# vue-cli

# Vue-cli的使用

https://cn.vuejs.org/

## 一、vue脚手架搭建

### 1.进入一个目录，创建项目

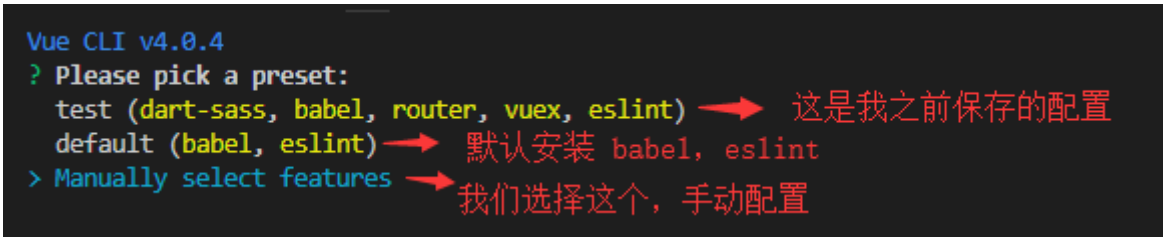
```
PS E:\> vue create project-one
```

对应命令：

```
vue create project-one
```

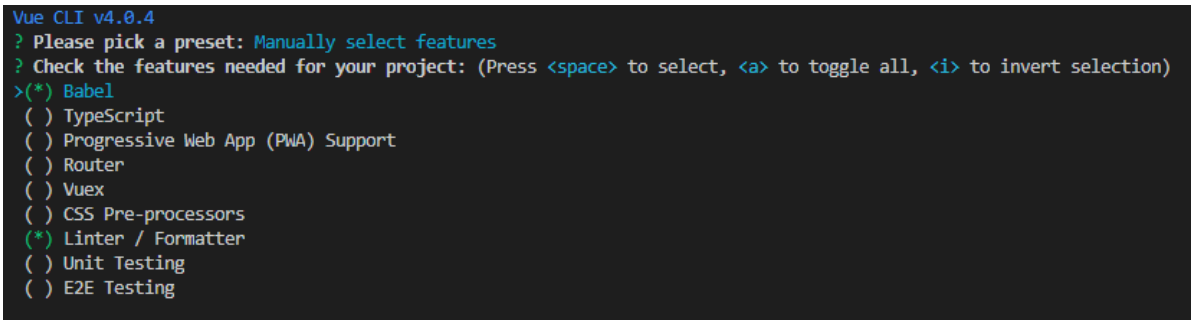
### 2.我们这里选择手动配置

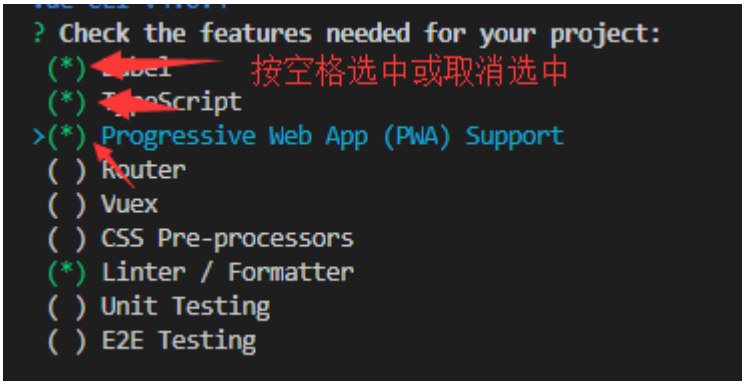
按 ↓ 选择“Manually select features”，再按 Enter

