~~WebStorm~~ IDEA+Vue前端开发学习笔记

目录

[一、 环境配置（win10） 2](#_Toc523731288)

[1、vue.js 三种方式安装 - CSDN博客（这篇博客很棒） 2](#_Toc523731289)

[2、解决：vue不是内部或外部命令 2](#_Toc523731290)

[3、webstorm 2018 激活破解方法大全 - CSDN博客 3](#_Toc523731291)

[4、IDEA安装vue插件，进行vue开发 3](#_Toc523731292)

[二、 学习vue基本知识（这些主要是引用CDN） 3](#_Toc523731293)

[1、 vue CDN 4](#_Toc523731294)

[2、 {{ }}，属性提取 4](#_Toc523731295)

[3、vue 属性绑定 v-bind , v-html 4](#_Toc523731296)

[1-3的源码： 4](#_Toc523731297)

[5、Vue-事件修饰符 6](#_Toc523731298)

[6、css作用域<style scoped> 6](#_Toc523731299)

[7、 6](#_Toc523731300)

[三、 开发一个完整的项目（由cli脚手架构建） 6](#_Toc523731301)

[1、创建一个完整项目的命令 6](#_Toc523731302)

[（1）创建一个名为firstApp项目 6](#_Toc523731303)

[（2）目录结构： 7](#_Toc523731304)

[（3）安装依赖（非常重要） 8](#_Toc523731305)

[（4）**运行项目** 10](#_Toc523731306)

[3、 在Vue-cli中引入Element Ui 11](#_Toc523731307)

[4、Element-UI 自定义样式 12](#_Toc523731308)

[5、 配置路由（实现页面跳转） 12](#_Toc523731309)

[6、在 Vue-cli 中引入 Semantic UI（该框架与vue不太兼容，建议不要用） 15](#_Toc523731310)

[7、通过axios来发送请求（axios的安装，引用） 17](#_Toc523731311)

[8、关于axios的一系列问题！！！（踩坑！） 18](#_Toc523731312)

[8.1、前后端交互的跨域问题 19](#_Toc523731313)

[8.2、axios 发送post请求时，参数丢失问题 21](#_Toc523731314)

[8.3、axios 问题集锦 22](#_Toc523731315)

[9、登录时保存信息，SessionStorage和Vuex的区别 23](#_Toc523731316)

[（1）SessionStorage 的使用方法 23](#_Toc523731317)

[(2)Vuex的使用方法（中大型项目推荐使用） 23](#_Toc523731318)

[10、vue项目改名怎么办？ 24](#_Toc523731319)

[11、生命周期（钩子函数）？？？ 24](#_Toc523731320)

[Vue组件钩子函数 24](#_Toc523731321)

[四、前端页面功能实现 25](#_Toc523731322)

[1、element 的layout布局（bootstrap栅格系统） 25](#_Toc523731323)

[2、CSS Position(定位) | 菜鸟教程 25](#_Toc523731324)

[3、导航栏，侧边栏的继承（组件的嵌套）（即局部注册组件） 26](#_Toc523731325)

[4、页面的跳转 28](#_Toc523731326)

[5、页面加载成功时，触发事件 28](#_Toc523731327)

[6、vue 加载远程图片（ip+端口+url） 28](#_Toc523731328)

[7、element-ui upload组件+springBoot 实现图片的上传 28](#_Toc523731329)

[7.1后端代码 29](#_Toc523731330)

[7.2、前端代码 30](#_Toc523731331)

[7.3、发现问题：数据更新，但组件没有更新 31](#_Toc523731332)

[8、upload组件（2）上传文件，携带额外参数，文件类型限制，文件大小限制 32](#_Toc523731333)

[8.1、后端代码 32](#_Toc523731334)

[8.2、前端代码（大体上是官网上的例子） 35](#_Toc523731335)

[9、视频播放器 vue-video-player 39](#_Toc523731336)

[9.1、安装依赖 39](#_Toc523731337)

[9.2、引入样式 39](#_Toc523731338)

[9.3、把VueVideoPlayer导入并挂在到vue上 40](#_Toc523731339)

[9.4、编写业务组件videoPlayer.vue 40](#_Toc523731340)

[10、列表的渲染 42](#_Toc523731341)

[11、表格的增删改查 42](#_Toc523731342)

[5、数组遍历 42](#_Toc523731343)

导言：本人之前一直在做后端，现在为了。。，决定学习前端。

本人前端一点基础都没有，什么js,html,css等都没学过，只是知道大概怎么用。本系列本人将从vue入手，学习前端。先学会怎么用，然后学习js,html,css夯实基础，理解其原理。最终目标是内外兼修，融汇贯通。

1. 环境配置（win10）

## 1、vue.js 三种方式安装 - CSDN博客（这篇博客很棒）

<https://blog.csdn.net/m0_37479246/article/details/78836686>

我这里采用了npm 安装（方法在上面链接中）

## 2、解决：vue不是内部或外部命令

vue不是内部或外部命令，配置一个Path系统变量就可以解决 - 沐木琴 - 博客园

<https://www.cnblogs.com/cindy79/p/7082281.html>

<https://segmentfault.com/q/1010000007804677/a-1020000007824598>

## 3、webstorm 2018 激活破解方法大全 - CSDN博客

<https://blog.csdn.net/voke_/article/details/76418116>

## 4、IDEA安装vue插件，进行vue开发

偶然间发现idea可以开发vue

当然选择她了

<https://www.jianshu.com/p/01d832e028ea>

但我安装失败了

网上找了许多方法，都没有

虽然无法安装插件，但只要你知道.vue的模板文件不影响，可以将html强行改成vue文件

举个例子：

test.vue

|  |
| --- |
| <template>  </template>  <script>   **export default** {  name: "test"  } </script>  <style scoped>  </style> |

1. 学习vue基本知识（这些主要是引用CDN）

Vue2.0小白入门教程(最全) - 网易云课堂

<https://study.163.com/course/introduction.htm?courseId=1004711010>

这篇教程视频还不错。边看边做。

vue.js框架是渐进式，所谓渐进式的一个特点是：可以使用其他优秀的前端框架

1. vue CDN

安装 — Vue.js

<https://cn.vuejs.org/v2/guide/installation.html#CDN>

1. {{ }}，属性提取

## 3、vue 属性绑定 v-bind , v-html

<a v-bind:href="website">web链接</a>

<input type="text" v-bind:value="name">

绑定标签：<p v-html="websiteTag"></p>

## 1-3的源码：

index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Vue.js</title>  <link rel="stylesheet" href="style.css">  <!-- 开发环境版本，包含了有帮助的命令行警告 -->  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>  </head>  <body>  <!-- vue-app是根容器 -->  <div id="vue-app">  <h1>{{ greet() }}</h1>  <h1>{{ name }}</h1>  <a v-bind:href="website">web开发</a>  <input type="text" v-bind:value="name">  <p v-html="websiteTag"></p>  </div>  <script src="app.js"></script>  </body>  </html> |

app.js

|  |
| --- |
| //实例化vue对象  new Vue({  el:"#vue-app",  data:{  name:"拉拉啦",  website:"http://www.baidu.com",  websiteTag:"<a>ddddddd</a>"  },  methods:{  greet: function() {  return 'Good Morning!'+" "+this.name;  }  }  })  /\*\*  \* el:element 需要获取的元素，一定是html中的跟容器元素  \* data:用于数据的存储  \* methods:用于存储各种方法  \*/ |

