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](#_Toc18510127)

[1. 概述 4](#_Toc18510128)

[2. 环境 4](#_Toc18510129)

[3. 创建APP项目[heima-leadnews-app] 4](#_Toc18510130)

[3.1 创建工程 4](#_Toc18510134)

[3.2 导入工程 5](#_Toc18510135)

[3.3 LESS集成 7](#_Toc18510136)

[3.3.1 安装LESS 7](#_Toc18510137)

[3.3.2 配置LESS 7](#_Toc18510138)

[3.4 项目结构 8](#_Toc18510139)

[3.4.1 项目结构 8](#_Toc18510140)

[3.4.2 项目依赖 8](#_Toc18510141)

[3.4.3 源码结构 10](#_Toc18510142)

[3.5 项目封装 10](#_Toc18510143)

[3.5.1 图标封装 10](#_Toc18510144)

[3.5.2 语言封装 12](#_Toc18510145)

[3.5.3 路由封装 13](#_Toc18510146)

[3.5.4 Reqeust封装 14](#_Toc18510147)

[3.5.5 配置封装 15](#_Toc18510148)

[3.5.6 存储封装 16](#_Toc18510149)

[3.6 安装AndroidStudio 18](#_Toc18510150)

[3.6.1 安装 18](#_Toc18510151)

[3.6.2 配置 19](#_Toc18510152)

[3.6.3 创建模拟器 20](#_Toc18510153)

[3.7 运行项目 22](#_Toc18510154)

[3.7.1 运行Web 22](#_Toc18510155)

[3.7.2 运行Android 23](#_Toc18510156)

[4 导入APP项目[heima-leadnews-app] 24](#_Toc18510157)

# 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

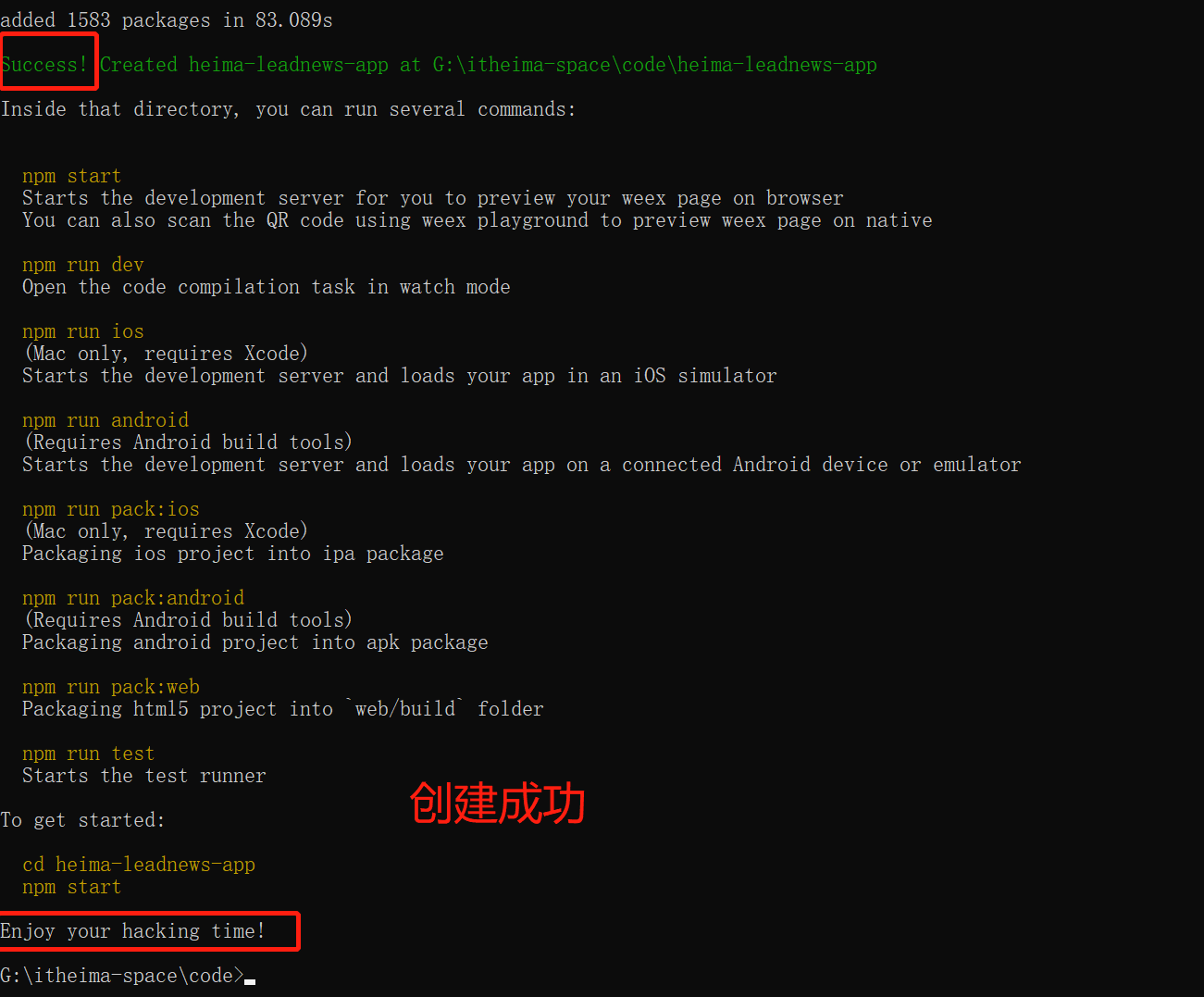
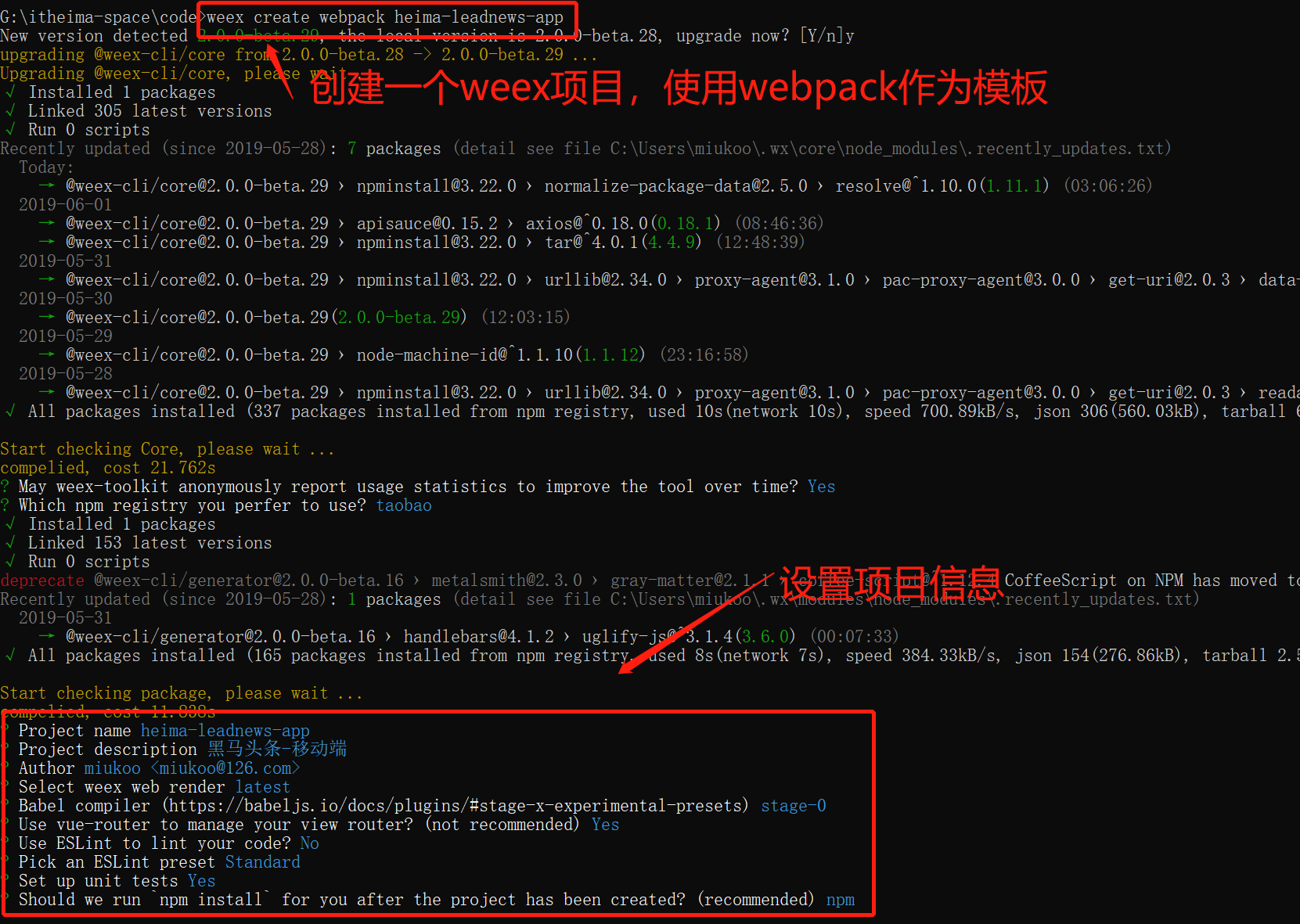
# 创建APP项目[heima-leadnews-app]

