
 // necessary to consistently work with multiple chunks via CommonsChunkPlugin  
 //chunksSortMode: 'dependency'  
 };  
 pathname=pathname.split**("/")[**1**]**;//去掉views  
 **if (**pathname **in** module.exports.**entry)** {  
 conf.**inject** = **'body'**;  
 //如果每个html没有进入这里的话，那么全部js将会插入html  
 conf.**chunks** = **[**pathname, **'vendor'**, **'manifest'**, **'bootstrap']**,  
 conf.**hash** = **true**;  
 }  
 module.exports.**plugins**.push**(new** HtmlWebpackPlugin**(**conf**))**;  
}

4、针对对ES6语法检查过于严格，为此进行调整，找到.eslintrc.js

'indent': ['warn', 4],  
'semi': 0, // off semi  
'eol-last': 0,  
'no-unused-vars': 0,

1. vue中单独提取SCSS文件，这个SCSS文件background引用图片路径问题

目录结构



Comon.scss

**@charset "utf-8"**;  
**@import "base/cssreset.scss"**;  
**@import "base/necessary.scss"**;  
**@import "base/common.scss"**;  
**@import "base/header.scss"**;  
**@import "base/footer.scss"**;

\_necessary.scss

**@import "variable.scss"**;  
**@import "function.scss"**;  
**@import "mixin.scss"**;  
**@import "extend.scss"**;

在index.js引用scss

*require***('scss/home/home.scss')**

在extent.scss中图片背景无论使用那种相对路径，都不正确；

（如果生成的html里面看到background是static目录下的话才是对的，它把scss的文件路径进行了转换）

错误

&[**lazy**=**loading**]{  
 **background**:**url**(**../../images/icon/lazy\_load\_loading.png**) **scroll no-repeat center center important**();  
 **background-size**: **px2rem**(159) **important**();  
}

错误

&[**lazy**=**loading**]{  
 **background**:**url**(**../images/icon/lazy\_load\_loading.png**) **scroll no-repeat center center important**();  
 **background-size**: **px2rem**(159) **important**();  
}

错误

&[**lazy**=**loading**]{  
 **background**:**url**(**/assets/images/icon/lazy\_load\_loading.png**) **scroll no-repeat center center important**();  
 **background-size**: **px2rem**(159) **important**();  
}

错误

&[**lazy**=**loading**]{  
 **background**:**url**(..**/assets/images/icon/lazy\_load\_loading.png**) **scroll no-repeat center center important**();  
 **background-size**: **px2rem**(159) **important**();  
}

以上的暂时解决方法是涉及图片路径移到common.scss里面去。

Scss图片相对路径定位问题，比如\_footer.scss，反而正确

&:**after**{  
 **background**: **url**(**../images/icon/home\_on.png**) **no-repeat top center**;  
 **@include bg\_size\_px**(36);  
 **color**: $color\_55C1DD;  
}

目录结构\_extentd.scss图片路径是根据common.scss来相对定位的，因为comon.scss import了这个\_footer.scss



另外如果在index.vue中写上

<**style lang="scss" scoped**>  
 **@import '../../assets/scss/home/home.scss'//相对index.vue定位**</**style**>"

那么在\_extentd.scss的图片路径是相对这个index.vue来定位的，这样是找到图片

&[**lazy**=**loading**]{  
 **background**:**url**(**../../assets/images/icon/lazy\_load\_loading.png**) **scroll no-repeat center center important**();  
 **background-size**: **px2rem**(159) **important**();  
}

6、Vue使用<https://github.com/surmon-china/vue-awesome-swiper>，获取渲染后dom长度为0的问题。

<**swiper :options="swiperOption" id="cal" ref="mySwiperA"**>  
 <**swiper-slide class="slide bg\_lazyload" lazy="loading" v-for="banner in indexData.banner\_list"  
 v-bind:data-image="banner.image"**><**a href=""**></**a**></**swiper-slide**>  
 <**div class="swiper-pagination" slot="pagination"**></**div**>  
</**swiper**>

**swiperOption**: {  
 **autoplay**: 50000,  
 **pagination**: **'.swiper-pagination'**,  
 **notNextTick**: **true**,  
 onInit: **function (**swiper**)** {  
 **console**.log**(**swiper.slides.lenght**)//数据为2，但是这里是0** },  
 onSlideNextEnd: **function (**swiper**)** {  
 **console**.log**(**swiper**)**;  
 }  
}

mounted: **function ()** {  
 **this**.**$http**.get**('../../../static/api/home/homepageV3.json')**.then**((**response**)** => {  
 // indexData，然后html就可以遍历  
 console.log**(**response.**data)  
 this**.**indexData** = response.**data** }, **(**response**)** => {  
 **console**.log**(**response**)** }**)**}

解决方案：[vue生命周期来解释为什么vue-awesome-swiper插件中length=0](vue生命周期来解释为什么vue-awesome-swiper插件中length=0.docx)