4、vue-事件（点击，双击，鼠标事件）

@click 与v-on:click效果是一样的

<div id="vue-app">

<button v-on:click="age++">长一岁</button>

<button v-on:click="age--">减一岁</button>

<br>

========================================

<button @click="add">长一岁</button>

<button v-on:click="subtract">减一岁</button>

<br>

========================================

<button @dblclick="add">长十岁</button>

<button v-on:dblclick="subtract">减岁</button>

<br>

<h1>my age is {{ age }}</h1>

<br>

<div id="canvas" v-on:mousemove="updateXY">

{{x}},{{y}}

</div>

</div>

## 5、Vue-事件修饰符

## 6、css作用域<style scoped>

scoped 给class加一个唯一标识，样式只适用当前组件

## 7、

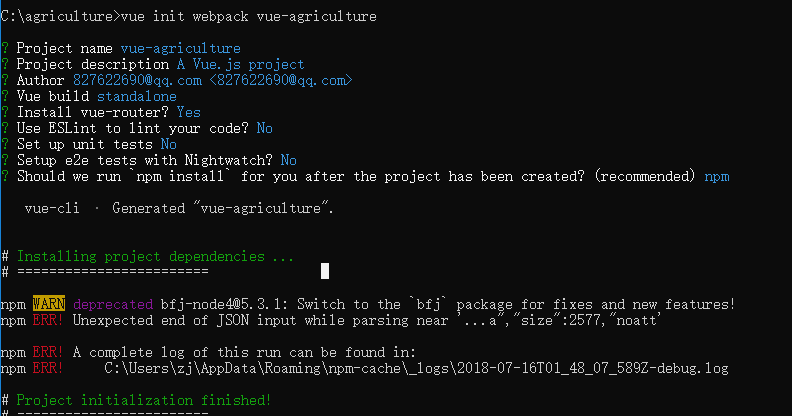
1. 开发一个完整的项目（由cli脚手架构建）

## 1、创建一个完整项目的命令

### （1）创建一个名为firstApp项目

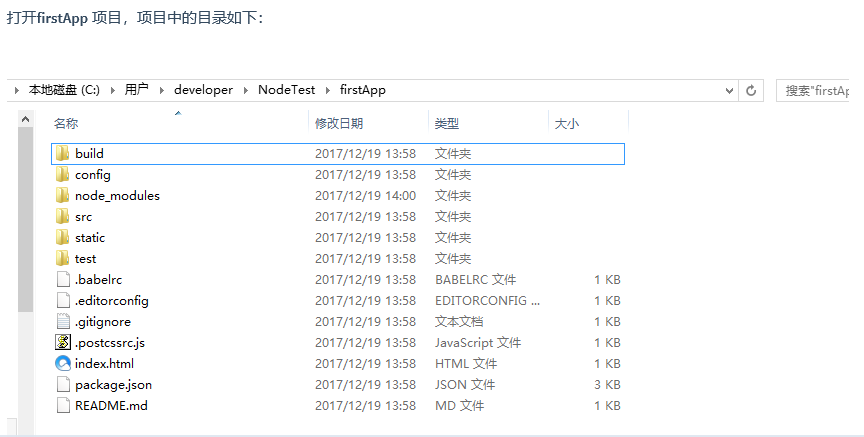
在命令行中运行命令 vue init webpack firstApp 。解释一下这个命令，这个命令的意思是初始化一个名为firstApp项目，其中webpack是构建工具，也就是整个项目是基于webpack的。其中firstApp是整个项目文件夹的名称，这个文件夹会自动生成在你指定的目录中

我下面这张图是：项目文件夹名：vue-agriculture



自动装依赖错误，因为我用了淘宝镜像。所以装依赖要自己手动装，请继续往下看

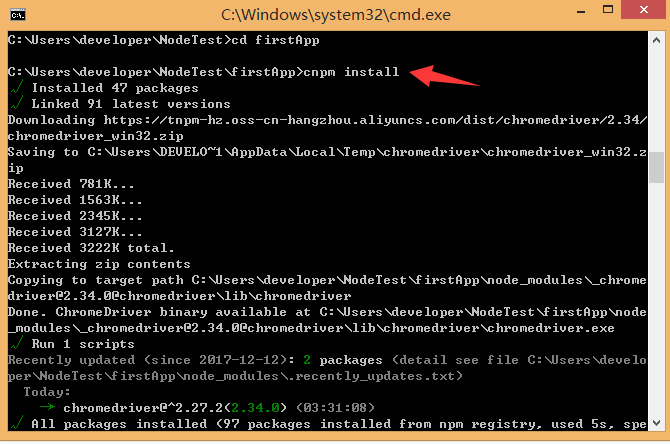
### （2）目录结构：



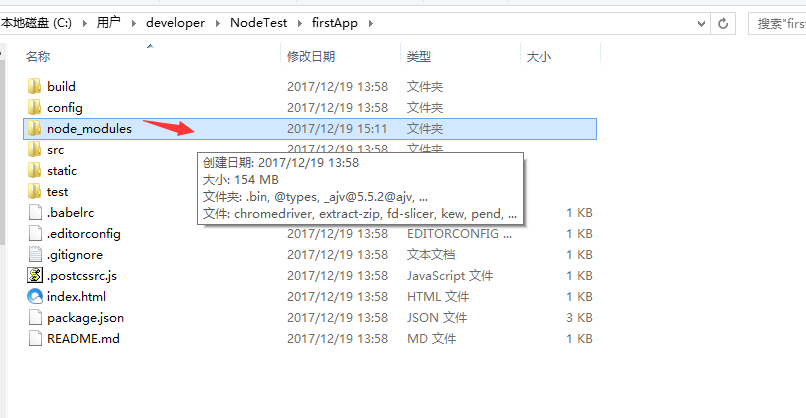
|  |
| --- |
| 介绍一下目录及其作用：       build：最终发布的代码的存放位置。       config：配置路径、端口号等一些信息，我们刚开始学习的时候选择默认配置。       node\_modules：npm 加载的项目依赖模块。       src：这里是我们开发的主要目录，基本上要做的事情都在这个目录里面，里面包含了几个目录及文件：               assets:放置一些图片，如logo等               components:目录里放的是一个组件文件，可以不用。               App.vue：项目入口文件，我们也可以将组件写这里，而不使用components目录。              main.js ：项目的核心文件       static：静态资源目录，如图片、字体等。       test：初始测试目录，可删除        .XXXX文件：配置文件。       index.html：首页入口文件，可以添加一些meta信息或者同统计代码啥的。       package.json：项目配置文件。       README.md：项目的说明文件。 |