APP项目是黑马头条的APP端UE展示工程。



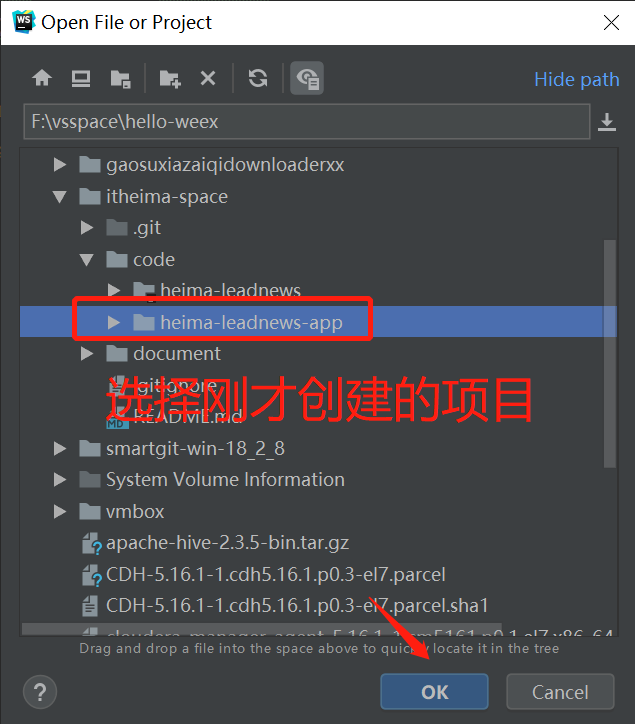
## 创建工程

* 进入项目存放路径
* 输入weex create webpack heima-leadnews-app命令
* 选择设置项目信息

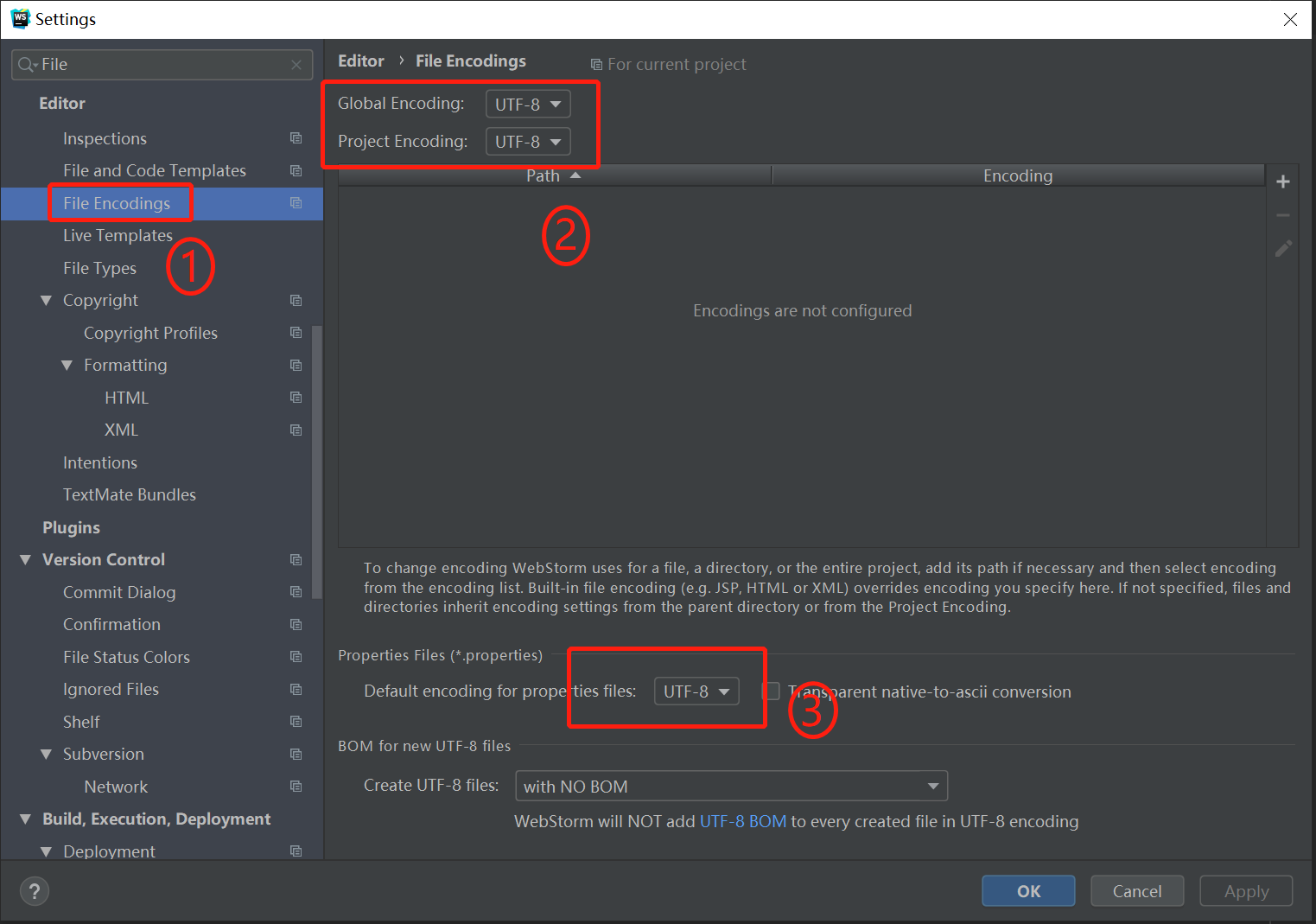


## 导入工程

* 打开WebStorm，File->Open



* 设置项目字符编码，File->Setting



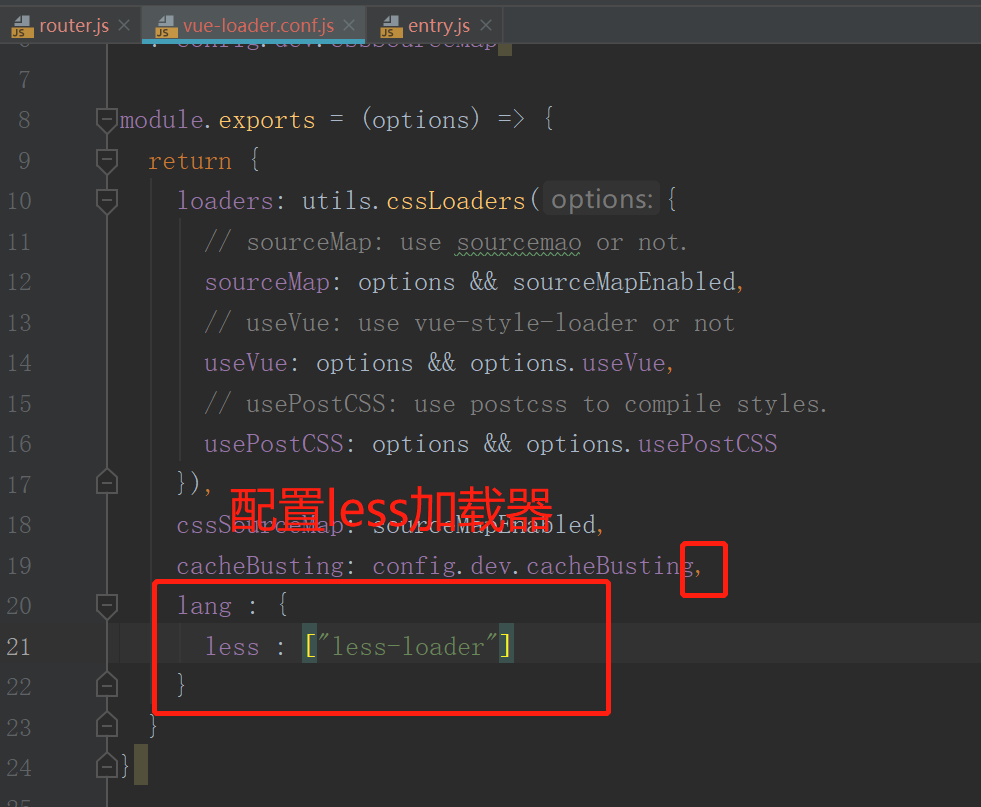
## LESS集成

### 安装LESS



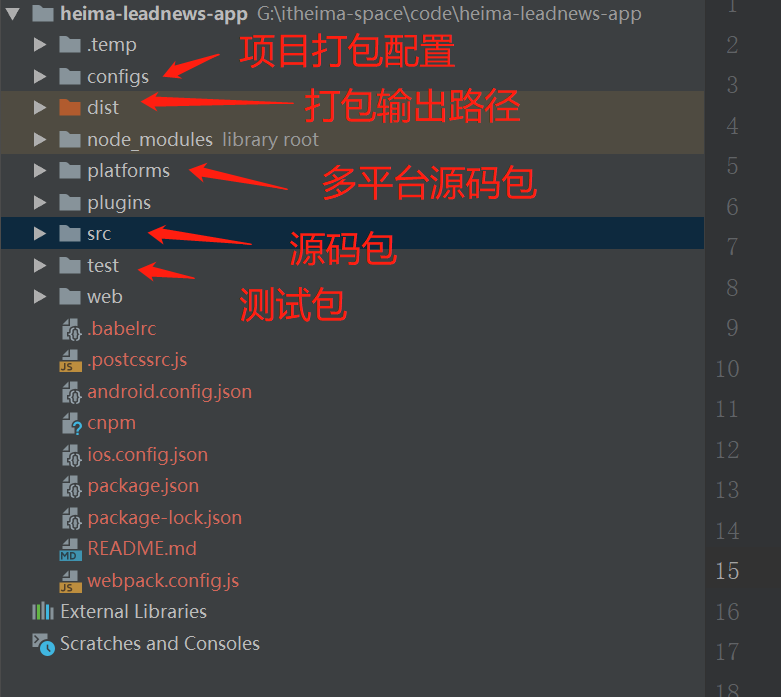
### 配置LESS

* 在vue-loader.conf.js中配置刚安装的less加载器。



## 项目结构

### 项目结构



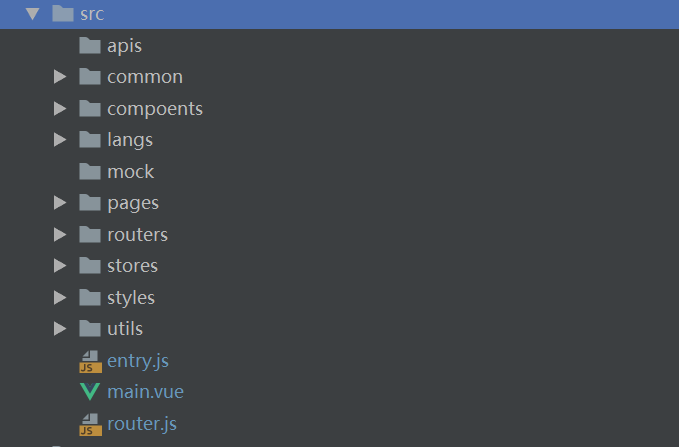
### 项目依赖

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. "phantom-limb": "0.0.1",
3. "vue": "^2.5.11",
4. "vue-router": "^3.0.1",
5. "weex-ui": "^0.6.14",
6. "weex-vue-render": "^1.0.17"
7. },
8. "devDependencies": {
9. "autoprefixer": "^8.0.0",
10. "babel-core": "^6.26.0",
11. "babel-loader": "^7.1.1",
12. "babel-preset-stage-0": "^6.24.1",
13. "chai": "^4.1.2",
14. "css-loader": "^0.28.8",
15. "extract-text-webpack-plugin": "^3.0.2",
16. "friendly-errors-webpack-plugin": "^1.6.1",
17. "fs-extra": "^5.0.0",
18. "glob": "^7.1.2",
19. "html-webpack-plugin": "^2.30.1",
20. "html-webpack-plugin-for-multihtml": "^2.30.2",
21. "ip": "^1.1.5",
22. "karma": "^1.7.1",
23. "karma-coverage": "^1.1.1",
24. "karma-mocha": "^1.3.0",
25. "karma-phantomjs-launcher": "^1.0.4",
26. "karma-phantomjs-shim": "^1.5.0",
27. "karma-sinon-chai": "^1.3.3",
28. "karma-sourcemap-loader": "^0.3.7",
29. "karma-spec-reporter": "0.0.31",
30. "karma-webpack": "^2.0.6",
31. "less": "^3.9.0",
32. "less-loader": "^5.0.0",
33. "mocha": "^5.2.0",
34. "node-notifier": "^5.1.2",
35. "portfinder": "^1.0.13",
36. "postcss-import": "^11.0.0",
37. "postcss-loader": "^2.0.9",
38. "postcss-plugin-weex": "^0.1.6",
39. "raw-loader": "^0.5.1",
40. "rimraf": "^2.6.2",
41. "script-ext-html-webpack-plugin": "^1.8.5",
42. "sinon": "^4.1.3",
43. "sinon-chai": "^2.14.0",
44. "vue-loader": "^12.2.0",
45. "vue-style-loader": "^3.0.3",
46. "vue-template-compiler": "^2.5.11",
47. "vuex": "^3.1.1",
48. "vuex-router-sync": "^5.0.0",
49. "webpack": "^3.10.0",
50. "webpack-dev-server": "^2.9.7",
51. "webpack-merge": "^4.1.0",
52. "webpack-uglify-parallel": "^0.1.4",
53. "weex-loader": "^0.7.2",
54. "weex-vue-precompiler": "^0.1.17",
55. "ws": "^1.1.0"
56. }

注： 1、weex-ui 是官方提供一套UI跨平台UI库，<https://alibaba.github.io/weex-ui>

### 源码结构



* apis ：按页面存放各个页面的api文件
* common ：存放公共配置和api
* compoents ：存放公用组件
* langs ：存放国际化语言包
* mock ：存放单元测试数据
* pages ：存放页面源码
* routers ：存放路由配置源码
* stores : 存放本地缓存源码
* styles ：存放样式源码
* utils ：存放通用工具类

## 项目封装

### 图标封装

#### Fontawesome介绍

Fontawesome是一套开源的图标字体库。由于字体图标渲染速度教快于图片，所以项目中图标将演示Fontawesome的使用。

#### Fontawesome API

在项目utils下新建font.js文件，内容为：

1. let utilFunc = {
2. initIconFont () {
3. let domModule = weex.requireModule('dom');