7、底部跳页状态选中问题

首先在my.vue加上一个标志

<**input type="hidden" value="3" id="footerIcon"**>

Footer.vue使用

<**template**>  
 <**footer class="footer"**>  
 <**ul class="clearfix"**>  
 <**li**><**a href="index.html" data-font="首页"**></**a**> </**li**>  
 <**li**><**a href="" data-font="目的地" class="destination"**></**a**> </**li**>  
 <**li**><**a href="" data-font="购物车" class="cart"**></**a**></**li**>  
 <**li**><**a href="my.html" data-font="我的" class="my"**></**a**> </**li**>  
 </**ul**>  
 </**footer**>  
</**template**>  
<**script**>  
**import** { Vue, $ } **from 'js/base'  
export default** {  
 mounted: **function ()** {  
 **const** footerIcon = **document**.getElementById**('footerIcon')**.**value** $**('.footer ul li')**.eq**(**footerIcon**)**.children**('a')**.addClass**('on')** }  
}  
</**script**>  
<**style lang='scss' scoped**>  
 **@import "../../assets/scss/base/necessary.scss"**;  
</**style**>

功能实现了，但是这个好不vue咯，改变一种思路组件通讯

**父传值给子组件(使用props方式，my.vue包含common-footer)**

My.vue

<**template**>  
 <**common-footer footerIndex="1"**></**common-footer**>  
 </**div**>  
</**template**>  
<**script**>  
**import** CommonFooter **from '../../components/common/footer.vue'  
export default**{  
 data **()** {  
 **return** {  
 **indexData**: **[]** }  
 },  
 **components**: {  
 **'common-footer'**: CommonFooter  
 },  
 **methods**: {  
 }  
}  
</**script**>

My.js(common-footer放在这里的话就不是父子组件关系了)

**import** { Vue, $, Common } from **'js/base'  
import** myHeader from **'../../components/my/my-header.vue'  
console**.log**(**Common.**isLogin)***require***('./scss/my.scss')  
import** myCenter from **'./my.vue'  
var** homeVue = **new** Vue**(**{  
 **el**: **'#my'**,  
 **template**: **'<div class="pageview"><my-header></my-header><my></my></div>'**,  
 **components**: {  
 **'my-header'**: myHeader,  
 **'my'**: myCenter  
 }  
}**)**

Footer.vue

<**template**>  
 <**footer class="footer"**>  
 <**ul class="clearfix"**>  
 <**li**><**a href="index.html" data-font="首页"**></**a**> </**li**>  
 <**li**><**a href="" data-font="目的地" class="destination"**></**a**> </**li**>  
 <**li**><**a href="" data-font="购物车" class="cart"**></**a**></**li**>  
 <**li**><**a href="my.html" data-font="我的" class="my"**>{{**'footerIndex'**}}</**a**></**li**>  
 </**ul**>  
 </**footer**>  
</**template**>  
<**script**>  
**import** { Vue, $ } **from 'js/base'  
export default** {  
 **props**: **['footerIndex']**// 上面的**'footerIndex'**拿到是的my.vue的**'footerIndex'**值  
</**script**>

最终修改为

**export default** {  
 **props**: **['footerIndex']**,  
 mounted: **function ()** {  
 **let** self = **this  
 document**.querySelector**('footer')**.**children[**0**]**.**children[**self.footerIndex**]**.**children[**0**]**.**classList**.add**('on')** }  
}  
</**script**>

备注：父组件访问子组件的另一种方式ref

my.vue

<template>  
 <div class="container padding\_tb\_120 my">

<div>test</div>  
 <common-footer footerIndex="3" ref="child"></common-footer>  
 </div>  
</template>  
<script>  
**import** { Vue, $, Common } **from** 'js/base'  
**import** CommonFooter **from** '../../components/common/footer.vue'  
**export default**{  
 data () {  
 **return** {  
 counter: 0,  
 isLogin: Common.isLogin(),  
 loginImgSrc: [require('./images/logined.jpg'), require('./images/nologin.jpg')],  
 my: []  
 }  
 },  
 mounted: **function** () {  
 **if** (**this**.isLogin) {  
 **this**.$http.get('../../../static/api/center/getCenter.json').then((response) => {  
 // indexData，然后html就可以遍历  
 response.data['isLogin'] = Common.isLogin()  
 console.log(response.data)  
 **this**.my = response.data  
 // 触发getMessageCount事件，并且传递参数  
 **this**.$bus.emit('getMessageCount', { num: response.data.notification\_amount });

// 父组件my.vue访问子组件footer.vue的属性footerIndex  
 console.log(**this**.$refs.child.footerIndex)  
 }, (response) => {  
 console.log(response)  
 })  
 }  
 },  
 components: {  
 'common-footer': CommonFooter  
 }  
}  
</script>

8、使用img :src绑定图片，出现404问题

这样是对的

<div class="photo"><img src="./images/logined.jpg"></div>

然而用了

<div class="photo"><img v-bind:src="isLogin?'./images/logined.jpg':'./images/nologin.png'"></div>

报404找不到图片

应该是webpack找不到路径

解决方法

<div class="photo"><img v-bind:src="isLogin?loginImgSrc[0]:loginImgSrc[1]"></div>

data () {  
 **return** {  
 loginImgSrc: [require('./images/logined.jpg'), require('./images/nologin.jpg')]  
 }  
},

9、非父子组件之间的通讯（使用bus方式）

插件地址：https://github.com/yangmingshan/vue-bus

这里要实现的功能是：my.vue中ajax返回了一个notification\_amount=10的消息数量，我要把这个消息数量赋值给my-header.vue中的.my\_message这个data-count属性中。

my.js(my.vue返回的ajax数据要返回给my-header.vue,他们之间是非父子关系，因为<my-header>与<my>为并排)