### （3）安装依赖（非常重要）

安装项目所需要的依赖：在项目目录下，执行 cnpm install   (这里可以用cnpm代替npm了)



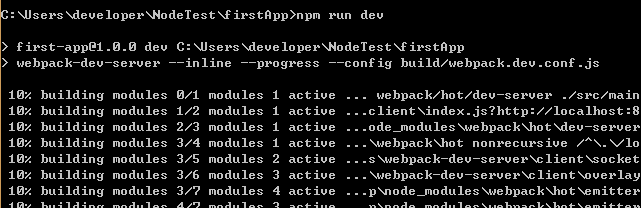
    安装完成之后，我们到自己的项目中去看，会多一个node\_modules文件夹，这里面就是我们所需要的依赖包资源。



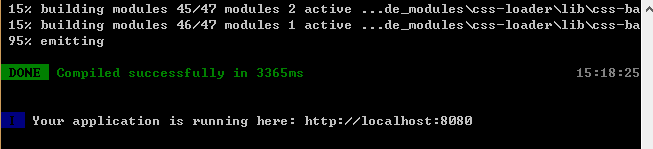
    安装完依赖包资源后，我们就可以运行整个项目了。

（4）**运行项目**

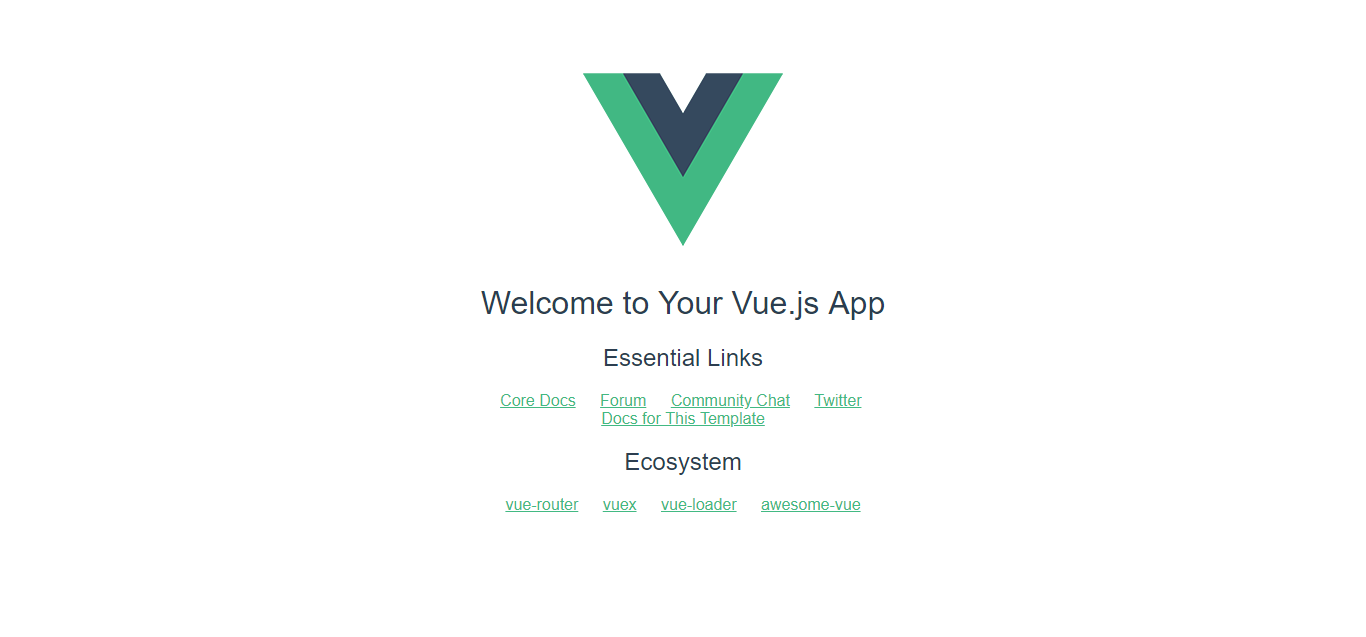
在项目目录中，运行命令 npm run dev ，会用热加载的方式运行我们的应用，热加载可以让我们在修改完代码后不用手动刷新浏览器就能实时看到修改后的效果。



    项目启动后，在浏览器中输入项目启动后的地址：



在浏览器输入 <http://localhost:8080> 即可看到



表明成功

## 在Vue-cli中引入Element Ui

(1)安装element-ui

$ cnpm i element-ui -S

建议固定vue和element-ui的版本，避免将来版本升级后产生冲突

(2) 引入element-ui（此为完整引入，若想按需引入请移步官方文档）

修改 /src/main.js 为：（这部分可以看官方文档）

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | import Vue from 'vue' import ElementUI from 'element-ui' import 'element-ui/lib/theme-chalk/index.css'  import App from './App'  Vue.config.productionTip = false  Vue.use(ElementUI)  /\* eslint-disable no-new \*/ new Vue({  el: '#app',  template: '<App/>',  components: { App } }) |  |

红色为需添加的

## 4、[Element-UI 自定义样式](https://segmentfault.com/q/1010000009483822)

还未解决

1、猜想：我也遇到了类似的问题，网上找了好久，官方的方法比较适合针对某个项目进行全方位的样式重构，如果只是在开发中引用一两个组件太麻烦，如果直接在组件样式里重构，因为有scoped的存在，会给类名加一个唯一标识符，所以重写样式无法修改，去掉scoped又会污染其他组件样式，我最后想了个方法。直接在组件里加一层div，类名和组件一致，在这个类名下进行样式修改然后把style的scoped去掉，这样又能针对这个组件重置UI库的样式又不会影响其他组件。如果想整体修改的话直接去改源码就行了，或者在App.vue下改。

2、<https://vue-loader-v14.vuejs.org/zh-cn/features/scoped-css.html>

<https://segmentfault.com/a/1190000012184604?utm_source=tuicool&utm_medium=referral#articleHeader6>

<https://blog.csdn.net/Mr_HeXS/article/details/80375244>

## 配置路由（实现页面跳转）

1. 之前脚手架创建项目时，已经勾选路由，就不要额外引入

也不要额外的配置

以后如果新建了一个页面test.vue,要想能跳转到，我们需要在router/index.js中增加：

|  |
| --- |
| **import** test **from** '@/components/test'  **export default new** Router({  mode:'history',//删除url上的/#/  routes: [  {  path: '/',  name: 'HelloWorld',  component: HelloWorld  },  {  path: '/test',  name: 'test',  component: test  }  ] }) |