4. domModule.addRule('fontFace', {
5. 'fontFamily': "fontawesome",
6. 'src': "url('https://cdn.bootcss.com/font-awesome/4.7.0/fonts/fontawesome-webfont.ttf?v=4.7.0')"
7. });
8. }
9. };
11. export default utilFunc;

#### 引入Fontawesome

在index.vue文件created钩子方法中，调用initIconFont方法：

1. **<script>**
2. import util from "@/utils/font"
4. export default {
5. name: 'HeiMa-App',
6. created () {
7. util.initIconFont();
8. }
9. }
10. **</script>**

#### 使用Fontawesome

1. **<text** class="icon middle-icon"**>** **&#xf039;</text>**
2. **<style>**
3. .icon{
4. color: #ffffff;// 控制颜色
5. font-family: fontawesome;//指定为图标字体
6. font-size: 28px;//控制大小
7. }
8. **</style>**

注： 1、图标字符编码参考：<https://fontawesome.com/cheatsheet?from=io>

2、在JS中使用需要使用unicode编码，比如\uf039

### 语言封装

语言包的封装主要用于支持页面上静态内容的国际化支持。动态交互中的提示信息国际化将交由后端处理。

#### 中文包

在langs目录下增加zh.js，存储中文语言包。

1. const lang = {
2. load\_new\_text:'正在与火星通信......',
3. load\_more\_text:'正在加载更多......'
4. }
5. export default lang

#### 语言路由

在langs目录下增加lang.js，用于由于语言包的选择。

1. import zh from "./zh"
3. const lang = zh
5. export default lang

注：项目默认使用zh语言包，后期再对该路由进行功能扩展。

#### 全局设置

在entry.js文件中导入并设置为全局变量：

1. import lang from '@/langs/lang'
2. Vue.prototype.$lang = lang

使用示例：

1. this.$lang.load\_new\_text

### 路由封装

* 在routers文件下新增index.js文件：

1. import Home from '@/pages/index/index'
3. let routes = [
4. {
5. path: '/',
6. name: 'index',
7. component: Home
8. }
9. ]
11. let concat = (router) =**>** {
12. routes = routes.concat(router)
13. }
15. export default  routes;

* 在src/router.js文件中导入路由：

1. import Router from 'vue-router'
2. import routes from '@/routers/index'
4. Vue.use(Router)
6. export const router = new Router({
7. routes:routes
8. });

### 导航封装

创建文件src\utils\navigator.js

1. function Navigator(){
2. **this**.navigator = []
3. **this**.count = 0
4. **this**.max = 9
5. **this**.temp={}
6. }
7. Navigator.prototype= {
8. push : function(to,form,next) {
9. **if** (**this**.temp['path'] != to['path']) {
10. **this**.navigator[**this**.count] = form
11. **this**.count++;
12. **if** (**this**.count > **this**.max) **this**.count = 0;
13. }**else**{
14. **this**.temp={}
15. }
16. next()
17. },
18. back : function(){
19. **this**.count=**this**.count-1;
20. **if**(**this**.count<0)**this**.count=**this**.max;
21. **this**.temp=**this**.navigator[**this**.count]
22. **return** **this**.temp
23. }
24. }
26. export **default** **new** Navigator();

并在src/router.js中增加以下代码，注册导航器生效

1. // 注册导航器属性
2. router.$navigator = navigator
3. // 注册返回方法
4. router.back = function(){
5. let to = **this**.$navigator.back()
6. **if**(to){
7. **this**.push(to)
8. }
9. }
10. // 路由之前记录路由处理
11. router.beforeResolve((to, from, next) => {
12. router.$navigator.push(to, from, next)
13. })