**import** { Vue, $, Common } from 'js/base'  
**import** myHeader from '../../components/my/my-header.vue'  
require('./scss/my.scss')  
**import** myCenter from './my.vue'  
**var** homeVue = **new** Vue({  
 el: '#my',  
 template: '<div class="pageview"><my-header></my-header><my></my></div>',  
 components: {  
 'my-header': myHeader,  
 'my': myCenter  
 }  
})

my.vue（使用emit来触发）

<template>  
 <div class="container padding\_tb\_120 my">  
 <div class="summary margin\_b\_20 bg\_white">  
 <a class="flex flex\_v\_center info" href="#">  
 <div class="photo"><img v-bind:src="isLogin?loginImgSrc[0]:loginImgSrc[1]"></div>  
 <div class="flex\_item arrow\_right">  
 <h1 v-if="isLogin">{{my.nickname}}</h1>  
 <p v-if="isLogin">查看个人资料</p>  
 <p v-if="!isLogin" class="noLogin">未登录</p>  
 </div>  
 </a>  
 <ul class="flex bottom">  
 <li class="flex\_item"><a href="" :data-count="isLogin?my.coupon\_amount:0">优惠券</a></li>  
 <li class="flex\_item"><a href="" :data-count="isLogin?my.comment\_amount:0">我的点评</a></li>  
 </ul>  
 </div>  
 <ul class="flex order margin\_b\_20 bg\_white">  
 <li class="flex\_item">  
 <a href="" data-font="全部订单"><em></em></a>  
 </li>  
 <li class="flex\_item">  
 <a href="" data-font="待付款" class="paying"><em :data-count="isLogin?my.need\_pay\_amount:0"></em></a>  
 </li>  
 <li class="flex\_item">  
 <a href="" data-font="待评价" class="comment"><em :data-count="isLogin?my.need\_comment\_amount:0"></em></a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center">  
 <h1 class="flex\_item"><span class="tb\_icon" :data-count="isLogin?my.taobao\_order\_amount:0">淘宝订单</span></h1>  
 </a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center traveller">  
 <h1 class="flex\_item">常用出行人</h1>  
 </a>  
 </li>  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center hotel">  
 <h1 class="flex\_item">常住酒店</h1>  
 </a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center collect">  
 <h1 class="flex\_item">我的收藏</h1>  
 </a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center service">  
 <h1 class="flex\_item">客服电话</h1>  
 </a>  
 </li>  
 </ul>  
 <common-footer footerIndex="3"></common-footer>  
 </div>  
</template>  
<script>  
**import** { Vue, $, Common } **from** 'js/base'  
**import** CommonFooter **from** '../../components/common/footer.vue'  
**export default**{  
 data () {  
 **return** {  
 isLogin: Common.isLogin(),  
 loginImgSrc: [require('./images/logined.jpg'), require('./images/nologin.jpg')],  
 my: []  
 }  
 },  
 mounted: **function** () {  
 **if** (**this**.isLogin) {  
 **this**.$http.get('../../../static/api/center/getCenter.json').then((response) => {  
 // indexData，然后html就可以遍历  
 response.data['isLogin'] = Common.isLogin()  
 console.log(response.data)  
 **this**.my = response.data

// 触发'getMessageCount' **this**.$bus.emit('getMessageCount', { num: response.data.notification\_amount });  
 }, (response) => {  
 console.log(response)  
 })  
 }  
 },  
 components: {  
 'common-footer': CommonFooter  
 },  
 methods: {  
 }  
}  
</script>

然后在my-header.vue来监听这个getMessageCount这个事件

<template>  
 <header>  
 <a href="" class="header\_left my\_setting"></a>  
 <h1>{{title}}</h1>  
 <a href="" class="header\_right my\_message" data-count="0"></a>  
 </header>  
</template>  
  
<script>  
 **export default** {  
 data () {  
 **return** {  
 title: '我的'  
 }  
 },  
 mounted: **function** () { **this**.$bus.on('getMessageCount', **this**.setMessageCount);  
 },  
 methods: {  
 setMessageCount: **function** (params) {  
 document.querySelector('.my\_message').setAttribute('data-count', params.num)  
 }  
 }  
 }  
</script>

总结my-header.vue获取my.vue数据

第一：base.js添加作为全局搭桥

**import** VueBus from 'vue-bus'  
Vue.use(VueBus)

第二：my.vue使用$bus.emit来触发事件，并且传递参数

第三：my-header.vue使用$bus.on来监听事件，获取emit传过来的参数

10、子组件传递数据给父组件，demo

参考：http://www.cnblogs.com/QRL909109/p/6166209.html

流程就是：

1、footer.vue使用$emit传入触发一个自定义事件'getMessageNumber'，并且把1作为参数

2、my.vue使用@getMessageNumber来监听，并且调用了test方法

footer.vue

<template>  
 <footer class="footer">  
 <ul class="clearfix">  
 <li><a href="index.html" data-font="首页"></a> </li>  
 <li><a href="" data-font="目的地" class="destination"></a> </li>  
 <li><a href="" data-font="购物车" class="cart" data-count="0"></a></li>  
 <li><a href="my.html" data-font="我的" class="my"></a></li>  
 </ul>  
 </footer>  
</template>  
<script>  
**import** { Vue, $, Common } **from** 'js/base'  
**export default** {  
 props: ['footerIndex'],  
 mounted: **function** () {  
 **let** self = **this** document.querySelector('footer').children[0].children[self.footerIndex].children[0].classList.add('on')  
 **if** (Common.isLogin()) {  
 **this**.$http.get('../../../static/api/order/countShopCartNum.json').then((response) => {  
 console.log(response.data)  
 document.querySelector('.cart').setAttribute('data-count', response.data.shop\_cart\_num)  
 **this**.$emit('getMessageNumber', 1);  
 }, (response) => {  
 console.log(response)  
 })  
 }  
 }  
}  
</script>

