搭建

heima-leadnews

前端工程

VERSION - 1.0.0

|  |  |  |
| --- | --- | --- |
| 文档版本说明 | | |
| 文件状态  [ ] 草稿  [ √ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本 |  |
| 作者 | miukoo |
| 完成日期 | 2019-05-31 |

目录

[目录 2](#_Toc19114854)

[1. 概述 3](#_Toc19114855)

[2. 环境 3](#_Toc19114856)

[3. 创建管理系统项目[heima-leadnews-admin] 3](#_Toc19114857)

[3.1 创建工程 3](#_Toc19114861)

[3.2 导入工程 4](#_Toc19114862)

[3.3 SASS集成 5](#_Toc19114863)

[3.3.1 SASS安装 5](#_Toc19114864)

[3.3.2 配置SASS 5](#_Toc19114865)

[3.4 项目结构 6](#_Toc19114866)

[3.4.1 项目结构 6](#_Toc19114867)

[3.4.2 项目依赖 6](#_Toc19114868)

[3.4.3 源码结构 8](#_Toc19114869)

[3.5 项目封装 8](#_Toc19114870)

[3.5.1 Request封装 8](#_Toc19114871)

[3.6 运行项目 11](#_Toc19114872)

[3.6.1 运行Web 11](#_Toc19114873)

[4. 导入管理系统项目[heima-leadnews-admin] 11](#_Toc19114874)

# 概述

前端分为heima-leadnews-app（移动APP）、heima-leadnews-wemedia（自媒体）、heima-leadnews-admin（管理系统）三个独立项目。

这三个项目核心都是依托MVVM框架Vue技术，并进行拓展：

* heima-leadnews-app 拓展了WEEX+VUE的跨平台开发
* heima-leadnews-wemedia 拓展了ECHARTS+VUE的图表开发
* heima-leadnews-admin 则主要作为学员前端实战练习

# 环境

项目依赖环境（需提前安装好）：

* Nodejs已安装
* Cnpm、Nrm、Webpack
* Weex Toolkit
* WebStorm
* Weex Playground
* AndroidStudio
* Vue-cli