### Reqeust封装

Request的封装主要为项目提供网络支持。

#### request开发

在common文件中增加request.js

1. var querystring=require("querystring");
2. function Request() {
3. var stream;
4. }
5. Request.prototype={
6. \_\_check : function(){
7. if(!this.stream){
8. this.stream = weex.requireModule("stream");
9. }
10. return this.stream;
11. },
12. post : function(path,body){
13. let stream = this.\_\_check();
14. return new Promise((resolve, reject) =**>** {
15. stream.fetch({
16. method: 'POST',
17. url: path,
18. type: 'json',
19. body:body
20. }, (response) =**>** {
21. if (response.status == 200) {
22. resolve(response.data)
23. }
24. else {
25. reject(response)
26. }
27. }, () =**>** {})
28. })
29. },
30. get : function(path,parms){
31. let stream = this.\_\_check();
32. if(parms){
33. let tmp = querystring.stringify(parms)
34. if(path.indexOf("?")==-1){
35. tmp="?"+tmp;
36. }else{
37. tmp="&"+tmp;
38. }
39. path+=tmp;
40. }
41. return new Promise((resolve, reject) =**>** {
42. stream.fetch({
43. method: 'GET',
44. url: path,
45. type: 'json'
46. }, (response) =**>** {
47. if (response.status == 200) {
48. resolve(response.data)
49. }
50. else {
51. reject(response)
52. }
53. }, () =**>** {})
54. })
55. }
56. }
57. export  default new Request()

#### 全局设置

在entry.js文件中把reuqest作为全局设置：

1. import request from '@/common/request'
2. Vue.prototype.$request = request

### 配置封装

#### 配置开发

在common文件中增加config.js

1. const  config = {
3. urls:{
4. baseUrl:'/toutiao',
5. load:'load',
6. loadmore:'loadmore',
7. loadnew:'loadnew',
8. // 解决多平台问题
9. getBase : function(){
10. if(weex.config.env.platform=='Web'){
11. return config.urls.baseUrl;
12. }else{
13. return "http://m.toutiao.com"
14. }
15. },
16. get:function(name){
17. let tmp = config.urls[name];
18. if(tmp)
19. return config.urls.getBase()+"/"+tmp;
20. else
21. return config.urls.getBase()+"/"+name;
22. }
23. }
25. }
26. export default config

#### 全局设置

在entry.js文件中把apis作为全局设置：

1. import conf from '@/common/conf'
2. Vue.prototype.$config = conf

### 存储封装

#### 存储开发

在stores文件中增加store.js

1. function Cache(){
2. var storage;
3. var TOKEN\_KEY = "TOKEN\_KEY"
4. }
5. Cache.prototype={
6. setToken : function(token){
7. return this.\_\_setItem(this.TOKEN\_KEY,token);
8. },
9. getToken : function(){
10. return this.\_\_getItem(this.TOKEN\_KEY);
11. },
12. clearToken : function(){
13. return this.\_\_removeItem(this.TOKEN\_KEY);
14. },
15. \_\_check : function(){
16. if(!this.storage){
17. this.storage = weex.requireModule("storage");
18. }
19. return this.storage;
20. },
21. \_\_setItem : function(key,value){
22. let storage = this.\_\_check();
23. return new Promise((resolve, reject)=**>**{
24. storage.setItem(key, value, function(e){
25. if(e.result=='success'){
26. resolve(true)
27. }else{
28. reject(false)
29. }
30. });
31. });
32. },
33. \_\_getItem : function(key){
34. let storage = this.\_\_check();
35. return new Promise((resolve, reject)=**>**{
36. storage.getItem(key, function(e){
37. if(e.result=='success'){
38. resolve(e.data)
39. }else{
40. reject(e)
41. }
42. });
43. });
44. },
45. \_\_removeItem : function(key){
46. let storage = this.\_\_check();
47. return new Promise((resolve, reject)=**>**{
48. storage.removeItem(key, function(e){
49. if(e.result=='success'){
50. resolve(true)
51. }else{
52. reject(false)
53. }
54. });
55. });
56. }
57. }
58. const  che = new Cache();
59. export  default che