以上的下划线

即可访问

后来我发现这个方法不太好，但可以用。

1. 如果脚手架创建项目时，已经没有勾选路由，需要（推荐）

参考文档：

<https://router.vuejs.org/zh/guide/#html>

(1).安装路由模块

npm install vue-router --save-dev

(2).在main.js中进行如下操作

//引入vue-router

import VueRouter from 'vue-router'

//使用路由

Vue.use(VueRouter)

//配置路由

const router = new VueRouter({

routes:[

{path:"/路径"，component:vue名}

{...}

]，

mode:"history" //删除网址路径上的/#/

})

|  |
| --- |
| **import** Vue **from** 'vue' **import** Router **from** 'vue-router'  Vue.use(Router);  **const** router = **new** Router({  mode:'history',  routes: [  {  path: '/test',  name:'test',  component: (resolve) => require(['../components/test.vue'], resolve)  },  {  path: '/login',  name:'login',  component: (resolve) => require(['../components/login.vue'], resolve)  },  {  path: '/index',  name:'index',  component: (resolve) => require(['../components/index.vue'], resolve)  },  ]  });  **export default** router; |

//实例化引用router

在main.js下面的new Vue({})中添加“router，”即可

|  |
| --- |
| **new** Vue({  el: '#app',  router,  components: { App },  template: '<App/>'  // render: h => h(App) }).$mount('#app'); |

(3).App.vue中添加显示

<router-view></router-view> //这个一定要加

或者

<router-view/>

## 6、在 Vue-cli 中引入 Semantic UI（该框架与vue不太兼容，建议不要用）

(1) 首先，我们需要先安装 jQuery

cnpm **install** --save **jquery**

(2) 然后在 webpack.dev.config.js 文件中，添加

|  |
| --- |
| // plugins 区块内  **new** webpack.ProvidePlugin({  $ : "jquery",  jQuery : "jquery",  "window.jQuery": "jquery",  "root.jQuery" : "jquery"  }) |

(3) 随后安装 semantic-ui-css

cnpm **install** semantic-ui-css --save

(4) 之后在 webpack.base.config.js 文件中添加 //Semantic-UI的那部分

|  |
| --- |
| // resolve 区块  resolve : {  extensions: ['', '.js', '.vue'],  fallback : [path.join(\_\_dirname, '../node\_modules')],  alias : {  'vue' : 'vue/dist/vue.common.js',  'src' : path.resolve(\_\_dirname, '../src'),  'assets' : path.resolve(\_\_dirname, '../src/assets'),  'components': path.resolve(\_\_dirname, '../src/components'),  // Semantic-UI  'semantic' : path.resolve(\_\_dirname, '../node\_modules/semantic-ui-css/semantic.min.js')  }  } |

（5）随后在 webpack.dev.config.js 文件中，添加

|  |
| --- |
| plugins: [  **new** webpack.ProvidePlugin({  $ : "jquery",  jQuery : "jquery",  "window.jQuery": "jquery",  "root.jQuery" : "jquery",  // Semantic-UI  semantic : 'semantic-ui-css',  Semantic : 'semantic-ui-css',  'semantic-ui': 'semantic-ui-css'  }), |

（6）接下来，引入 css 和 js 文件:

参考：

vue项目中引入外部css以及js文件的方法 - CSDN博客

<https://blog.csdn.net/wojiaomaxiaoqi/article/details/78050501>

我在main.js引入

|  |
| --- |
| **import** semantic **from** 'semantic'  **import** '../node\_modules/semantic-ui-css/semantic.min.css' |

（7）随便找个semantic组件测试一下，即可

参考：

在 Vue-cli 中引入 Semantic UI - wiibe - SegmentFault 思否

<https://segmentfault.com/a/1190000007248072>

## 7、通过axios来发送请求（axios的安装，引用）

(1）.安装axios

cnpm install axios

(2). 启动项目

npm run dev

（3）

安装其他插件的时候，可以直接在 main.js 中引入并 Vue.use()，但是 axios 并不能 use，只能每个需要发送请求的组件中即时引入.为了解决这个问题，有两种开发思路，一是在引入 axios 之后，修改原型链，二是结合 Vuex，封装一个 aciton。具体的实施请往下看~

方案一：改写原型链

在main.js中全局引入

import axios from 'axios'

Vue.prototype.$axios = axios

(4).在需要的方法中如下面的例子使用：

|  |
| --- |
| methods: {  login:**function** () {   **var** fd = **new** FormData(); //模拟form表单提交数据  fd.append('username',**this**.ruleForm2.username);  fd.append('password',**this**.ruleForm2.pass);   **let** config = { //请求配置  headers: {  'Content-Type': 'multipart/form-data'  }  };   **this**.$axios.post("http://112.74.53.186:8080/auth/login", fd,config).then( res => {  console.log(res)  }).catch( res => {  console.log(res)  })   } } |
|  |

参考资料：

方案二：

Vue 爬坑之路（六）—— 使用 Vuex + axios 发送请求 - WiseWrong - 博客园

<https://www.cnblogs.com/wisewrong/p/6402183.html>

<https://juejin.im/post/5a02a898f265da43052e0c85>

使用说明 · Axios 中文说明 · 看云(中文官方文档一定要看！！！)

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

## 8、关于axios的一系列问题！！！（踩坑！）

环境：前端vue（本地） 后端springboot（部署在服务器上）

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

### 8.1、前后端交互的跨域问题

问题如下：

|  |
| --- |
| C:\Users\zj\Desktop\2345截图20180725223755.png |
| C:\Users\zj\Desktop\2345截图20180725223809.png |

解决方案（前后端都要配置）：

后端：**在springboot中如何通过cors协议解决跨域问题**

|  |
| --- |
| @Configuration  public class MyWebAppConfigurer extends WebMvcConfigurerAdapter{  @Override  public void addCorsMappings(CorsRegistry registry) {  registry.addMapping("/\*\*");//允许跨域  } |

参考：

springboot中通过cors协议解决跨域问题 - 坏~牧羊人 - 博客园

<https://www.cnblogs.com/520playboy/p/7306008.html>

但这样还不够，我还是有问题。接下来配置前端

前端：如何使用http-proxy-middleware 代理解决（项目使用vue-cli脚手架搭建）

（1）、打开config/index.js,在proxyTable中添写如下代码：

|  |
| --- |
| proxyTable: {  '/api': {  target:'http://112.74.53.xxx:8080', // 你接口的域名  //secure: false, // 如果是https接口，需要配置这个参数  changeOrigin:**true**, // 如果接口跨域，需要进行这个参数配置  pathRewrite: {  '^/api':'http://112.74.53.xxx:8080'  }  } }, |
|  |