# 创建管理系统项目[heima-leadnews-admin]

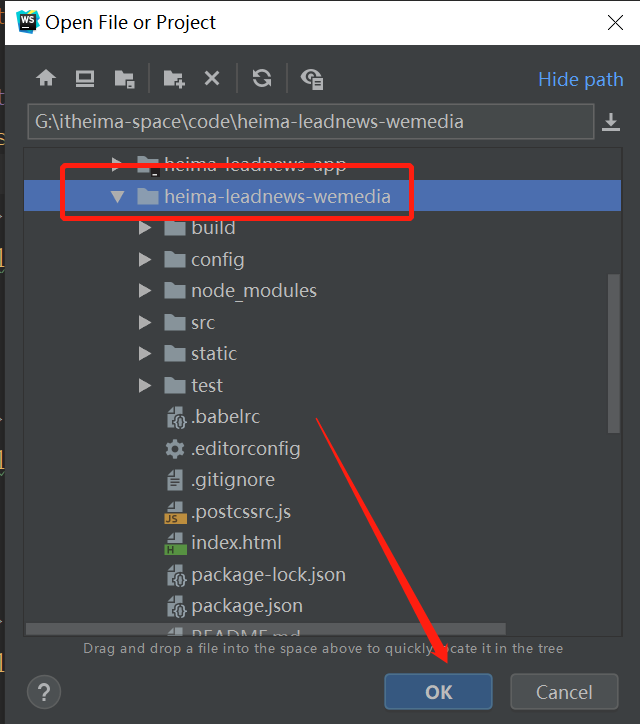


## 创建工程

* 进入项目存放路径
* 输入vue init webpack heima-leadnews-admin命令
* 选择设置项目信息

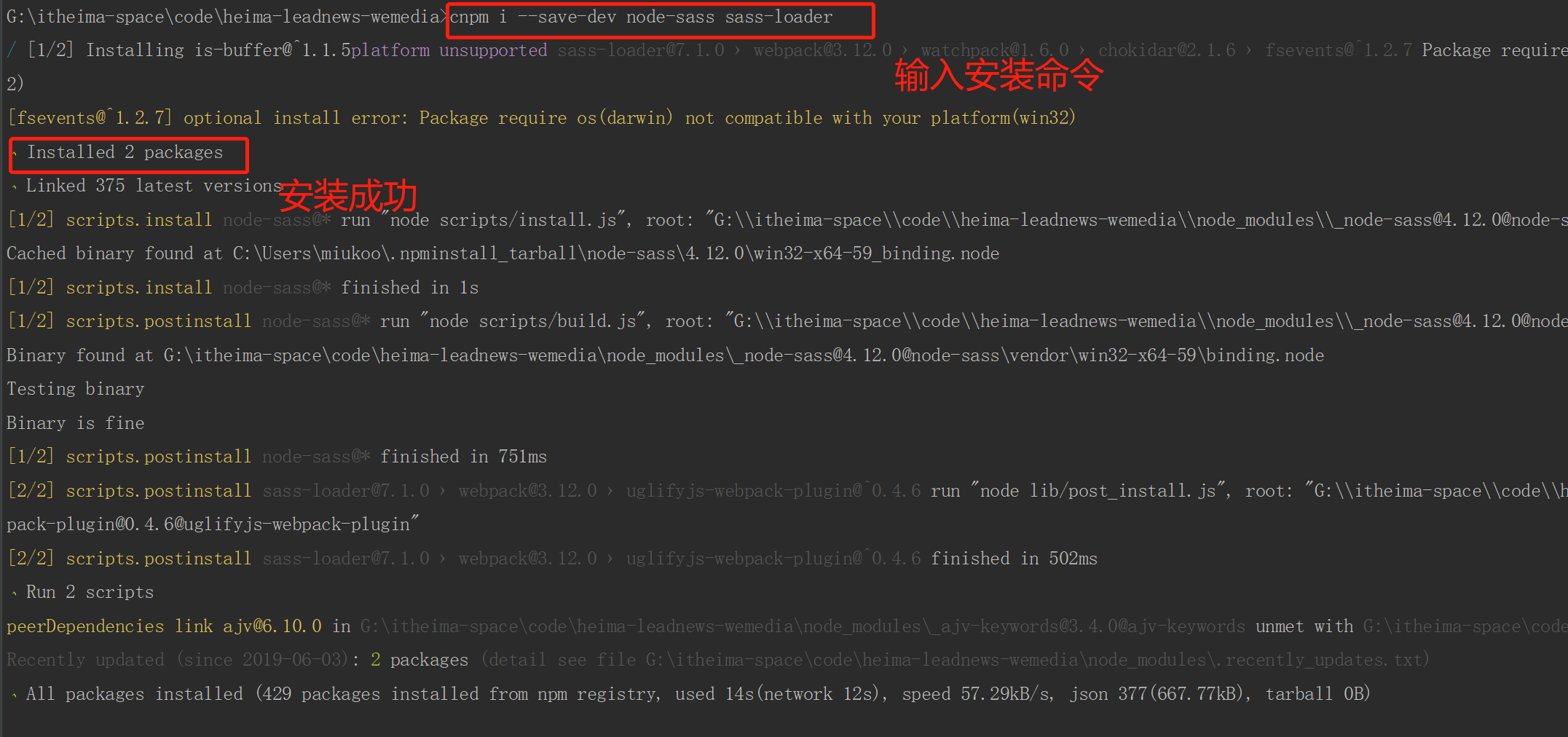


## 导入工程



## SASS集成

### SASS安装

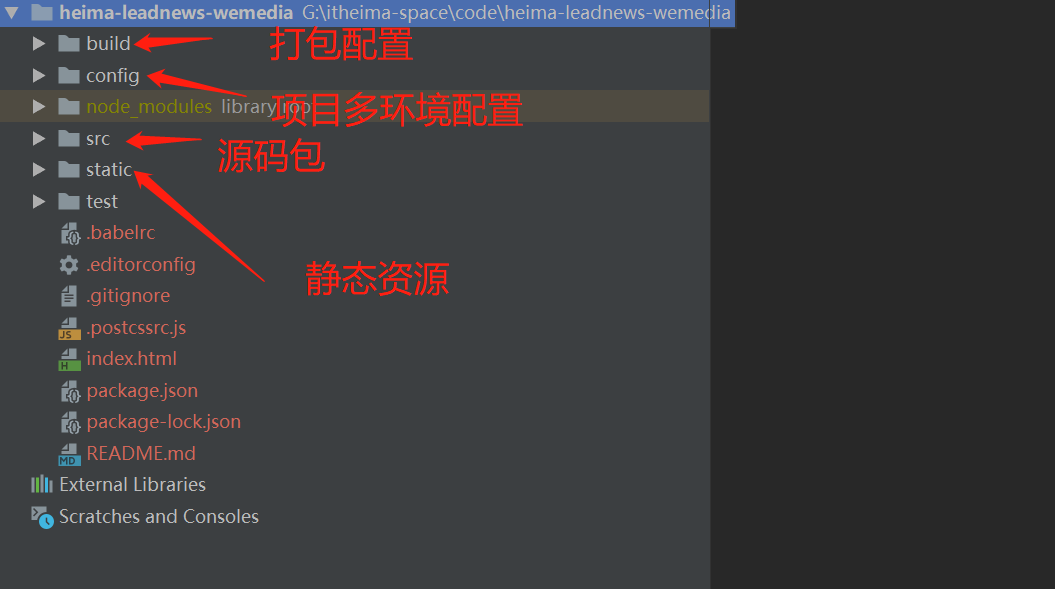


### 配置SASS

在vue-loader.conf.js文件中配置sass加载器

## 项目结构

### 项目结构

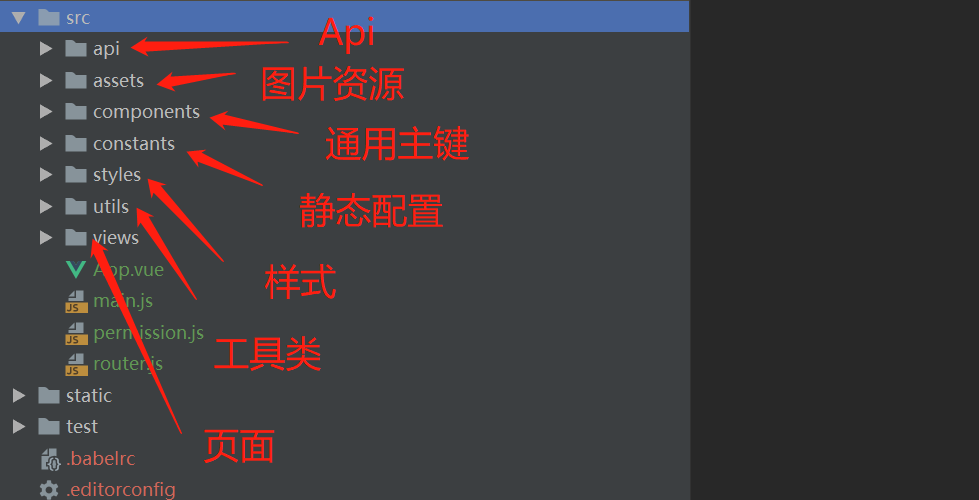


### 项目依赖

修改package.json相关部分后，在命令行执行cnpm –i进行依赖安装。

1. "dependencies": {
2. "@tinymce/tinymce-vue": "^2.0.0",
3. "axios": "0.18.0",
4. "echarts": "4.1.0",
5. "element-ui": "^2.4.11",
6. "js-cookie": "^2.2.0",
7. "json-bigint": "^0.3.0",
8. "nprogress": "0.2.0",
9. "tinymce": "4.9.1",
10. "vue": "^2.6.5",
11. "vue-router": "^3.0.1",
12. "vuex": "^3.0.1"
13. },
14. "devDependencies": {