#### 全局设置

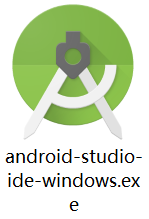
在entry.js文件中把apis作为全局设置：

1. import store from '@/stores/store'
2. Vue.prototype.$store = store

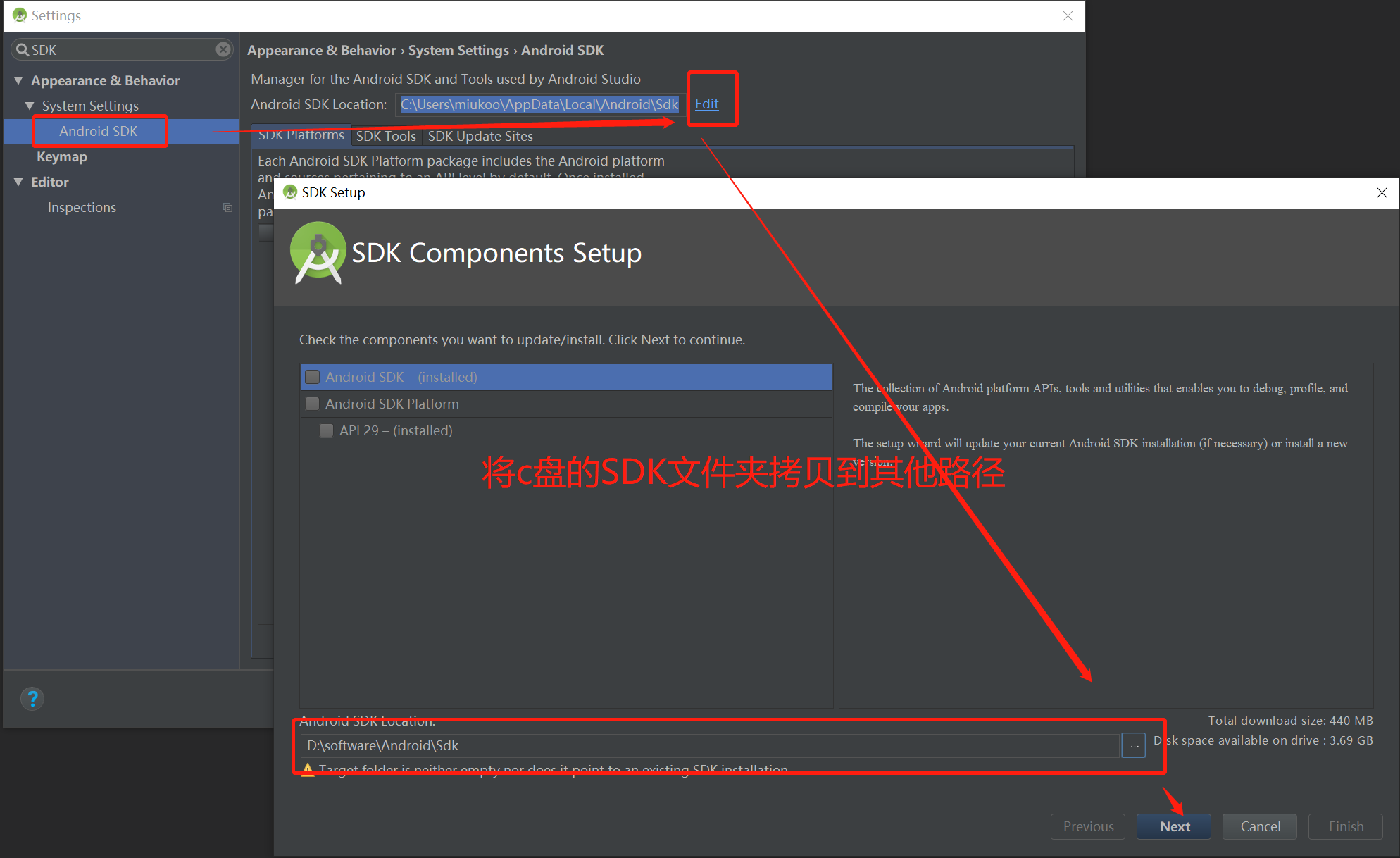
## 安装AndroidStudio

### 安装

* 安装之前确保JDK环境以安装并配置好
* 下载并双击安装



### 配置

* 运行AndroidStudio，导入‘heima-leadnews-app\platforms\android’项目；
* 导入项目模块app
* 配置下载代理【可选步骤】
  + 打开网站http://ping.chinaz.com/dl.google.com选择一个最快的IP地址
  + 在hosts文件中配置xxx.xxx.xx.xx dl.google.com
* 移动SDK目录文件夹【可选步骤】  
  