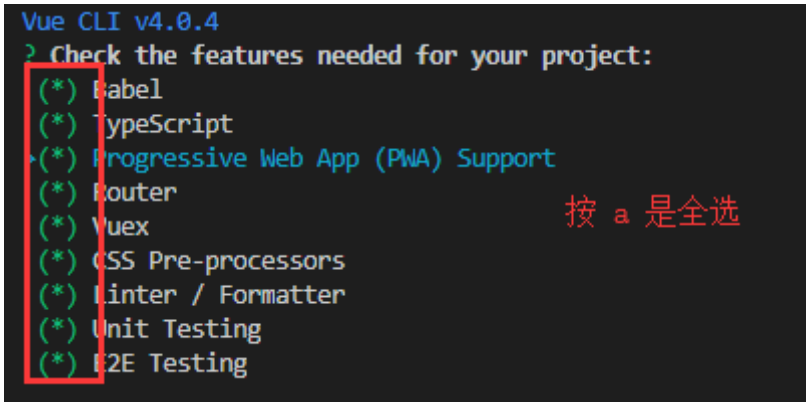


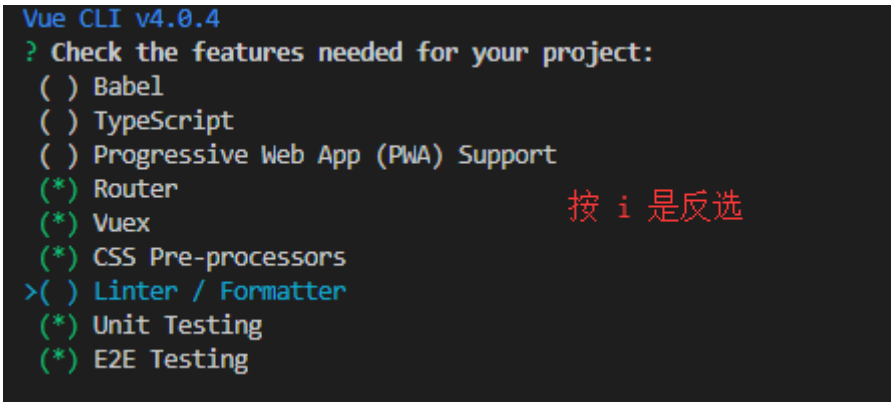

### 3.选择你需要的配置项

通过↑ ↓ 箭头选择你要配置的项，按 空格 是选中，按 a 是全选，按 i 是反选。具体每个配置项表示什么意思在下面会有说明。

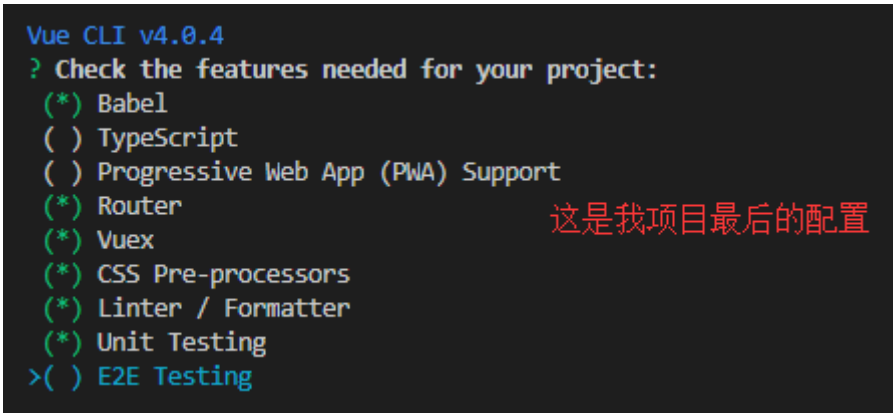




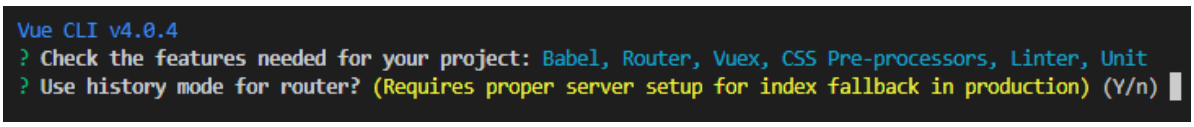




```
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection)
>( ) Babel //转码器，可以将ES6代码转为ES5代码，从而在现有环境执行。
( ) TypeScript// TypeScript是一个JavaScript（后缀.js）的超集（后缀.ts）包含并扩展了 JavaScript 的语法，需要被编译输出为 JavaScript在浏览器运行
( ) Progressive Web App (PWA) Support// 渐进式Web应用程序
( ) Router // vue-router（vue路由）
( ) Vuex // vuex（vue的状态管理模式）
( ) CSS Pre-processors // CSS 预处理器（如：less、sass）
( ) Linter / Formatter // 代码风格检查和格式化（如：ESlint）
( ) Unit Testing // 单元测试（unit tests）
( ) E2E Testing // e2e（end to end） 测试
```




选完之后按 Enter。分别选择每个对应功能的具体包。选你擅长的，没有擅长的，就选使用广的，哈哈，方便咨询别人。

## 3.1 选择是否使用history router

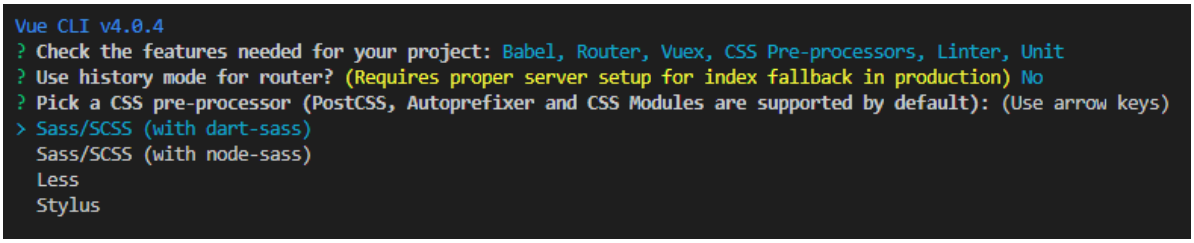



Vue-Router 利用了浏览器自身的hash 模式和 history 模式的特性来实现前端路由（通过调用浏览器提供的接口）。

- 我这里建议选n。这样打包出来丢到服务器上可以直接使用了，后期要用的话，也可以自己再开起来。
- 选yes的话需要服务器那边再进行设置。

  Use history mode for router? (Requires proper server setup for index fallback in production)

## 3.2 选择css 预处理器

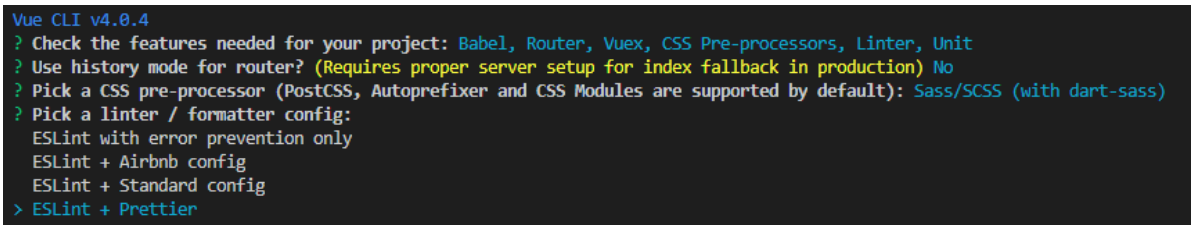



我选择的是Sass/Scss(with dart-sass)