my.vue

<template>  
 <div class="container padding\_tb\_120 my">  
 <div class="summary margin\_b\_20 bg\_white">  
 <a class="flex flex\_v\_center info" href="#">  
 <div class="photo"><img v-bind:src="isLogin?loginImgSrc[0]:loginImgSrc[1]"></div>  
 <div class="flex\_item arrow\_right">  
 <h1 v-if="isLogin">{{my.nickname}}</h1>  
 <p v-if="isLogin">查看个人资料</p>  
 <p v-if="!isLogin" class="noLogin">未登录</p>  
 </div>  
 </a>  
 <ul class="flex bottom">  
 <li class="flex\_item"><a href="" :data-count="isLogin?my.coupon\_amount:0">优惠券</a></li>  
 <li class="flex\_item"><a href="" :data-count="isLogin?my.comment\_amount:0">我的点评</a></li>  
 </ul>  
 </div>  
 <ul class="flex order margin\_b\_20 bg\_white">  
 <li class="flex\_item">  
 <a href="" data-font="全部订单"><em></em></a>  
 </li>  
 <li class="flex\_item">  
 <a href="" data-font="待付款" class="paying"><em :data-count="isLogin?my.need\_pay\_amount:0"></em></a>  
 </li>  
 <li class="flex\_item">  
 <a href="" data-font="待评价" class="comment"><em :data-count="isLogin?my.need\_comment\_amount:0"></em></a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center">  
 <h1 class="flex\_item"><span class="tb\_icon" :data-count="isLogin?my.taobao\_order\_amount:0">淘宝订单</span></h1>  
 </a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center traveller">  
 <h1 class="flex\_item">常用出行人</h1>  
 </a>  
 </li>  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center hotel">  
 <h1 class="flex\_item">常住酒店</h1>  
 </a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center collect">  
 <h1 class="flex\_item">我的收藏</h1>  
 </a>  
 </li>  
 </ul>  
 <ul class="item margin\_b\_20 bg\_white">  
 <li>  
 <a href="" class="arrow\_right flex flex\_v\_center service">  
 <h1 class="flex\_item">客服电话</h1>  
 </a>  
 </li>  
 </ul>  
 <common-footer footerIndex="3" @getMessageNumber="test"></common-footer>  
 </div>  
</template>  
<script>  
**import** { Vue, $, Common } **from** 'js/base'  
**import** CommonFooter **from** '../../components/common/footer.vue'  
**export default**{  
 data () {  
 **return** {  
 isLogin: Common.isLogin(),  
 loginImgSrc: [require('./images/logined.jpg'), require('./images/nologin.jpg')],  
 my: []  
 }  
 },  
 mounted: **function** () {  
 **if** (**this**.isLogin) {  
 **this**.$http.get('../../../static/api/center/getCenter.json').then((response) => {  
 // indexData，然后html就可以遍历  
 response.data['isLogin'] = Common.isLogin()  
 console.log(response.data)  
 **this**.my = response.data  
 }, (response) => {  
 console.log(response)  
 })  
 }  
 },  
 components: {  
 'common-footer': CommonFooter  
 },  
 methods: {  
 test: **function** (s) {  
 console.log(s)  
 }  
 }  
}  
</script>

总结：组件之间通讯方式

1. 父子组件之间的通讯

* 父传子

1、props

2、ref

* 子传父

1、emit，on结合

1. 非父子之间通讯

1、bus

2、A，B为兄弟组件，且他们有共同的父组件

A调父组件，父组件传给B组件（未实践）

3、vuex（未实践）

11、使用全局filters来过滤数据

建立一个文件filters.js

**const** filters = {  
 toFriendlyTime: **function (**b, timestr**)** {  
 **console**.log**(**b**)  
 console**.log**(**timestr**)  
 return** timestr + b  
 }  
}  
// 使用：Common.isLogin  
module.exports = filters

list.vue

<**template**>  
 <**div class="container padding\_t\_122 messageList"**>  
 <**ul class="bg\_white"**>  
 <**li v-for="list in listData"**>  
 <**a href=""**>  
 <**h1 class="top flex flex\_v\_center"**>  
 <**div class="title flex\_item"**><**span data-count="1"**>{{list.title}}</**span**></**div**>  
 <**div class="time"**><**time**>{{list.create\_time | toFriendlyTime('b',list.create\_time)}}</**time**></**div**>  
 </**h1**>  
 <**p class="detail"**>{{list.content}}</**p**>  
 </**a**>  
 </**li**>  
 </**ul**>  
 <**mugen-scroll :handler="fetchData" :should-handle="!loading"**>  
 <**list-bottom**></**list-bottom**>  
 </**mugen-scroll**>  
 </**div**>  
</**template**>  
<**script**>  
**import** MugenScroll **from 'vue-mugen-scroll'  
import** listBottom **from '../../components/common/list-bottom.vue'  
import** filters **from '../../assets/js/filters'  
export default**{  
 data **()** {  
 **return** {  
 **listData**: **[]**,  
 **loading**: **false**,  
 **count**: 1  
 }  
 },  
 **components**: {  
 MugenScroll,  
 listBottom  
 },  
 **filters**: {  
 **toFriendlyTime**: filters.**toFriendlyTime // 注册filters** },  
 methods: {  
 fetchData **()** {  
 **if (**!**this**.**loading)** {  
 **this**.**loading** = **true**;  
 **this**.**$http**.get**('../../../static/api/center/getNotification.json')**.then**((**response**)** => {  
 console.log**(**response.**data)** setTimeout**(()** => {  
 **for (let** i = 0; i < response.**data**.notification\_list.length; i++**)** {  
 **this**.**listData**.push**(**response.**data**.notification\_list**[**i**])** }  
 // 不明白为什么要提到外面  
 **this**.**loading** = **false** }, 5000**)** }, **(**response**)** => {  
 **console**.log**(**response**)** }**)** }  
 }  
 }  
}  
</**script**>