* 配置ANDROID\_HOME

配置环境变量ANDROID\_HOME=D:\software\Android\Sdk

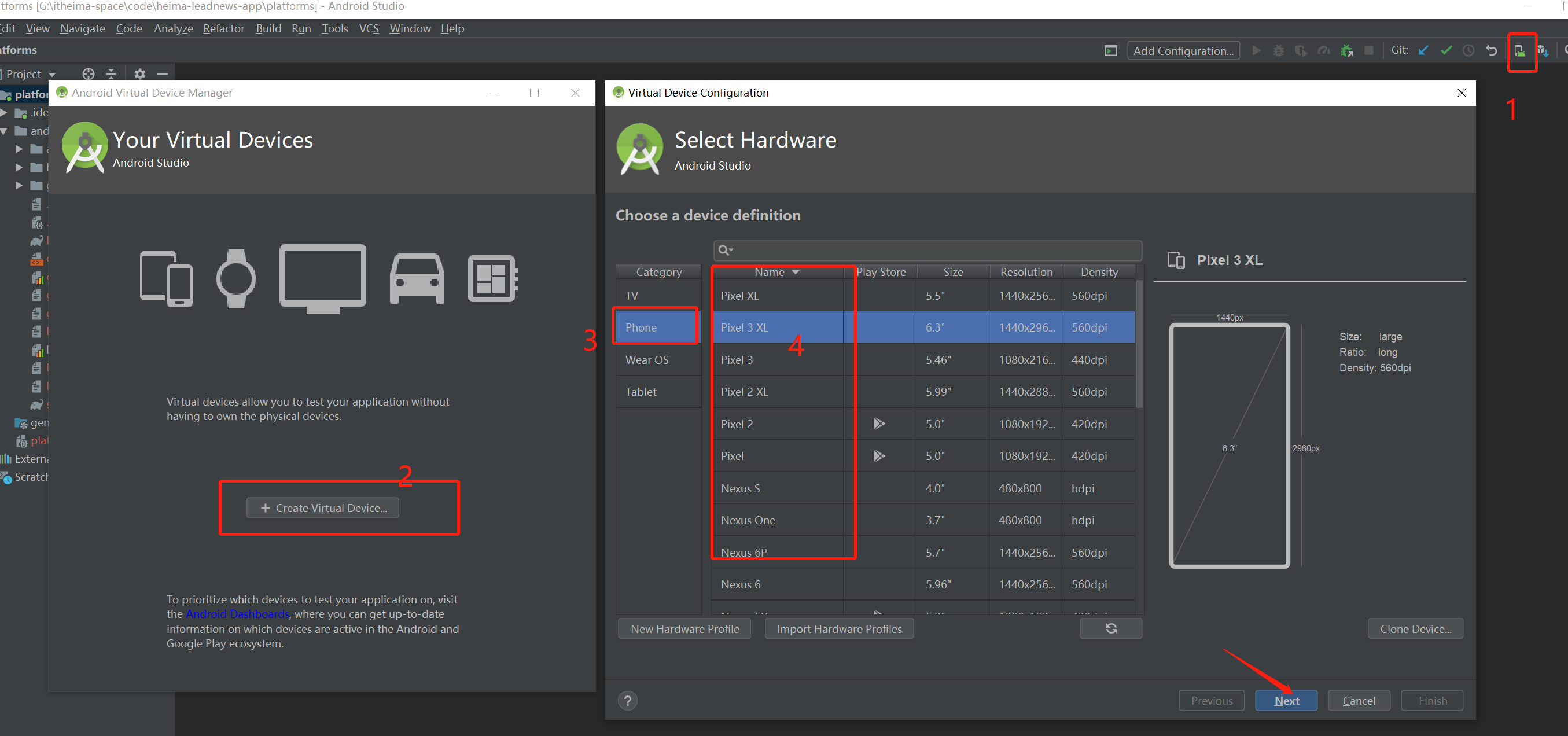
* 配置ANDROID\_SDK\_HOME【可选步骤】

配置环境变量ANDROID\_SDK\_HOME=D:\

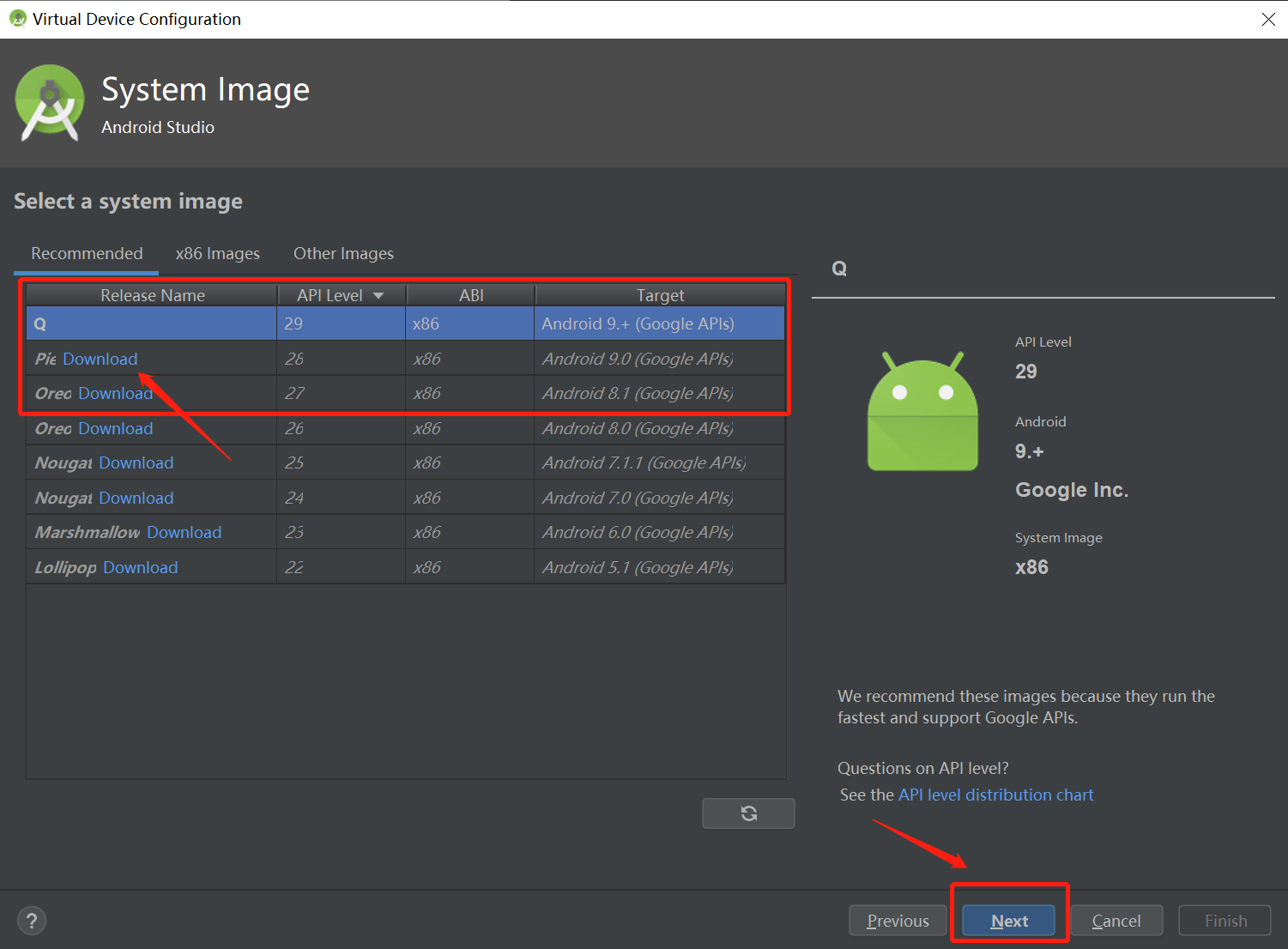
* 配置Path
  + %ANDROID\_HOME%\platform-tools
  + %ANDROID\_HOME%\tools