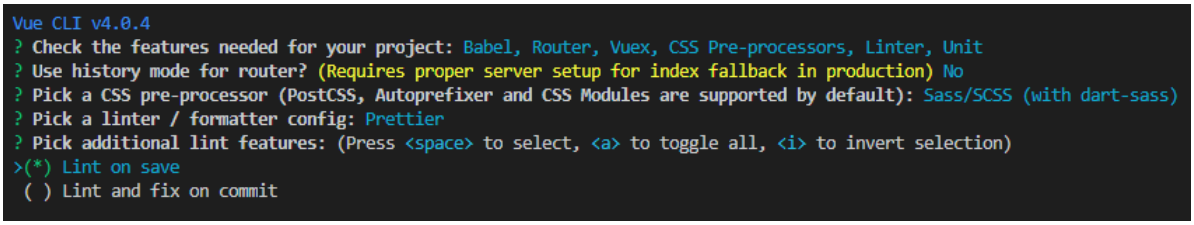
node-sass是自动编译实时的，dart-sass需要保存后才会生效。sass 官方目前主力推dart-sass 最新的特性都会在这个上面先实现。（该回答参考http://www.imooc.com/qadetail/318730）

## 3.3 选择Eslint代码验证规则

提供一个插件化的javascript代码检测工具，ESLint + Prettier //使用较多



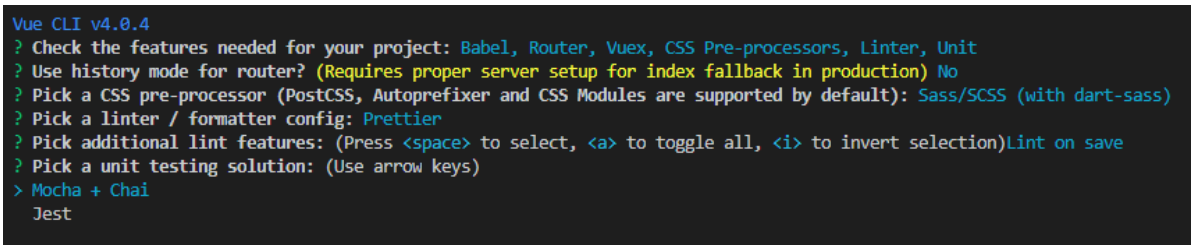
## 3.4 选择什么时候进行代码规则检测




```
( ) Lint on save // 保存就检测
( ) Lint and fix on commit // fix和commit时候检查
```

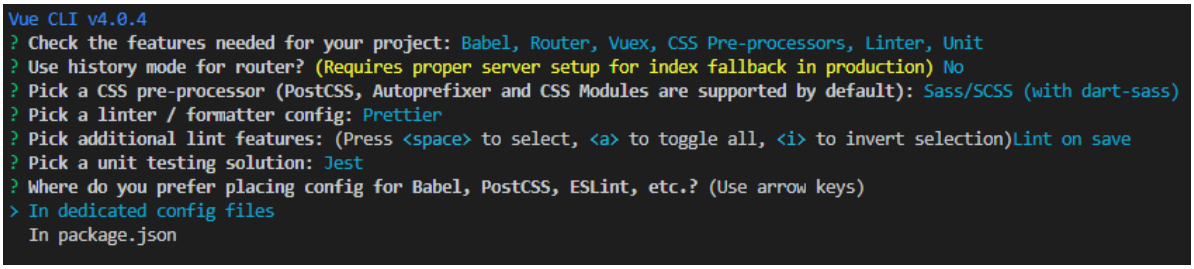
建议选择保存就检测，等到commit的时候，问题可能都已经积累很多了。

## 3.5 选择单元测试



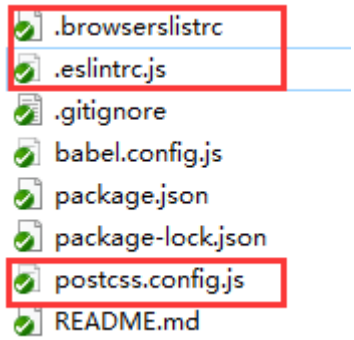

```
> Mocha + Chai //mocha灵活,只提供简单的测试结构，如果需要其他功能需要添加其他库/插件完成。必须在全局环境中安装
Jest //安装配置简单，容易上手。内置Istanbul，可以查看到测试覆盖率，相较于Mocha:配置简洁、测试代码简洁、易于和babel集成、内置丰富的expect
```

## 3.6 选择如何存放配置




```
> In dedicated config files // 独立文件放置
  In package.json // 放package.json里
```

如果是选择 独立文件放置，项目会有单独如下图所示的几件文件。




## 3.7 是否保存当前配置

```
Vue CLI v4.0.4
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter, Unit
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Sass/SCSS (with dart-sass)
? Pick a linter / formatter config: Prettier
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)Lint on save
? Pick a unit testing solution: Jest
? Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? In package.json
? Save this as a preset for future projects? (y/N)
```