(2) 使用axios发送get请求时直接使用“/api”：

|  |
| --- |
| getPersonalInformation:**function** () {  **var** token='Bearer '+sessionStorage.getItem("Token");  **let** config = {  headers: {  'Authorization':token  }  };  console.log(token);  **var** url = "/api/getPersonalInformation";  console.log(url);  **this**.$axios.get(url,config).then(res=> {  console.log(res);  }) .catch( res => {  console.log(res); })  } |

3、(下面方法并未测试)

changeOrigin参数，接收一个布尔值，如果设置为true,那么本地会虚拟一个服务端接收你的请求并代你发送该请求，这样就不会有跨域问题了，所以这只适用于开发环境。

|  |  |  |
| --- | --- | --- |
| 通过这中方法去解决跨域，打包部署时还按这种方法会出问题。解决方法如下：   |  |  | | --- | --- | | 1  2  3  4  5 | let serverUrl = '/api/'  //本地调试时  // let serverUrl = 'http://f.apiplus.cn/'  //打包部署上线时  export default {    dataUrl: serverUrl + 'bj11x5.json'  } | |

参考：vue跨域解决方法 - 王永存 - 博客园

<https://www.cnblogs.com/wangyongcun/p/7665687.html>

vue.js - axios添加了header头信息后发送的get请求自动变成option请求了，为什么？？？ - SegmentFault 思否

<https://segmentfault.com/q/1010000009544146>

到此，开发环境下跨域问题就完美解决了

### 8.2、axios 发送post请求时，参数丢失问题

axios默认发送数据时，数据格式是Request Payload，而并非我们常用的Form Data格式

要了解form data和request payload格式的区别！！！

从axios看前后端数据交互 - CSDN博客

<https://blog.csdn.net/Brave_Coder/article/details/78379073>

vue axios POST请求中参数以form data和request payload形式的原因 - CSDN博客

<https://blog.csdn.net/yusirxiaer/article/details/78057720>

<https://blog.csdn.net/u010187242/article/details/73609186>

vue 添加axios组件，解决post传参数为null问题 - CSDN博客

<https://blog.csdn.net/wild46cat/article/details/78447467>

axios,post请求传递不了参数 - 我是好人的回答 - SegmentFault 思否

<https://segmentfault.com/q/1010000008698726/a-1020000008699952>

看了上面的文章你一定会对这类问题有所了解，下面来贴出解决方案:

1. post请求，模拟form表单传参（数据格式Form Data格式）

|  |
| --- |
| var fd = new FormData()  fd.append('file',’111’)  let config = {  headers: {  'Content-Type': 'multipart/form-data'  }  }  axios.post(that.uploadUrl, fd,config).then( res => {  console.log(res)  }).catch( res => {  console.log(res)  })  } |

2、axios传参（默认的数据格式Request Payload）

|  |  |
| --- | --- |
| **return** {  ruleForm2: {  password: '',  username: '',  }, |  |
| login:**function** () {   **let** config = {  headers: {  'Content-Type': 'application/x-www-form-urlencod'  }  };   **this**.$axios.post("http://112.74.53.xxx:8080/auth/login", **this**.ruleForm2,config).then( res => {  console.log(res);   }).catch( res => {  console.log(res)  }) |  |

或者

|  |
| --- |
| this.$axios.post("http://112.74.53.xxx:8080/auth/login", {  ‘username’:’111’,  ‘password’:’222’,  },config).then( res => {  console.log(res);   }).catch( res => {  console.log(res)  }) |
|  |

### 8.3、axios 问题集锦

1、

|  |
| --- |
| C:\Users\zj\Desktop\QQ截图20180820221217.png |

本以为是axios get请求中文乱码的问题，后来发现是405（Method not allowed）的问题

后台是post请求，但请求是get请求

<https://blog.csdn.net/qq_35804654/article/details/77272304>

## 9、登录时保存信息，SessionStorage和Vuex的区别

<https://segmentfault.com/q/1010000011938678>

### （1）SessionStorage 的使用方法

登录时将信息存到session中，注销时

|  |
| --- |
| sessionStorage.setItem('Token', res.data.body.token);//将token存到sesion中 |
| sessionStorage.getItem('Token');//从 sesion中读取‘Token’ |

### (2)Vuex的使用方法（中大型项目推荐使用）

Vuex 是一个专为 Vue.js 应用程序开发的**状态管理模式**。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。Vuex 也集成到 Vue 的官方调试工具 [devtools extension](https://github.com/vuejs/vue-devtools)，提供了诸如零配置的 time-travel 调试、状态快照导入导出等高级调试功能。

官方文档：

<https://vuex.vuejs.org/zh/installation.html>

1、安装Vuex

cnpm install vuex --save

需要注意的是这里一定要加上 --save，因为你这个包我们在生产环境中是要使用的

2、在src目录下，新建一个vuex文件夹（这个不是必须的），并在文件夹下新建store.js文件，文件中引入我们的vue和vuex。

|  |
| --- |
| **import** Vue **from** 'vue'; **import** Vuex **from** 'vuex'; |

3、使用我们vuex，引入之后用Vue.use进行引用。

|  |
| --- |
| Vue.use(Vuex); |

通过这三步的操作，vuex就算引用成功了，接下来我们就可以尽情的玩耍了。

4、如何使用

（1）现在我们store.js文件里增加一个常量对象。store.js文件就是我们在引入vuex时的那个文件。

|  |
| --- |
| **const** state={  count:1 }; |

（2）用export default 封装代码，让外部可以引用。

|  |
| --- |
| **export default new** Vuex.Store({  state }) |

参考资料：

<https://juejin.im/entry/59191b6b0ce4630069f6a3ad>

<https://segmentfault.com/a/1190000005863691>

## 10、vue项目改名怎么办？

**1、 删除 node\_modules 文件夹（如果修改项目名称，需要在在package.json中修改对应的name）**

2、重新安装依赖 cnpm install

3、启动项目 npm run dev

## 11、生命周期（钩子函数）？？？

Vue组件钩子函数（与mothods方法平级）

beforeCreate:function(){}组件实例化之前执行的函数

created:function(){}组件实例化完毕，但页面还未显示

beforeMount:function(){}组件挂载前，页面仍未显示，但虚拟Dom已经配置

mount:function(){}组件挂载后，此方法执行后，页面显示

beforeUpdate:function(){}组件更新前，页面仍未更新，但虚拟Dom已经配置

updated:function(){}组件更新，此方法执行后，页面显示

beforeDestory:function(){}组件销毁前

destoryed:function(){}组件销毁

# 四、前端页面功能实现

## 1、element 的layout布局（bootstrap栅格系统）

全局 CSS 样式 · Bootstrap v3 中文文档

<https://v3.bootcss.com/css/>

栅格系统用于通过一系列的行（row）与列（column）的组合来创建页面布局，你的内容就可以放入这些创建好的布局中。下面就介绍一下 Bootstrap 栅格系统的工作原理：