注意，如果传一个参数可以省略写

<**div class="time"**><**time**>{{list.create\_time | toFriendlyTime}}</**time**></**div**>

如果是多个参数，而又想把v-for里面的数据传递(字符串放在最前面list. create\_time最好放在最后)

<**div class="time"**><**time**>{{list.create\_time | toFriendlyTime('b', list.create\_time)}}</**time**></**div**>

https://cn.vuejs.org/v2/guide/syntax.html#Filters

12、设置textContent后不更新文字问题

List.vue

<**template**>  
 <**div class="container padding\_t\_122 messageList"**>  
 <**ul class="bg\_white"**>  
 <**li v-for="list in listData"**>  
 <**a href=""**>  
 <**h1 class="top flex flex\_v\_center"**>  
 <**div class="title flex\_item"**><**span data-count="1"**>{{list.title}}</**span**></**div**>  
 <**div class="time"**><**time**>{{list.create\_time | toFriendlyTime}}</**time**></**div**>  
 </**h1**>  
 <**p class="detail"**>{{list.content}}</**p**>  
 </**a**>  
 </**li**>  
 </**ul**>  
 <**mugen-scroll :handler="fetchData" :should-handle="!loading"**>  
 <**list-bottom ref="listBottom"**></**list-bottom**>  
 </**mugen-scroll**>  
 </**div**>  
</**template**>  
<**script**>  
**import** { Vue, $, Common } **from 'js/base'  
import** MugenScroll **from 'vue-mugen-scroll'  
import** listBottom **from '../../components/common/list-bottom.vue'  
import** filters **from '../../assets/js/filters'  
export default**{  
 data **()** {  
 **return** {  
 **listData**: **[]**,  
 **loading**: **false**,  
 **currentPageCount**: 1,  
 **pageCount**: 0  
 }  
 },  
 **components**: {  
 MugenScroll,  
 listBottom  
 },  
 **filters**: {  
 **toFriendlyTime**: filters.**toFriendlyTime**// 注册filters  
 },  
 methods: {  
 fetchData **()** {  
 **if (**!**this**.**loading)** {  
 **this**.**loading** = **true**;  
 **if (this**.**currentPageCount** > **this**.**pageCount** && **this**.**pageCount** !== 0**)** {  
 **this**.**loading** = **true**

**// 然后上面的文字并没有改过来  
 this**.**$refs**.listBottom.**$el**.**children**.**textContent** = **'没有更多数据了'**  
 **return false** }  
 **this**.**$http**.get**('../../../static/api/center/getNotification.json')**.then**((**response**)** => {  
 console.log**(**response.**data)** setTimeout**(()** => {  
 **for (let** i = 0; i < response.**data**.notification\_list.length; i++**)** {  
 **this**.**listData**.push**(**response.**data**.notification\_list**[**i**])** }  
 **this**.**loading** = **false** // 不明白为什么要提到外面,不提到这里会不断请求不止  
 **this**.**pageCount** = Common.index2PageCount**(**response.**data**.total\_index**)  
 this**.**currentPageCount**++  
 }, 3000**)** }, **(**response**)** => {  
 **console**.log**(**response**)** }**)** }  
 }  
 }  
}  
</**script**>

list-bottom.vue

<**template**>  
 <**div class="list\_bottom"**><**span class="text"**>{{text}}</**span**></**div**>  
</**template**>  
<**script**>  
**export default** {  
 data **()** {  
 **return** {  
 **text**: **'数据加载中...'** }  
 }  
}  
</**script**>

低级错误

解决方法：

**this**.**$refs**.listBottom.**$el**.**children[**0**]**.**textContent** = **'没有更多数据了'**

13、使用$set来添加到data注意的地方

<**template**>  
 <**header class="center\_header"**>  
 <**a href="" class="header\_left my\_setting"**></**a**>  
 <**h1**>{{title}}</**h1**>  
 <**a href="messageList.html" class="header\_right my\_message" data-count="0"**></**a**>  
 </**header**>  
</**template**>  
  
<**script**>  
 **export default** {  
 data **()** {  
 **return** {  
 **title**: **'我的'**,  
 **'headerInfo'**: [] // 一定要定义，即使是空的也要  
 }  
 },  
 mounted: **function ()** {  
 **this**.**$bus**.on**('getMessageCount'**, **this**.setMessageCount**)**;  
 **this**.**$bus**.on**('setCenterHeader'**, **this**.setCenterHeader**)**;  
 },  
 **methods**: {  
 setMessageCount: **function (**params**)** {  
 **document**.querySelector**('.my\_message')**.setAttribute**('data-count'**, params.**num)** },  
 setCenterHeader: **function (**params**)** {