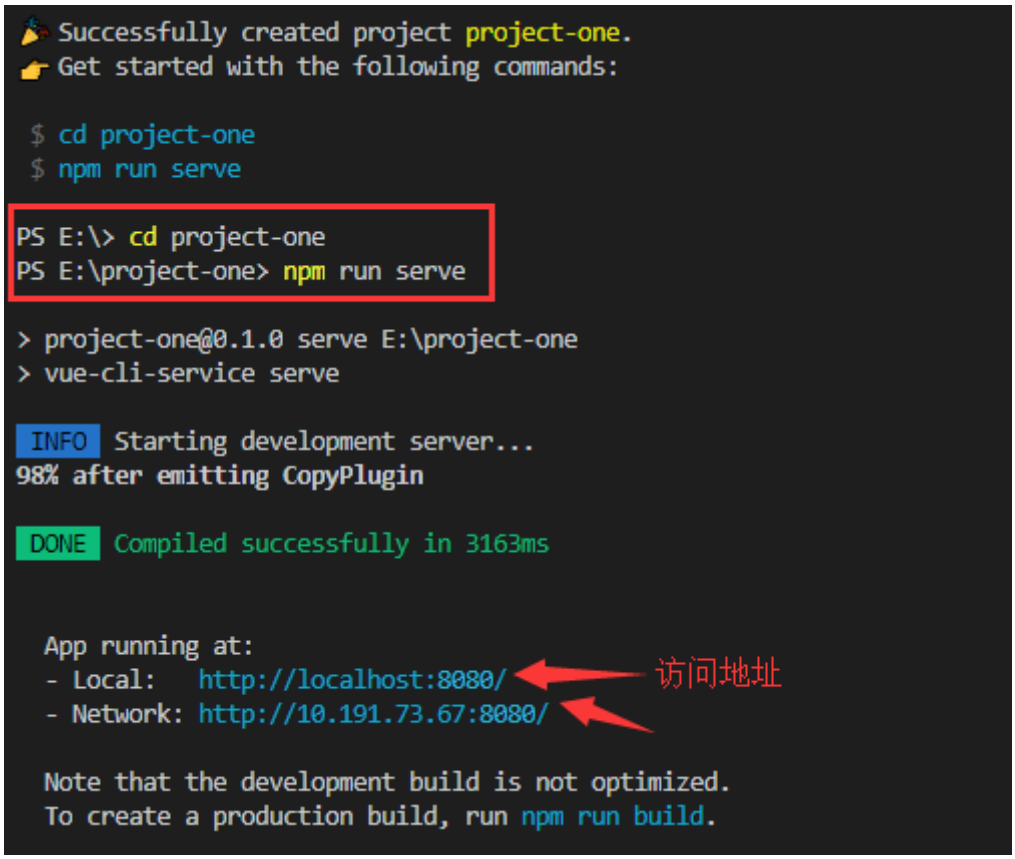
键入N不记录，如果键入Y需要输入保存名字，如第2步所看到的我保存的名字为test。

## 4.等待创建项目

```
✨ Creating project in E:\project-one.
🗃  Initializing git repository...
⚙  Installing CLI plugins. This might take a while...


> yorkie@2.0.0 install E:\project-one\node_modules\yorkie
> node bin/install.js

setting up Git hooks
done


> core-js@2.6.10 postinstall E:\project-one\node_modules\babel-runtime\node_modules\core-js
> node postinstall || echo "ignore"


> core-js@3.3.2 postinstall E:\project-one\node_modules\core-js
> node postinstall || echo "ignore"


> core-js-pure@3.3.2 postinstall E:\project-one\node_modules\core-js-pure
> node postinstall || echo "ignore"

added 1379 packages from 1006 contributors in 82.524s
🚀  Invoking generators...
📦  Installing additional dependencies...

added 62 packages from 50 contributors and updated 1 package in 16.127s
⚓  Running completion hooks...

📄  Generating README.md...

🎉  Successfully created project project-one.
👉  Get started with the following commands:

 $ cd project-one
 $ npm run serve
```

创建项目成功

### 5.执行它给出的命令，可以直接访问了




## 二、项目目录介绍

1. node_modules 存放第三方依赖
2. public 存放静态文件夹
   2.1 favicon.ico 是网站图标

2.2 index.html 页面入口文件

3. src 存放源码文件夹
   3.1 assets 存放图片，css

3.2 components 存放组件

3.3 views 存放视图组件

3.4 router 存放路由配置

3.5 store 存放 vuex 配置

3.6 plugins 存放插件配置

3.7 App.vue 根组件

3.8 main.js 入口js

4.browserslistrc 配置使用CSS兼容性插件的使用范围

5. .eslintrc.js ESlint配置
6. .gitignore 指定文件无需提交到git上
7. balel.config.js 使用一些预设
8. package.json 项目描述及依赖
9. package-lock.json 版本管理使用的文件
10. .editorco 配置文件,规范编辑器的配置

11. 11 vue.config.js webpack配置文件

# 三、vue路由基础复习

1.路由跳转的第一种方式及传参

```
<router-link to="/">Home</router-link> |
<router-link to="/about?id=1">About</router-link> |
<router-link to="/my/:id">My</router-link>
```

2.路由跳转的第二种方式及传参

```
//第一种路由跳转方式  及 传参
this.$router.push({path: "/detail", query:{id: 1}})
```

3.路由跳转的第三种方式及传参

```
//第二种路由跳转方式
this.$router.push({name: "Detail", params:{id: this.id}})
```

4.路由配置

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'
import About from '../views/About'

Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'Home',
    meta: {
      title: "首页"
    },
    component: Home
  },
  {
    path: '/about',
    name: 'About',
    meta: {
      title: "关于"
    },
    component: About
  },
  {
    path: "/my",
    name: "My",
    meta: {
      title: "我的"
    },
    component: () => import('../views/My.vue')
```

```
  },
  {
    path: "/detail",
    name: "Detail",
    meta: {
      title: "详情"
    },
    component: () => import('../views/Detail.vue')
  }
]

const router = new VueRouter({
  routes
})

export default router
```

# 四、vue组件基础复习

1.父组件给子组件传参

```
//父组件 通过绑定v-bind:属性
<template>
<div class="about">
    <Child :msg2="msg" />
</div>
</template>
<script>
import Child from "@/components/Child.vue"
export default {
  // 定义属性
  name:'About',
  components: {
    Child
  },
  data() {
    return {
      msg: "我是父组件的数据"
    }
  }
}
</script>
//子组件使用props接收
<template>
  <div class="child">
      {{ msg2 }}
  </div>
</template>
<script>
export default {
  // 定义属性
  name:'Child',
//   props: {
//       msg: String
```

```
//   },
  props: ["msg2"],
  data() {
    return {
      childData: "子组件的数据"
    }
  }
}
</script>
```

2.子组件给父组件传参