### 创建模拟器

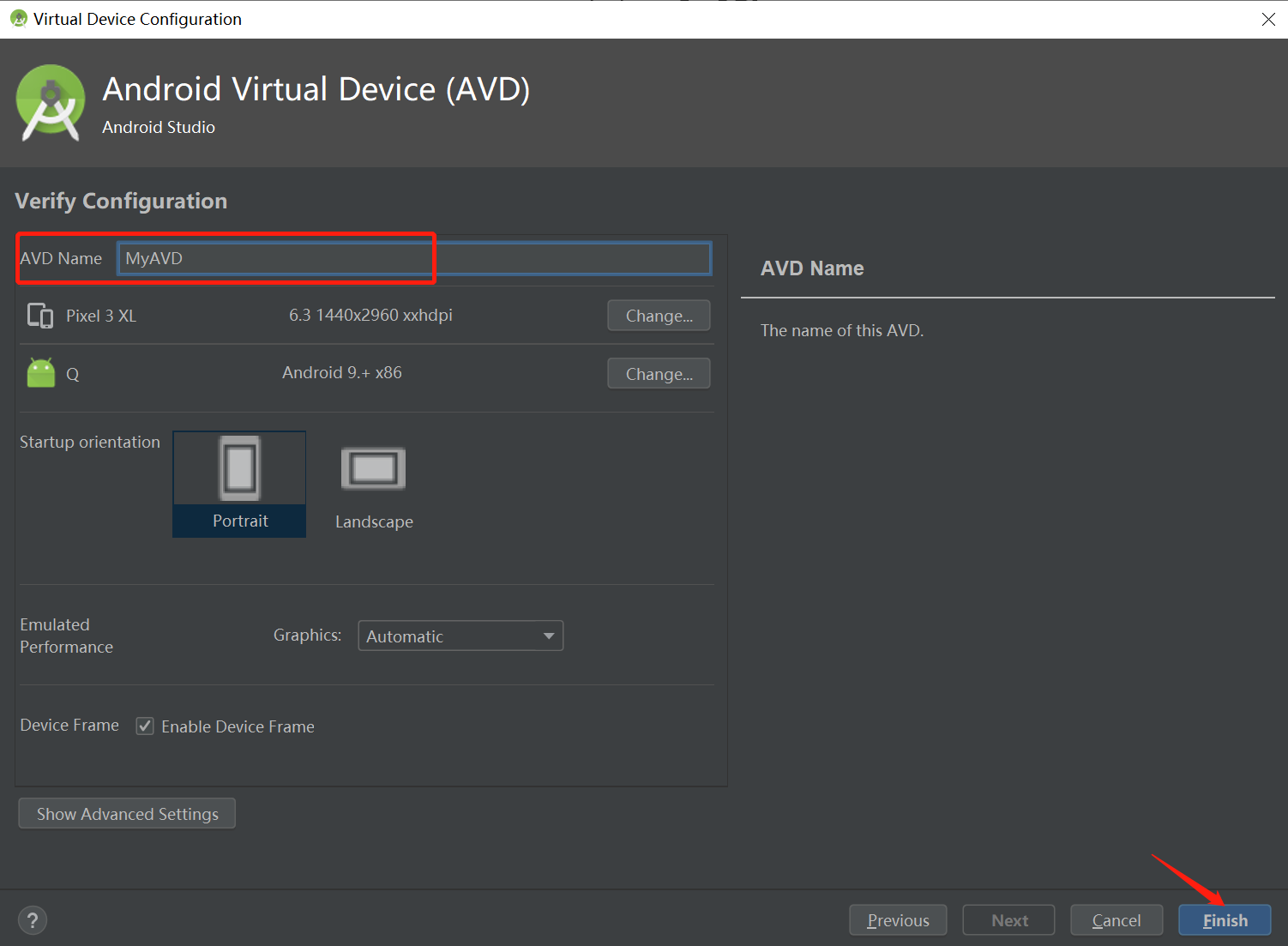
* 选择模拟器



* 下载模拟器



* 填写AVD信息



## 运行项目

### 运行Web

在命令行运行 npm run serve 默认浏览器便会到开对应网页。

也可使用Playground扫描页面上的二维码，在手机上体验网页。

**注意**：项目默认选用无线网络IP运行，如需修改，可以修改config/config.js和config/ webpack.dev.conf.js文件中的下列代码：

const ip = require('ip').address('WLAN');