//data要定义**'headerInfo'**否则报错  
 **this**.$set**(this**.$data, **'headerInfo'**, params**)** // document.querySelector('.header\_left')  
 }  
 }  
 }  
</**script**>

14、针对13中问题使用$set方法动态添加data,vue无法对2层以上如(a:{b:{c:1}})的object数据在template进行{{}}输出

List.vue

**export default**{  
 data **()** {  
 **return** {  
 **listData**: **[]**,  
 **loading**: **false**,  
 **currentPageCount**: 1,  
 **pageCount**: 0  
 }  
 },  
 mounted: **function ()** {  
 **this**.**$bus**.**emit('setCenterHeader'**, {**title**: **'消息中心'**, **left**: {**className**: **'back'**, **href**: **''**}, **'right'**: {**hide**: **true**}}**)** }  
}  
</**script**>

my-header

<**template**>  
 <**header class="center\_header"**>  
 <**a href="" :class="{'header\_left my\_setting':headerInfo.length==0,  
 'header\_left back':headerInfo.length!=0,  
 }"**></**a**>  
 <**h1**>{{title}}</**h1**>  
 <**a href="messageList.html" :class="{'header\_right my\_message':headerInfo.length==0,  
 'header\_right':headerInfo.length!=0&&headerInfo.right.hide==true,  
 }" data-count="0"**></**a**>  
 {{headerInfo['right']['hide']}}// 报错，而{{headerInfo.right}}可以获取{hide:true}  
 </**header**>  
</**template**>  
  
<**script**>  
 **export default** {  
 data **()** {  
 **return** {  
 **title**: **'我的'**,  
 **headerInfo**: **[]** }  
 },  
 mounted: **function ()** {  
 **this**.**$bus**.on**('getMessageCount'**, **this**.setMessageCount**)**;  
 **this**.**$bus**.on**('setCenterHeader'**, **this**.setCenterHeader**)**;  
 },  
 **methods**: {  
 setMessageCount: **function (**params**)** {  
 **document**.querySelector**('.my\_message')**.setAttribute**('data-count'**, params.**num)** },  
 setCenterHeader: **function (**params**)** {  
 **this**.$set**(this**.$data, **'title'**, params.**title)  
 this**.$set**(this**.$data, **'headerInfo'**, params**)**  
 }  
 }  
 }  
</**script**>

不知道怎么解决，但是用了另外一种解决方法去解决：就是:class使用computed过滤

<template>  
 <header class="center\_header">  
 <a href="" :class="setLeftHeader"></a>  
 <h1>{{title}}</h1>  
 <a href="messageList.html" :class="setRightHeader" data-count="0"></a>  
 </header>  
</template>  
  
<script>  
 **export default** {  
 data () {  
 **return** {  
 title: '我的',  
 headerInfo: []  
 }  
 },  
 mounted: **function** () {  
 **this**.$bus.on('getMessageCount', **this**.setMessageCount);  
 **this**.$bus.on('setCenterHeader', **this**.setCenterHeader);  
 },  
 computed: {  
 setLeftHeader: **function** () {  
 **return** {  
 'header\_left my\_setting': **this**.headerInfo.length === 0,  
 'header\_left back': **this**.headerInfo.length !== 0  
 }  
 },  
 setRightHeader: **function** () {  
 **return** {  
 'header\_right my\_message': **this**.headerInfo.length === 0,  
 'header\_right hide': **this**.headerInfo.length !== 0 && **this**.headerInfo.right.hide === **true // 这里三层是可以获取到的** }  
 }  
 },  
 methods: {  
 setMessageCount: **function** (params) {  
 document.querySelector('.my\_message').setAttribute('data-count', params.num)  
 },  
 setCenterHeader: **function** (params) {  
 **this**.$set(**this**.$data, 'title', params.title)// 改变数据并且更新视图  
 **this**.$set(**this**.$data, 'headerInfo', params)  
 }  
 }  
 }  
</script>

15、@获取当前点击元素

<a href="javascript:void(0)" :class="setLeftHeader" @click="leftHeader"></a>

methods: {  
 leftHeader: **function** (e) {  
 console.log(e.target.className)  
 }

}

16、问题1：.js文件引入.scss文件没有自动添加浏览器前缀问题；问题2、npm run dev 和npm run build 编译后 浏览器前缀解析（丢失部分前缀）不一样问题

https://github.com/vuejs/vue-loader/issues/424

utils.js

解决问题1

exports.styleLoaders = **function (**options**)** {  
 **var** output = **[]**;  
 **var** loaders = exports.cssLoaders**(**options**)**;  
  
 **for (var** extension **in** loaders**)** {  
 // var loader = loaders[extension];  
 // 解决.js文件引入scss无法添加前缀问题  
 **var** loader = loaders**[**extension**]**.split**('!')**;  
 // 解决.js文件引入scss无法添加前缀问题  
 **var** isPreProcesser = **['less'**, **'sass'**, **'scss'** ,**'stylus'**, **'styl']**.some**(function (**v**)** {  
 **return** v === extension  
 }**)** // 解决.js文件引入scss无法添加前缀问题  
 **if (**isPreProcesser**)** {  
 loader.splice**(**-1, 0, **'postcss-loader')** }  
 output.push**(**{  
 test: **new** RegExp**('\\.'** + extension + **'$')**,  
 // loader: loader  
 // 解决.js文件引入scss无法添加前缀问题  
 loader: loader.join**('!')** }**)** }  
  