```
//子组件 子组件给父组件传参 通过$emit映射一个自定义方法
<template>
  <div class="child">
      <button @click="toFatherData" >给父组件传递参数</button>
  </div>
</template>
<script>
export default {
  // 定义属性
  name:'Child',
  data() {
    return {
      childData: "子组件的数据"
    }
  },
  methods: {
    //子组件给父组件传参 通过$emit映射一个自定义方法
    toFatherData(){
        this.$emit("fun", this.childData)
    }
  }
}
</script>
//父组件 通过v-on绑定子组件的映射方法接收
<template>
  <div class="about">
    <Child @fun="childMethod" />
  </div>
</template>
<script>
import Child from "@/components/Child.vue"

export default {
  // 定义属性
  name:'About',
  components: {
    Child
  },
  data() {
    return {
    }
  },
  //所有函数方法
```

```
  methods: {
    //子组件映射的接收方法
    childMethod(value) {
      console.log(value)
    }
  }
}
</script>
```

# 五、vuex的基本使用

- state：存储状态（变量）
- getters：对数据获取之前的再次编译，可以理解为state的计算属性。我们在组件中使用$sotre.getters.fun()
- mutations：修改状态，并且是同步的。在组件中使用$store.commit('',params)。这个和我们组件中的自定义事件类似。
- actions：异步操作。在组件中使用是$store.dispath('')
- modules：store的子模块，为了开发大型项目，方便状态管理而使用的。

```
export default new Vuex.Store({
  state: { //存储共享状态的数据源 可以在任何组件访问该状态数据
    count: 0
  },
  mutations: {//修改状态数据的同步方法
     mutationsAddCount(state, n = 0) {
        return (state.count += n)
    },
    mutationsReduceCount(state, n = 0) {
        return (state.count -= n)
    }
  },
  actions: {//修改状态数据的异步方法
      actionsAddCount(context, n = 0) {
        console.log(context)
        return context.commit('mutationsAddCount', n)
      },
      actionsReduceCount({ commit }, n = 0) {
        return commit('mutationsReduceCount', n)
      }
  },
    getters:{//计算属性，用于对store中的数据进行处理形成新的数据,不会修改数据源
        getterCount(state) {
         return "当前的数据是" + state.count
        }
    },
  modules: {//方便管理。配置公共模块
  }
})
```

```
<h3>{{$store.state.count}}</h3>
```

```
<h3>{{$sotre.getters.getterCount}}</h3>
methods: {
    handleAddClick(n){
      this.$store.commit('mutationsAddCount',n);
    },
    handleReduceClick(n){
      this.$store.commit('mutationsReduceCount',n);
    }
  }
methods:{
handleActionsAdd(n){
      this.$store.dispatch('actionsAddCount',n)
    },
    handleActionsReduce(n){
      this.$store.dispatch('actionsReduceCount',n)
    }
}
```

# 商城页面还原

## 1.配置Vant-ui

https://vant-contrib.gitee.io/vant/#/zh-CN/

## 2.还原页面

## 3.编写商城业务逻辑

## 4.项目打包

# mock数据的使用

1. 官方文档
   本文默认你对Mock.js有一定的了解，并且阅读过Mock.js的官方文档，因此本文就不在赘述关于Mock.js的基础知识。

Mock.js官方文档地址：http://mockjs.com/

Mock.js是一门生成随机数据，为前端Ajax请求提供数据接口的前端框架技术，主要是为了适应前后端独立开发，为前端开发提供虚拟的后台数据接口，供前端调试开发。

1.添加Mock.js

## npm i mockjs -S

2.封装Mock.js
先添加一个mock.js文件

mock.js代码如下

```
import Mock from 'mockjs' // 引入mockjs

import data from './data.json'