15. "autoprefixer": "^7.1.2",
16. "babel-core": "^6.22.1",
17. "babel-helper-vue-jsx-merge-props": "^2.0.3",
18. "babel-jest": "^21.0.2",
19. "babel-loader": "^7.1.1",
20. "babel-plugin-dynamic-import-node": "^1.2.0",
21. "babel-plugin-syntax-jsx": "^6.18.0",
22. "babel-plugin-transform-es2015-modules-commonjs": "^6.26.0",
23. "babel-plugin-transform-runtime": "^6.22.0",
24. "babel-plugin-transform-vue-jsx": "^3.5.0",
25. "babel-preset-env": "^1.3.2",
26. "babel-preset-stage-2": "^6.22.0",
27. "chalk": "^2.0.1",
28. "copy-webpack-plugin": "^4.0.1",
29. "css-loader": "^0.28.0",
30. "extract-text-webpack-plugin": "^3.0.0",
31. "file-loader": "^1.1.4",
32. "friendly-errors-webpack-plugin": "^1.6.1",
33. "html-webpack-plugin": "^2.30.1",
34. "jest": "^22.0.4",
35. "jest-serializer-vue": "^0.3.0",
36. "node-notifier": "^5.1.2",
37. "node-sass": "^4.12.0",
38. "optimize-css-assets-webpack-plugin": "^3.2.0",
39. "ora": "^1.2.0",
40. "portfinder": "^1.0.13",
41. "postcss-import": "^11.0.0",
42. "postcss-loader": "^2.0.8",
43. "postcss-url": "^7.2.1",
44. "rimraf": "^2.6.0",
45. "sass-loader": "^7.1.0",
46. "semver": "^5.3.0",
47. "shelljs": "^0.7.6",
48. "uglifyjs-webpack-plugin": "^1.1.1",
49. "url-loader": "^0.5.8",
50. "vue-jest": "^1.0.2",
51. "vue-loader": "^13.3.0",
52. "vue-style-loader": "^3.0.1",
53. "vue-template-compiler": "^2.5.2",
54. "webpack": "^3.6.0",
55. "webpack-bundle-analyzer": "^2.9.0",
56. "webpack-dev-server": "^2.9.1",
57. "webpack-merge": "^4.1.0"
58. }