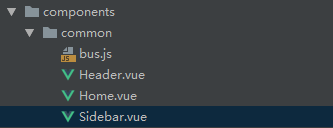
* “行（row）”必须包含在 .container （固定宽度）或 .container-fluid （100% 宽度）中，以便为其赋予合适的排列（aligment）和内补（padding）。
* 通过“行（row）”在水平方向创建一组“列（column）”。
* 你的内容应当放置于“列（column）”内，并且，只有“列（column）”可以作为行（row）”的直接子元素。
* 类似 .row 和 .col-xs-4 这种预定义的类，可以用来快速创建栅格布局。Bootstrap 源码中定义的 mixin 也可以用来创建语义化的布局。
* 通过为“列（column）”设置 padding 属性，从而创建列与列之间的间隔（gutter）。通过为 .row 元素设置负值 margin 从而抵消掉为 .container 元素设置的 padding，也就间接为“行（row）”所包含的“列（column）”抵消掉了padding。

## 2、CSS Position(定位) | 菜鸟教程

http://www.runoob.com/css/css-positioning.html

## 3、导航栏，侧边栏的继承（组件的嵌套）（即局部注册组件）

目录结构：



其中：header.vue是导航栏，sidebar.vue是侧边栏

home.vue是将导航栏，侧边栏结合起来的

home.vue:

|  |
| --- |
| <template>  <div class="wrapper">  <v-head></v-head>  <v-sidebar></v-sidebar>  <div class="content-box" :class="{'content-collapse':collapse}">  <div class="content">  <transition name="move" mode="out-in">  <keep-alive :include="tagsList">  <router-view></router-view>  </keep-alive>  </transition>  </div>  </div>  </div> </template>  <script>  **import** vHead **from** './Header.vue';//组件的引入  **import** vSidebar **from** './Sidebar.vue';  **import** vTags **from** './Tags.vue';  **import** bus **from** './bus';  **export default** {  data(){  **return** {  tagsList: [],  collapse: **false** }  },  components:{  vHead, vSidebar, “app-tag”:vTags //组件的注册,app-tag别名  },  created(){  bus.$on('collapse', msg => {  **this**.collapse = msg;  })   // 只有在标签页列表里的页面才使用keep-alive，即关闭标签之后就不保存到内存中了。  bus.$on('tags', msg => {  **let** arr = [];  **for**(**let** i = 0, len = msg.length; i < len; i ++){  msg[i].name && arr.push(msg[i].name);  }  **this**.tagsList = arr;  })  }  } </script> |

以上下划线的是关键。

最后通过路由

|  |
| --- |
| {  path: '/',  meta: {  title: '自述文件',  },  component: resolve => require(['../components/common/Home.vue'], resolve),  children: [  {  path: '/homepage',  component: resolve => require(['../components/page/HomePage.vue'], resolve),  meta: {title: '系统首页'}  },  ] } |

childern的下的所有组件(比如homepage)都会继承父组件Home.vue的导航栏，侧边栏（即内容）

## 4、页面的跳转

<http://www.cnblogs.com/wisewrong/p/6277262.html>

## 5、页面加载成功时，触发事件

<https://segmentfault.com/q/1010000009485473>

vue钩子函数 create（）

## 6、vue 加载远程图片（ip+端口+url）

|  |
| --- |
| <img v-bind:src="avater"> |
| data() {  **return** {  avater:sessionStorage.getItem("avater"),  } }, |

## 7、element-ui upload组件+springBoot 实现图片的上传

只能借鉴：

<https://blog.csdn.net/qq_39774931/article/details/81773706>

### 7.1后端代码

|  |
| --- |
| //上传头像（新）  @PostMapping("/ftp/headImg")  public Object uploadHeadImgNew(@RequestParam("file") MultipartFile[] files) throws IOException {  String username = SecurityContextHolder.*getContext*().getAuthentication().getName();//获取当前登录用户  System.*out*.println( username );  User user = userService.getUserByUsername(username);   FtpConfig ftpConfig = new FtpConfig();  //System.out.println( ftpConfig.getFTP\_ADDRESS() );  for (MultipartFile file : files) {  //Photo photo = new Photo();  String oldName = file.getOriginalFilename();// 获取图片原来的名字  System.*out*.println( oldName );  String picNewName = UploadUtils.*generateRandonFileName*(oldName);// 通过工具类产生新图片名称，防止重名  System.*out*.println( picNewName );  //String picSavePath = UploadUtils.generateRandomDir(picNewName);// 通过工具类把图片目录分级  String picSavePath = "/"+username;//以用户名为目录  System.*out*.println( picSavePath );  String pictureImageUrl = picSavePath+"/"+picNewName; /\*  \* photo.setPhotoUrl(picSavePath + "/");//  \* 设置图片的url--》就是存储到数据库的字符串url photo.setAlbumId(albumId);//  \* 设置图片所属相册id photo.setUser\_id("wk");  \* photo.setPhoteName(picNewName);  \*/  //photoList.add(photo);  FtpUtil.*pictureUploadByConfig*(ftpConfig, picNewName, picSavePath, file.getInputStream());// 上传到图片服务器的操作  // 添加到数据库  user.setAvater(pictureImageUrl); //设置头像路径  userService.saveOrUpdate(user);//修改用户信息  }   return new Reponse( true,"上传头像成功" );  } |

注意：接受的参数

### 7.2、前端代码

以下的代码来自elementui官网，做出一点改动（用下滑线标出）

|  |
| --- |
| <el-upload  name="file" //后台接收文件流的参数名  class="upload-demo"  :action="imageAction" //api接口  :headers="urlHeaders" //设置上传的请求头部  :on-preview="handlePreview"  :on-remove="handleRemove"  :before-remove="beforeRemove"  multiple  :limit="1" //限制文件数量  :on-exceed="handleExceed"  :file-list="fileList">  <el-button size="small" type="primary">点击上传</el-button>  <div slot="tip" class="el-upload\_\_tip">只能上传jpg/png文件，且不超过500kb</div> </el-upload>  <script>  data() {  **return** {    fileList: [],  imageAction:`api/ftp/headImg`,//请求地址。注意单引号，表示String型  urlHeaders:{  'Authorization':`Bearer ${sessionStorage.getItem("Token")}`//在请求头上增加token。注意格式！！！  }   } },  methods: {   handleRemove(file, fileList) {  console.log(file, fileList);  },  handlePreview(file) {  console.log(file);  },  handleExceed(files, fileList) {  **this**.$message.warning(`当前限制选择 1 个文件，本次选择了 ${files.length} 个文件，共选择了 ${files.length + fileList.length} 个文件`);  },  beforeRemove(file, fileList) {  **return this**.$confirm(`确定移除 ${ file.name }？`);  },    },  </script> |