Mock.mock('/data', 'get', data )
```

## 打包生成安卓apk

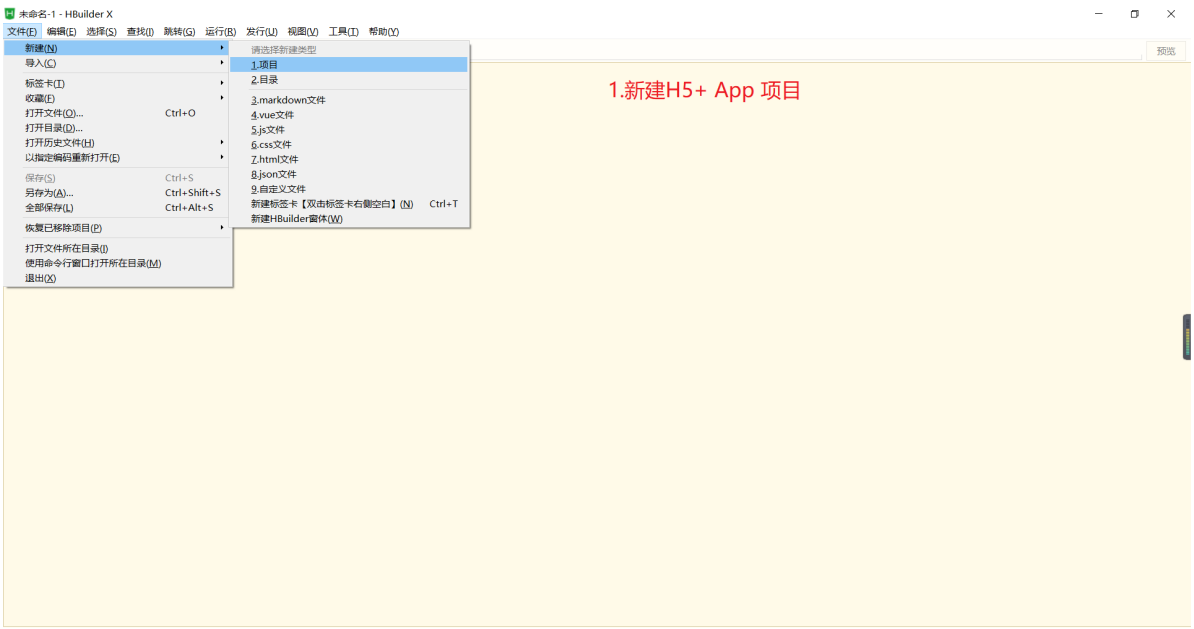
# 用HBuilderX 打包 vue 项目 为 App 的步骤

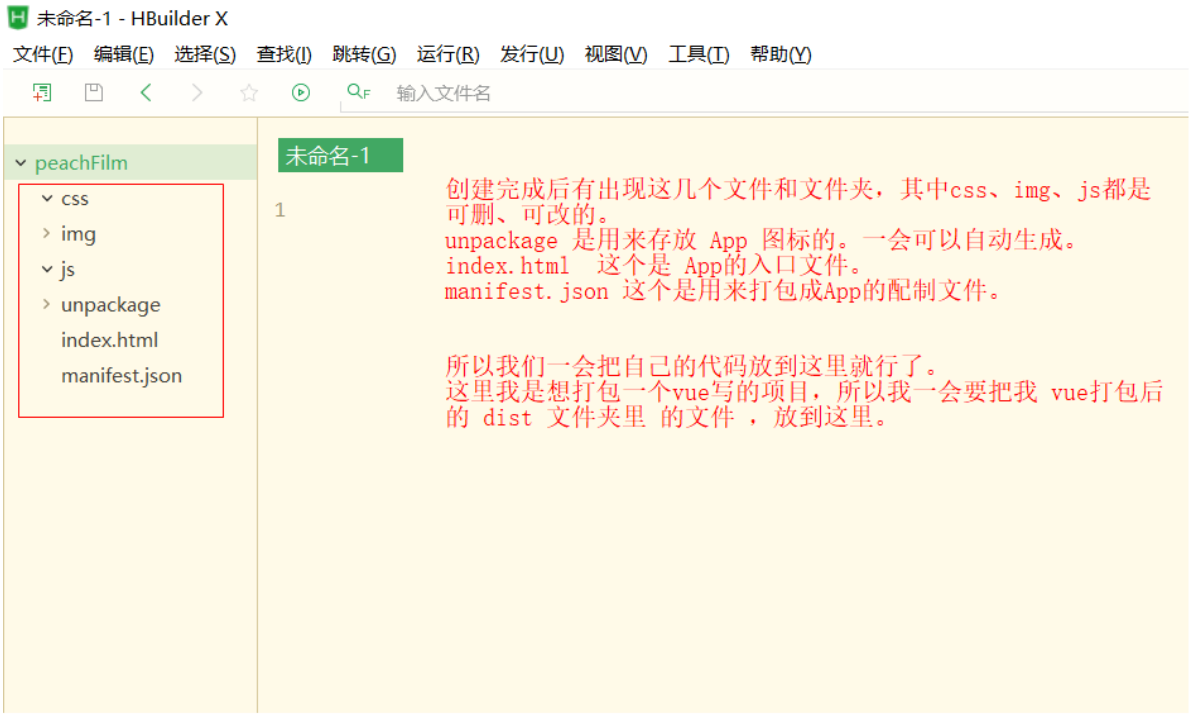
首先打包你的 vue 项目 生成 dist 文件夹，教程请移步 https://www.cnblogs.com/taohuaya/p/10256670.html

看完上面的教程，请确保 你是 将：

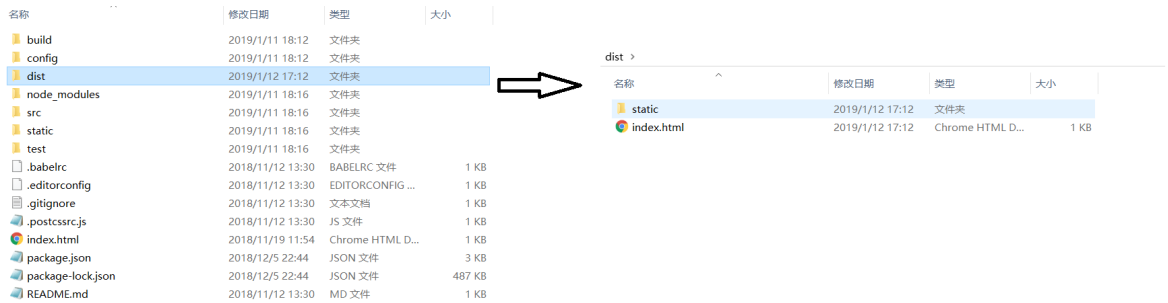
项目目录下的`config`文件夹里的`index.js`文件中,将`build`对象下的`assetsPublicPath`中的“/”，改为“./”后，打包生成的 dist 文件。

开始使用 HBuiderX 打包。（工具下载地址：http://www.dcloud.io/）



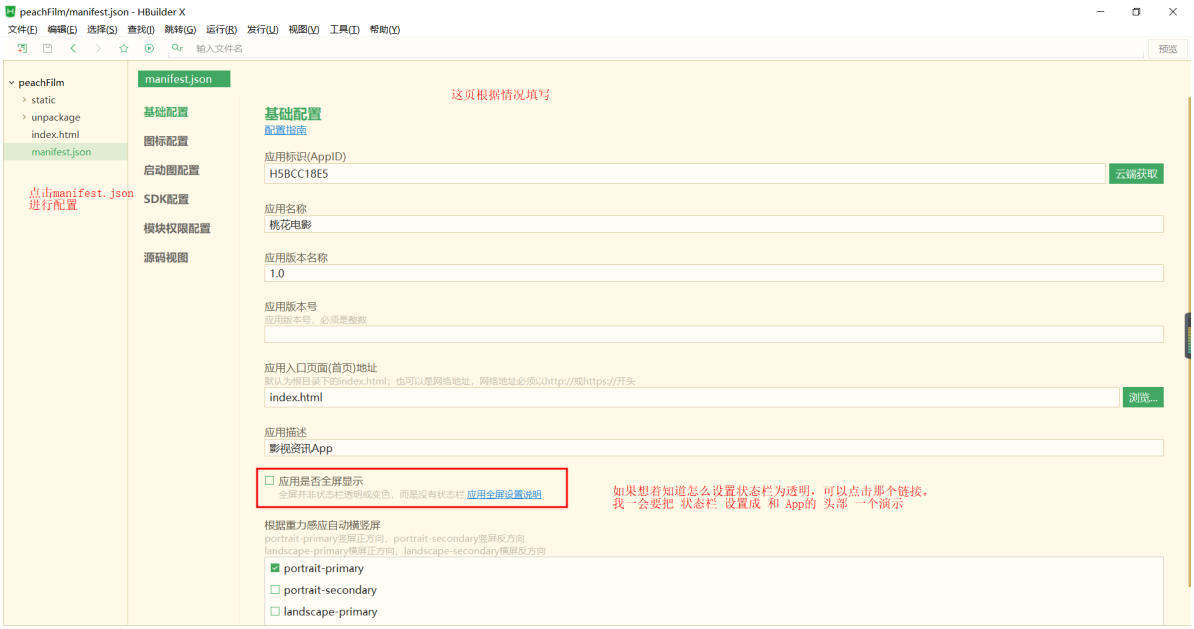



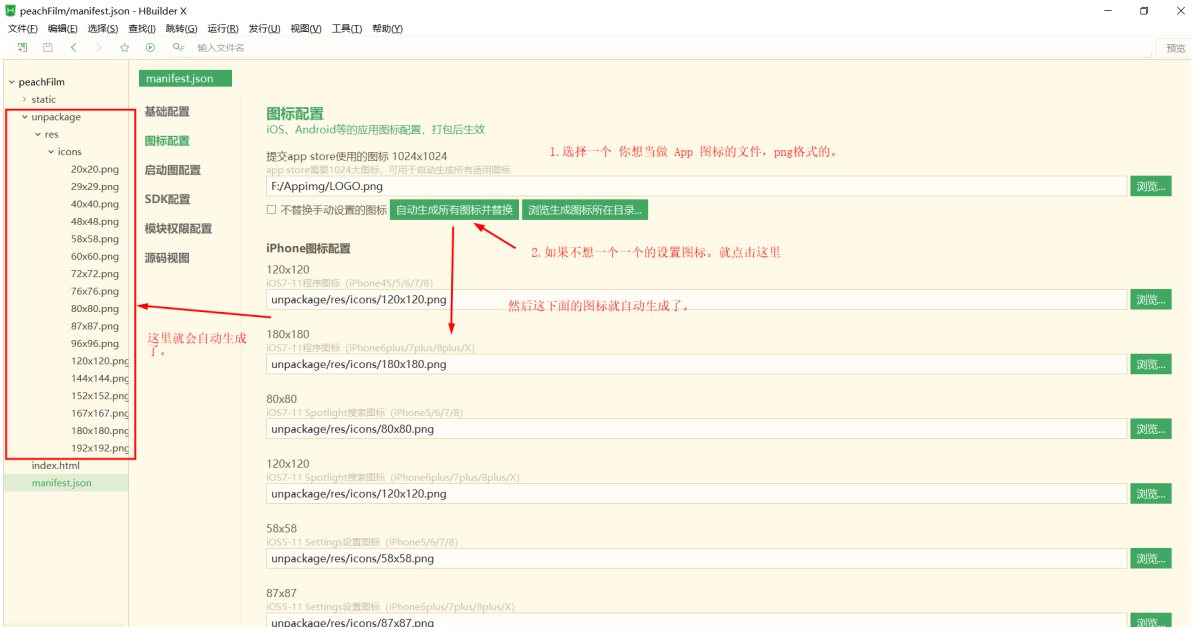