### 源码结构



## 项目封装

### Request封装

封装request主要目的在于统一处理一下事项：

* 请求授权头信息的处理
* 请求参数格式头信息设置
* 异常同一拦截处理

1. import axios from 'axios'
2. import BigInt from 'json-bigint'
3. import Router from  '@/router'
4. import { Message } from 'element-ui'
5. import { getUser } from   '@/utils/store'
6. import { API\_USERIMAGES , API\_HEAD} from  '@/constants/api'
7. import LoadingManage from './loading'
8. // create an axios instance
9. const service = axios.create({
10. baseURL: 'http://ttapi.research.itcast.cn/mp/v1\_0/', // api 的 base\_url
11. timeout: 10000, // request timeout
12. transformResponse (data) {
13. if(data)
14. return BigInt.parse(data)  //由于后端的数据库对id进行了变更 所以这里必须采用json-bigint插件来进行处理 保证数据正确
15. }
17. })
18. const IsImgUpload = (config) =**>** {
19. return  ((config.url == API\_USERIMAGES && config.method == 'post') || (config.url == API\_HEAD && config.method == 'patch'))
20. }
21. //request
22. service.interceptors.request.use(
23. config =**>** {
24. //LoadingManage.openLoading(); //打开弹层
25. let Authorization = getUser () ;
26. if(Authorization && Authorization.token){
27. //如果当前缓存中存在用户令牌 后台接口的参数格式
28. if(!IsImgUpload(config)){
29. /\*\*\*\*在非图片上传场景下 给默认的上传格式\*\*\*\*/
30. config.headers['Content-Type'] = 'application/json'
31. }else{
32. config.headers['Content-Type'] = 'multipart/form-data'
33. }
34. config.headers['Authorization'] ='Bearer'+ ' ' + Authorization.token
35. }
36. // 让每个请求携带token-- ['X-Token']为自定义key
38. return config
39. },
40. error =**>** {
41. Promise.reject(error)
42. }
43. )
45. // response
46. service.interceptors.response.use(
47. /\*\*
48. \* 监控拦截 如果出现 异常 则直接终止请求链
49. \*/
50. response =**>** {
51. //LoadingManage.closeLoading() //关闭loading弹层
52. //直接返回返回体中的结构
53. return response.data ? response.data.data : null
54. },
55. error =**>** {
56. LoadingManage.closeLoading() //关闭loading弹层
57. let message = ''
58. let  code = error.response ? error.response.status : ''
59. switch(code){
60. case 400:
61. message = '请求参数错误'
62. break;
63. case 401:
64. message = 'token过期或未传'
65. break;
66. case 403:
67. message = '操作失败'
68. break;
69. case 404:
70. message = '手机号不正确'
71. break;
72. case 500:
73. message = '服务器异常'
74. break;
75. case 507:
76. message = '服务器数据库异常'
77. break;
78. default :
79. message = '处理异常'
80. }
81. // messagemessage = message + ':' +  error.response ? error.response.data.message : error.response.data.message
82. Message({
83. message,
84. type: 'warning',
85. onClose:code  == '401' ? () =**>**   Router.replace({path:'/login'}) : null
86. })
87. return new Promise(function(){}) //终止当前的promise链
88. }
89. )

92. export default service

## 运行项目

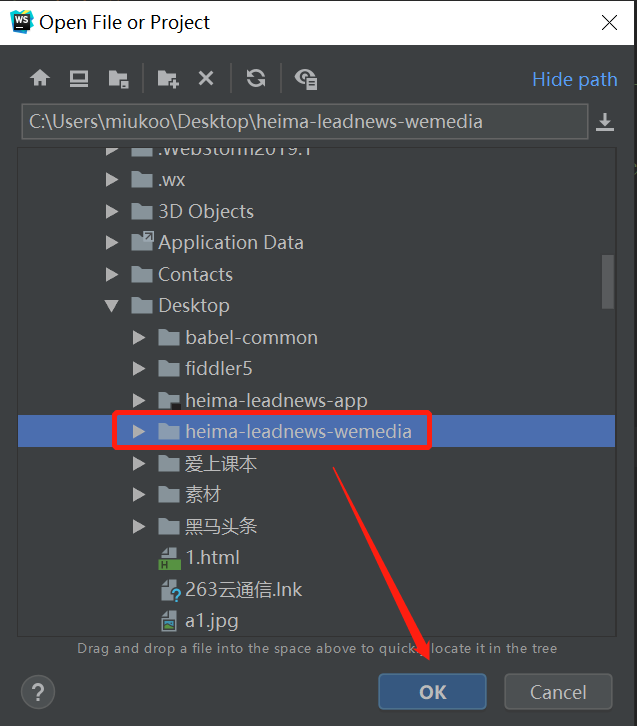
### 运行Web

在命令行运行 npm run dev 默认浏览器便会到开对应网页。



# 导入管理系统项目[heima-leadnews-admin]

* 解压[E01]heima-leadnews-admin.zip文件，并去掉[E01]
* 打开WebStorm，点击File->Open
* 在弹出的对话框中选择刚才解压的项目



* 打开终端，输入cnpm i安装依赖，安装完成后如无异常即导入成功