 **return** output  
}

解决问题2

exports.cssLoaders = **function (**options**)** {  
……  
 **function** *generateLoaders* **(**loaders**)** {  
 **var** sourceLoader = loaders.**map(function (**loader**)** {  
 **var** extraParamChar;  
 **if (**/\?/.test**(**loader**))** {  
 loader = loader.**replace(**/\?/, **'-loader?')** extraParamChar = **'&'** } **else** {  
 loader = loader + **'-loader'** extraParamChar = **'?'** // 解决npm run dev 和 npm run build 编译后前缀不一样的问题  
 **if (**loader === **'css-loader')** {  
 extraParamChar = **'?-autoprefixer&'** }  
 }

……  
}

webpack.base.conf.js

**...**

**var** autoprefixerConf = autoprefixer**(**{ **browsers**: **['last 2 versions'**,**'Android >= 4.0'**,**'iOS >= 6']** }**)**;  
**...**  
module.exports = {  
 **entry**: entries,  
 **module**: {

**...** },  
 // js 中引入的样式处理  
 **postcss**: **[**autoprefixerConf**]**,}

17、使用VUEX

[消息列表使用vuex来获取数据工作流程](消息列表使用vuex来获取数据工作流程.docx)

参考：<https://github.com/ChuckCZC/vue-demo-maizuo>

18、VUEX中对返回数据如果使用setTimeout来进行延迟加载，那么有可能导致数据更新不能同步到视图上（但是你滚屏一下视图就更新数据）

fetchData () {  
 **if** (!**this**.loading) {  
 **this**.loading = **true**;  
 **if** (!**this**.hasMore) {  
 **this**.finishAction()  
 **return false** }  
 **this**.$store.dispatch('getMessageList')  
 setTimeout(() => {  
 **this**.loading = **false//数据最新，但是视图不更新**  
 }, 3000)  
 }  
},

这个是为了模拟网络慢的时候加载，现在暂时将时间设置为0，网络慢模拟用chrome模拟器进行设置。

19、在点击确定登录ACTIONS（涉及调用异步接口返回），怎么去获取异步返回后做接下来的任务呢？

login.vue

loginSubmit: **function** () {  
 **this**.submitDisabled = **true  
 this**.$store.dispatch('goLogin', {'accout': **this**.account, 'password': **this**.password}).then((res) => {  
 console.log(res)  
 })  
},

vuex/module/login.js

**const** actions = {  
 goLogin: **function** ({commit}, params) {  
 **return new** Promise((resolve, reject) => {  
 api.getLogin(params, **function** (res) {  
 commit(types.LOGIN\_IS\_SUCCESS, res);  
 resolve(res)  
 })  
 })  
 }  
}

api.js

getLogin: **function** (params, cb) {  
 axios.get(url + 'user/login.json?t=' + **new** Date() \* 1 + '&callback=?').then(**function** (res) {  
 **if** (res.status >= 200 && res.status < 300) {  
 cb(res.data)  
 }  
 }).catch((error) => {  
 **return** Promise.reject(error)  
 })  
}

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

20、由于多页面的html文件引入了相同的第三方资源，有些是CDN资源，有些是项目中的资源，能不能有一种方法，可以include这里JS文件，免得到时候修改一个资源，要所有的html都要改，**待解决**。

https://github.com/jso0/html-webpack-cdn-plugin

21、数据驱动视图，数据驱动视图，数据驱动视图，毕竟是为了提高工作效率的，能用则用，最近在用axios代替vue-resource，并且使用VUEX发现了一个问题，就是jquery的思想没有改变过来，获取购物车数量这个功能

<**li**><**a href="" data-font="购物车" class="cart" data-count="0"**></**a**></**li**>

以前一般是这样数据的，ajax返回数据，然后使用jquery获取那个dom，再把属性为ajax返回的数据

mounted: **function ()** {

**if (**Common.**isLogin())** {

**this**.**$store**.dispatch**('getShoppingCartNum')**

document.querySelector('.cart').setAttribute('data-count', this.$store.getters.getShoppingCartNum)

}

},

首先，虽然state虽然改为ajax返回的购物车数量，但是这个操作并没有使得data-count的值发生改变，还是默认的值（貌似只改变视图的值，而不是JS的）

方法，在action中ajax使用promise返回，然后再使用下面的，来这个异步返回来，接下来的操作

那么，this.$store.dispatch('getShoppingCartNum').then(….)