// const ip = ‘192.168.0.8’;



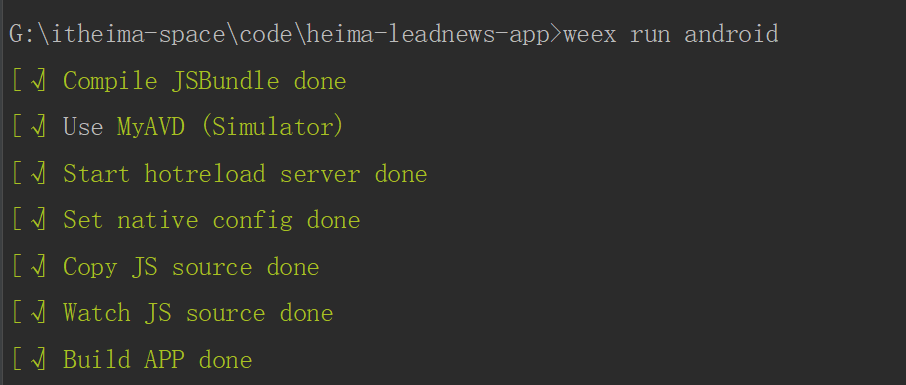
注：手机扫描时，手机和电脑应该在同一局域网内。

### 运行Android

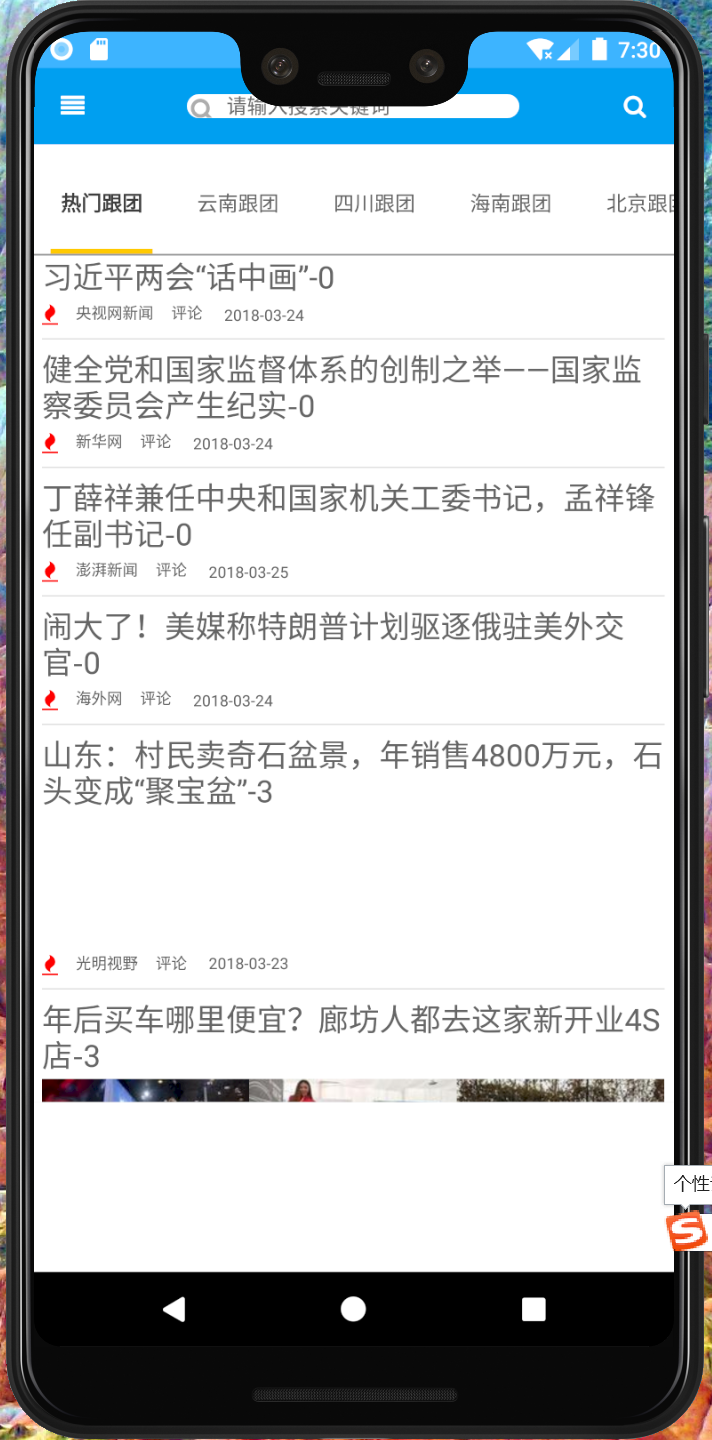
在Android模拟器上运行之前，需确保Android Studio环境正确安装，以及模拟器已创建。

准备就绪后，在项目目录下命令行执行weex run android，即可运行。

命令行截图：

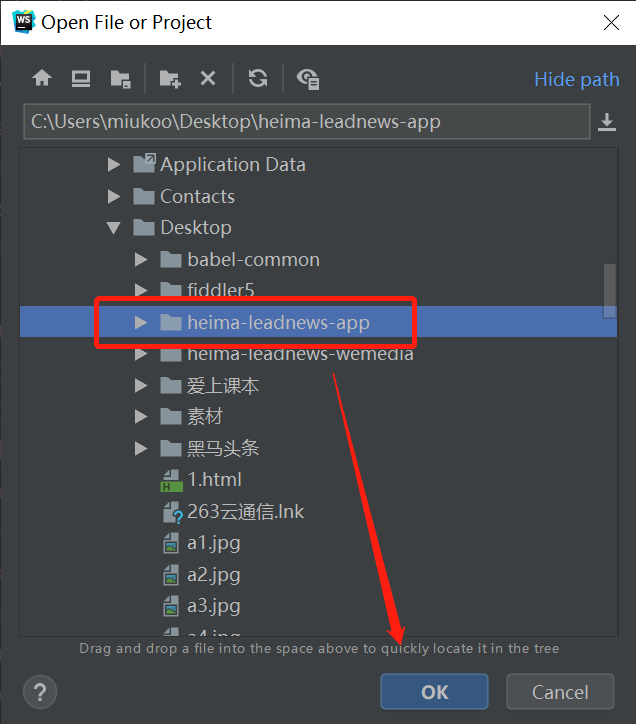


模拟器效果：



# 导入APP项目[heima-leadnews-app]

* 解压[E01]heima-leadnews-app.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