上面红色的都是关键

亲测，可行

### 7.3、发现问题：数据更新，但组件没有更新

|  |
| --- |
| <img class="user-avater" v-bind:src="avater" >  <el-button type="primary" @click="dialogVisible = **false**;personChange(nickname);getPersonalInformation()">提 交</el-button>  data() {  **return** {  avater:"",  } },  methods: {   getPersonalInformation:**function** () {  **var** token='Bearer '+sessionStorage.getItem("Token");  **let** config = {  headers: {  'Authorization':token,  }  };  console.log(token);  **var** url = "/api/getPersonalInformation";  console.log(url);  **this**.$axios.get(url,config).then(res=> {  console.log(res);  **if**(res.data.success===**true**) {  //sessionStorage.setItem('avater', res.data.body.avater);  **this**.personalInfo = res.data.body;  **this**.username = res.data.body.username;  **this**.nickname = res.data.body.nickname;  **this**.avater =store.state.pictureUrl+res.data.body.avater;  } **else** {  **this**.$message.error("获取个人信息失败");  }  }).catch( res => {  console.log(res);  })  }, |

通过vue的数据双向绑定，来实现组件实时更新

## 8、upload组件（2）上传文件，携带额外参数，文件类型限制，文件大小限制

### 8.1、后端代码

|  |
| --- |
| //ftp+nginx 上传视频 @PostMapping("/ftp/videoUpload") public Object FtpVideoUpLoad(@RequestParam("title") String title,  @RequestParam("description") String description,  @RequestParam("file") MultipartFile file,  @RequestParam("tags") String tags) throws IOException {   String author = SecurityContextHolder.*getContext*().getAuthentication().getName();//获取当前登录用户   Video video = new Video();  String fileName = file.getOriginalFilename();//获取视频文件名   boolean bflag = false;  if (file.getSize() != 0 && !file.isEmpty()) {//如果视频文件非空  bflag = true;  // 判断文件大小  if (file.getSize() <= *upload\_maxsize*) {//判断文件大小是否超载  bflag = true;  if (videoService.checkFileType( fileName )) {// 文件类型判断  bflag = true;  } else {  bflag = false;  System.*out*.println( "文件类型不允许" );  }  } else {  bflag = false;  System.*out*.println( " 文件大小超范围" );  }  } else {  bflag = false;  System.*out*.println( "文件为空" );  }   if (bflag) {//若上传的文件符合以上条件,则上传文件到临时文件夹中   // 获得文件扩展名  String fileEnd = videoService.getFileExt(fileName);  //产生新图片名称，防止重名(不含扩展名)  String newFileName = videoService.getName(fileName);   // 设置转码后文件的 临时保存路径,因为转码已注释，所以可以随便填  String codcFilePath ="";  // 获取配置的转换工具（ffmpeg.exe）的存放路径  String ffmpegPath = ".\\src\\main\\tools\\ffmpeg.exe";  //设置截图的临时存放路径  String mediaPicPath ="./src/main/webapp/"+ newFileName +".jpg";  //源文件临时存放路径  String filename = "./src/main/webapp/"+newFileName+fileEnd;   //保存文件到临时文件夹  BufferedOutputStream out = new BufferedOutputStream(  //C:\IDEA\_mode\_project\agriculture\src\main  new FileOutputStream( new File( filename) ) );//保存源文件到目录下  out.write( file.getBytes() );  out.flush();  out.close();  System.*out*.println( "保存文件到临时文件夹成功！" );   //视频第一帧化图截取  if (videoService.checkMediaType(fileEnd)) {//如果视频文件类型可以转码  boolean flag = videoService.executeCodecs(ffmpegPath, filename, codcFilePath,mediaPicPath); //该条语句是进行视频转码操作  if(flag) {  System.*out*.println( "视频转码成功！" );  }else {  System.*out*.println( "视频转码失败！" );  }  } else {  System.*out*.println( "视频文件类型不支持转码！" );  }   FtpConfig ftpConfig = new FtpConfig();  String picSavePath = "/"+author;//以用户名为目录   //将视频上传到服务器上  FtpUtil.*pictureUploadByConfig*(ftpConfig, newFileName+fileEnd, picSavePath,file.getInputStream() );// 上传到服务器的操作  //视频 在服务器上的相对地址  String videoUrl = picSavePath+"/"+newFileName+fileEnd;   File f = new File(mediaPicPath);  InputStream input = new FileInputStream(f);  //将图片上传到服务器上  FtpUtil.*pictureUploadByConfig*(ftpConfig, newFileName+".jpg", picSavePath,input );// 上传到服务器的操作  //截图 在服务器上的相对地址  String pictureUrl = picSavePath+"/"+newFileName+".jpg";  System.*out*.println( "上传文件到服务器成功！" );   //删除临时文件夹下的文件  File file1=new File(filename);  file1.delete();  File file2=new File(mediaPicPath);  file2.delete();  System.*out*.println( "删除临时文件成功" );   //保存到数据库  video.setAvater( videoUrl );  video.setTitle( title );  video.setAuthor( author );  video.setAuthor\_picture( userService.getUserByUsername( author ).getAvater() );  video.setDescription( description );  video.setTags(tags);  video.setPicture( mediaPicPath); //记录视频截图保存路径    } else {  return new Reponse( false,"上传失败" );  }   videoService.saveOrUpdate( video );//更新数据库  return new Reponse( true,"上传成功",video); } |