其实上面的可以这样改写：

<**li**><**a href="" data-font="购物车" class="cart" :data-count="getShoppingCartNum"**></**a**></**li**>

mounted: **function ()** {

**let** self = **this**

**document**.querySelector**('footer')**.**children[**0**]**.**children[**self.**footerIndex]**.**children[**0**]**.**classList**.add**('on')**

**if (**Common.**isLogin())** {

**this**.**$store**.dispatch**('getShoppingCartNum')**

}

},

**const** actions = {

getShoppingCartNum **(**{commit}**)** {

api.getShoppingCartNum**(function (**res**)** {

commit**(**types.SHOPPING\_GET\_NUM, res**)**;

}**)**

}

}

之后改变的值，会同步更新到view上

22、多个action都是异步，存在依赖关系，怎么去解决呢？虽然也可适应promise简化代码，**但是强烈建议使用Async/await，很好用**

Async/await报错的错就要安装babel-preset-stage-3

My.vue

**(**async **()** => {

**let** result = await **this**.$store.dispatch**('gerCenterData')**

**this**.**$store**.dispatch**('getMessageCount'**, result.notification\_amount**)**

}**)()**

My.js

**const** actions = {

gerCenterData: **function (**{commit}**)** {

**return** api.getCenterData**(function (**res**)** {

commit**(**types.CENTER\_GET\_DATA, res**)**

}**)**

}

}

api.js

getCenterData: **function (**cb**)** {

**return new** Promise**((**resolve, reject**)** => {

axios.get**(**url + **'center/getCenter.json?t='** + **new** Date**()** \* 1 + **'&callback=?')**.then**(function (**res**)** {

**if (**res.**status** >= 200 && res.**status** < 300**)** {

cb**(**res.**data)**

resolve**(**res.**data)**

}

}**)**.catch**((***error***)** => {

reject**(**error**)**

}**)**

}**)**

}

参考：

<http://www.tuicool.com/articles/ZZnuQzZ>

23、VUEX如果使用了获取数据，更新到视图上

...mapGetters**(**{

**indexData**: **'getIndexData'**

}**)**

那么下面不要在定义，否者报错

data **()** {

**return** {

**indexData**: **[]**

}

},

24、build后js插入到html <script>标签依赖顺序乱掉(dev环境正常)，由于生成出来的index.js是依赖vendor.js（里面包含vue,vuex等一些集合），所以运行会报Cannot read property 'call' of undefined的错误

<script type="text/javascript" src="/static/js/index.0e6eb0ee5806b34f2ff9.js?1e099539e95fb0408b89"></script>  
<script type="text/javascript" src="/static/js/vendor.6e68694550e415f910a2.js?1e099539e95fb0408b89"></script>

方法一：webpack.prod.conf.js注释掉new webpack.optimize.OccurenceOrderPlugin()

参考：<https://segmentfault.com/q/1010000007218738>

方法二：webpack.base.conf.js，添加chunksSortMode: 'dependency'

**for (var** pathname **in** pages**)** {  
 // 生成html相关配置  
 **var** conf = {  
 **filename**: pathname + **'.html'**, // html文件输出路径  
 **template**: pages**[**pathname**]**, // 模板路径  
 **inject**: **true**, // js插入位置  
 **chunksSortMode**: **'dependency'** };  
 pathname=pathname.split**("/")[**1**]**;//去掉views  
 **if (**pathname **in** module.exports.entry**)** {  
 conf.inject = **'body'**;  
 //如果每个html没有进入这里的话，那么全部js将会插入html  
 conf.chunks = **[**pathname, **'vendor'**, **'manifest'**, **'bootstrap']**,  
 conf.hash = **true**;  
 }  
 module.exports.plugins.push**(new** HtmlWebpackPlugin**(**conf**))**;

参考：

<http://stackoverflow.com/questions/35725384/webpackhtmlplugin-control-the-order-of-injected-files>

25、多页面定义重复名字的getters问题，那么获取getters只能获取一个

modules/messageList.js定义了

**const** getters = {  
 getMessageListGet: state => state.listData,  
 hasMore: state => state.hasMore  
}

而modules/collect.js也定义了一个相同的名字

**const** getters = {  
 getCollectList: state => state.collectList,  
 hasMore: state => state.hasMore  
}

vuex会报这个错误

[vuex] duplicate getter key: hasMore

在collect.vue

computed: {  
 ...mapState({  
 list: (state, getters) => {  
 **return** getters  
 }  
 })  
},

只返回一个hasMore,这个值还是在store.js引入模块靠前的那个

**export default new** Vuex.Store({  
 modules: {  
 messageList,  
 collect  
 },  
 strict: process.env.NODE\_ENV !== 'production'   
})

所以，如果公用的话要注意一下

26、v-show取值来源为data，之后state更新不触发视图更新

<**div class="checkbox\_wrap" v-show="checkboxShow"**>

data **()** {  
 **return** {  
 **loading**: **false**,  
 **first**: **true**,  
 **checkboxShow**: **this**.**$store**.**state**.**collect**.**showCheckbox** }  
},

**const** state = {  
 showCheckbox: **false**  
}

**const** mutations = {  
 **[**types.COLLECT\_SHOW\_CHECKBOX**] (**state**)** {  
 **if (**state.showCheckbox**)** {  
 state.showCheckbox = **false** } **else** {  
 state.showCheckbox = **true** }}  
}

**const** getters = {  
 getCheckboxShow: state => state.**showCheckbox**}

功能为点击一个按钮，实现隐藏显示的判断，如果视图上初始化绑定的数据来源是data的话，那么只有对这个data的更新（state中showCheckbox）不触发（也许data初始化之后不再触发视图）视图更新。

方法：

去掉data()中的checkboxShow，改为数据来源如下

...mapGetters({  
 checkboxShow: 'getCheckboxShow'  
})

或者

...mapState**(**{  
 checkboxShow: state => state.**collect**.**showCheckbox**  
}**)**

关于data()中数据触发视图更新可以使用$set或变异方法

一些技术难点参考项目（先记录）：

1、详情返回列表，固定位置

https://github.com/lzxb/vue-cnode?from=xitu