### 8.2、前端代码（大体上是官网上的例子）

|  |
| --- |
| <template> <div>  <el-radio-group v-model="labelPosition" size="small">  <el-radio-button label="left">左对齐</el-radio-button>  <el-radio-button label="right">右对齐</el-radio-button>  <el-radio-button label="top">顶部对齐</el-radio-button>  </el-radio-group>  <div style="margin: 20px;">上传视频</div>  <el-form :label-position="labelPosition" label-width="80px" :model="formLabelAlign">  <el-form-item label="视频标题">  <el-input v-model="formLabelAlign.name"></el-input>  </el-form-item>  <el-form-item label="视频简介">  <el-input v-model="formLabelAlign.region"></el-input>  </el-form-item>  </el-form>   <el-tag  :key="tag"  v-for="tag in dynamicTags"  closable  :disable-transitions="**false**"  @close="handleClose(tag)">  {{tag}}  </el-tag>  <el-input  class="input-new-tag"  v-if="inputVisible"  v-model="inputValue"  ref="saveTagInput"  size="small"  @keyup.enter.native="handleInputConfirm"  @blur="handleInputConfirm"  >  </el-input>  <el-button v-else class="button-**new**-tag" size="small" @click="showInput">+ 添加标签</el-button>   <el-upload  class="upload-demo"  name="file"  :action="apiAction"  :headers="urlHeaders"  accept=",.mp4,.jpg"//accept支持上传文件格式  :data="uploadExtraData"//携带额外参数  :on-preview="handlePreview"  :on-remove="handleRemove"  :before-remove="beforeRemove"  :beforeUpload="beforeAVatarUpload"//上传文件之前的钩子，参数为上传的文件，若返回 false 或者返回 Promise 且被 reject，则停止上传  multiple  :limit="1"  :on-exceed="handleExceed"  :file-list="fileList">  <el-button size="small" type="primary">点击上传</el-button>  <div slot="tip" class="el-upload\_\_tip">只能上传.mp4文件</div>  </el-upload>  </div> </template>  <script>  **export default** {  name: "VideoBlock",  data: **function** () {  **return** {  labelPosition: 'right',  formLabelAlign: {  name: '',  region: '',  },  dynamicTags: [],  inputVisible: **false**,  inputValue: '',  fileList: [],  apiAction: '/api/ftp/videoUpload',//api接口  urlHeaders: {  'Authorization': `Bearer ${sessionStorage.getItem("Token")}`//在请求头上增加token。注意格式！！！  },  uploadExtraData: { //需要携带的其他参数.注意格式！！！  title: "", //实例化参数时，实在没定值，必须赋空值  description: "",  tags:"",  },   };  },   methods: {   beforeAVatarUpload(file) {  **this**.uploadExtraData.title=**this**.formLabelAlign.name;//在这里赋值！！  **this**.uploadExtraData.description=**this**.formLabelAlign.region;  **this**.uploadExtraData.tags=**this**.dynamicTags;  **const** isLt2M = file.size / 1024 / 1024 < 10;//判断大小  **if** (!isLt2M) {  **this**.$message.error('上传文件大小不能超过 10MB!')  }  **return** isLt2M  },  handleClose(tag) {  **this**.dynamicTags.splice(**this**.dynamicTags.indexOf(tag), 1);  },   showInput() {  **this**.inputVisible = **true**;  **this**.$nextTick(\_ => {  **this**.$refs.saveTagInput.$refs.input.focus();  });  },   handleInputConfirm() {  **let** inputValue = **this**.inputValue;  **if** (inputValue) {  **this**.dynamicTags.push(inputValue);  }  **this**.inputVisible = **false**;  **this**.inputValue = '';  },  handleRemove(file, fileList) {  console.log(file, fileList);  },  handlePreview(file) {  console.log(file);  },  handleExceed(files, fileList) {  **this**.$message.warning(`当前限制选择 1 个文件，本次选择了 ${files.length} 个文件，共选择了 ${files.length + fileList.length} 个文件`);  },  beforeRemove(file, fileList) {  **return this**.$confirm(`确定移除 ${ file.name }？`);  }  }  } </script>  <style scoped>   .el-tag + .el-tag {  margin-left: 10px;  }  .button-new-tag {  margin-left: 10px;  height: 32px;  line-height: 30px;  padding-top: 0;  padding-bottom: 0;  }  .input-new-tag {  width: 90px;  margin-left: 10px;  vertical-align: bottom;  }  </style> |

|  |
| --- |
| accept=".jpg,.jpeg,.png,.gif,.bmp,.pdf,.JPG,.JPEG,.PBG,.GIF,.BMP,.PDF"  //accept支持上传文件格式 |

参考：

<https://blog.csdn.net/qq_39774931/article/details/81773706>

<https://www.cnblogs.com/LWJ-booke/p/8570438.html>

<https://blog.csdn.net/qq_39685062/article/details/77036582>

## 9、视频播放器 vue-video-player

surmon-china/vue-video-player: 🎞 @videojs component for @vuejs

<https://github.com/surmon-china/vue-video-player>

### 9.1、安装依赖

cnpm install vue-video-player --save

### 9.2、引入样式

|  |
| --- |
| // 第一个是videoJs的样式，后一个是vue-video-player的样式，因为考虑到我其他业务组件可能也会用到视频播放，所以就放在了main.js内  require('video.js/dist/video-js.css')  require('vue-video-player/src/custom-theme.css') |

### 9.3、把VueVideoPlayer导入并挂在到vue上

|  |
| --- |
| //在main.js内  import VideoPlayer from 'vue-video-player'  Vue.use(VideoPlayer); |

### 9.4、编写业务组件videoPlayer.vue

|  |
| --- |
| <template>    <div class="item">  <div class="player">  <video-player class="vjs-custom-skin"  :options="playerOptions"  :playsinline="**true**"  @ready="playerReadied($event)"></video-player>  </div>  </div>   </template>  <script>  // videojs  **import** videojs **from** 'video.js'   // export  **export default** {  //name: "videoPlayer",  data() {  **return** {  playerOptions: {  height: '360',  playbackRates: [0.7, 1, 1.3, 1.5, 1.7],  sources: [{  type: "video/mp4",  //src: "http://7xkwa7.media1.z0.glb.clouddn.com/sample\_video\_L"  src: "http://192.168.23.130/picture/18083764688/71171535630091616.mp4"  }],  poster: "https://surmon-china.github.io/vue-quill-editor/static/images/surmon-3.jpg",  }  }  },  methods: {  playerReadied(player) {  **const** track = **new** videojs.AudioTrack({  id: 'my-spanish-audio-track',  kind: 'translation',  label: 'Spanish',  language: 'es'  })  player.audioTracks().addTrack(track)  // Get the current player's AudioTrackList object.  **const** audioTrackList = player.audioTracks()  // Listen to the "change" event.  audioTrackList.addEventListener('change', **function**() {  // Log the currently enabled AudioTrack label.  **for** (**let** i = 0; i < audioTrackList.length; i++) {  **const** track = audioTrackList[i]  **if** (track.enabled) {  videojs.log(track.label)  **return** }  }  })  }  }  } </script> <style scoped>  </style> |

vue2.0+vue-video-player - CSDN博客

<https://blog.csdn.net/qq_40963664/article/details/79883159>

记一次vue2.0+vue-video-player实现hls播放全过程 - CSDN博客

<https://blog.csdn.net/fei565789229/article/details/78925395>

## 10、列表的渲染

列表渲染 — Vue.js（）

<https://cn.vuejs.org/v2/guide/list.html#ad>

不完整的例子，但又代表意义

|  |
| --- |
| <el-row>  <el-col :span="8" v-for="item in personalVideoList" :key="item.id" >  <el-card :body-style="{ padding: '0px' }">  <img :src="docUrl+item.picture" class="image">  <div style="padding: 14px;">   <span>{{item.title}}</span>  <div class="bottom clearfix">  <time class="time">{{ currentDate }}</time>  <el-button type="text" class="button" @click="play(item.picture,item.avater)">播放</el-button>  </div>  </div>  </el-card>  </el-col> </el-row> |

## 11、表格的增删改查

## 5、数组遍历

（1）v-for循环

|  |
| --- |
| <ul>  <li v-for="boy in boys">  {{ character }}  </li>  </ul> |

3、页面布局

使用element框架