这是我vue 项目打包后的dist 文件。

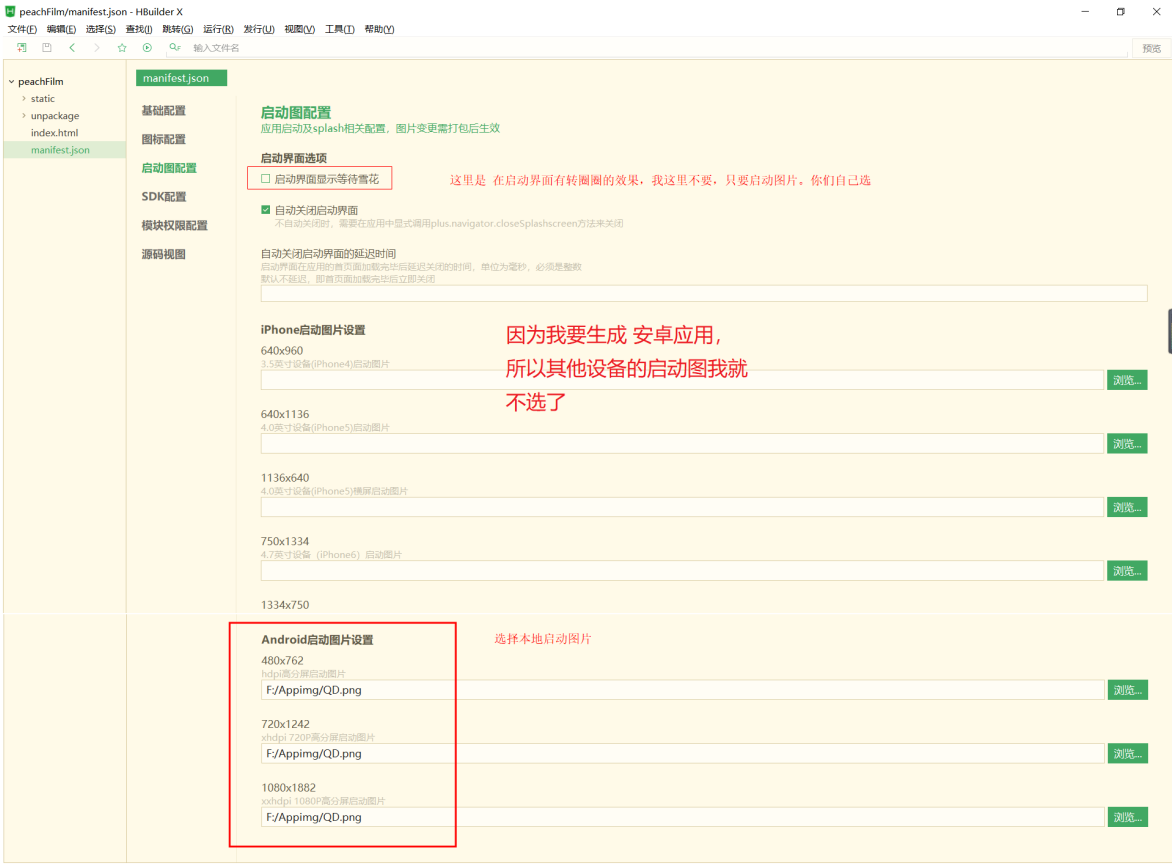



未命名-1 - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
输入文件名
peachFilm
css
img
js
static
unpackage
index.html
manifest.json
未命名-1
1
放入我的项目文件

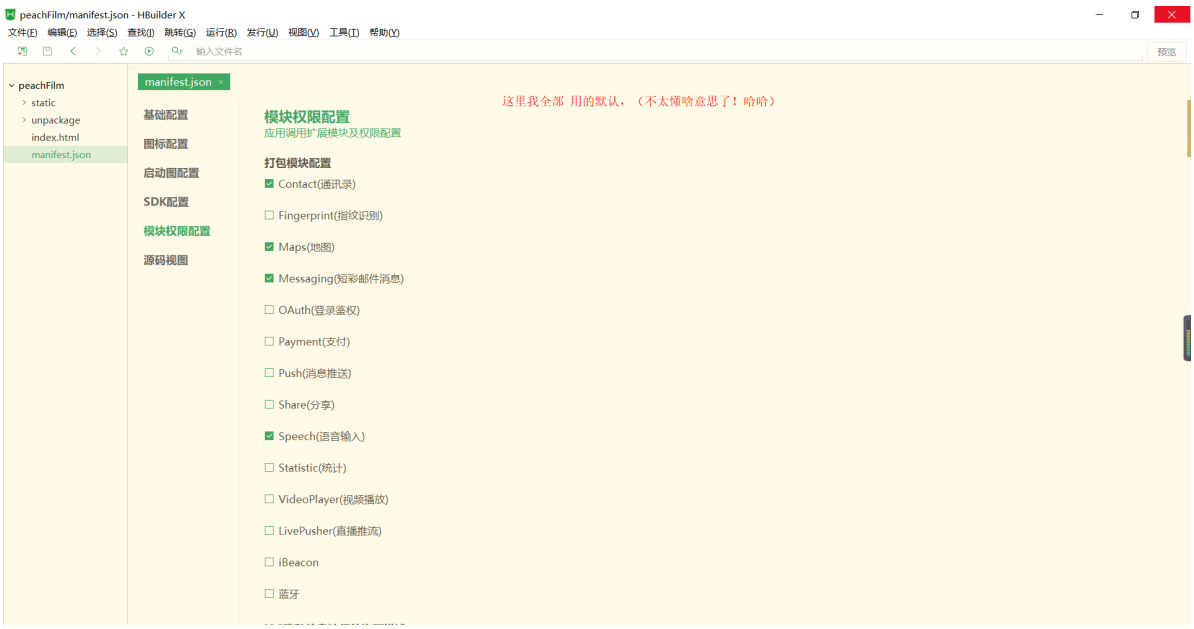

未命名-1 - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
输入文件名
peachFilm
css
img
js
static
unpackage
index.html
manifest.json
复制(C) Ctrl+C
粘贴(P) Ctrl+V
删除(D)
外部命令(E)
未命名-1
1
删除无用的 文件夹


peachFilm/manifest.json - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
输入文件名
预览
peachFilm
static
unpackage
index.html
manifest.json
点击manifest.json进行配置
manifest.json
基础配置
图标配置
启动图配置
SDK配置
模块权限配置
源码视图
这页根据情况填写
基础配置
配置指南
应用标识(AppID)
H5BCC18E5
云端获取
应用名称
桃花电影
应用版本名称
1.0
应用版本号
应用版本号，必须是整数
应用入口页面(首页)地址
默认为根目录下的index.html；也可以是网络地址，网络地址必须以http://或https://开头
index.html
浏览...
应用描述
影视资讯App
应用是否全屏显示
全屏并非状态栏透明或变色，而是没有状态栏 应用全屏设置说明
如果想着知道怎么设置状态栏为透明，可以点击那个链接。
我一会要把 状态栏 设置成 和 App的 头部 一个演示
根据重力感应自动横竖屏
portrait-primary竖屏正方向，portrait-secondary竖屏反方向
landscape-primary横屏正方向，landscape-secondary横屏反方向
portrait-primary
portrait-secondary
landscape-primary

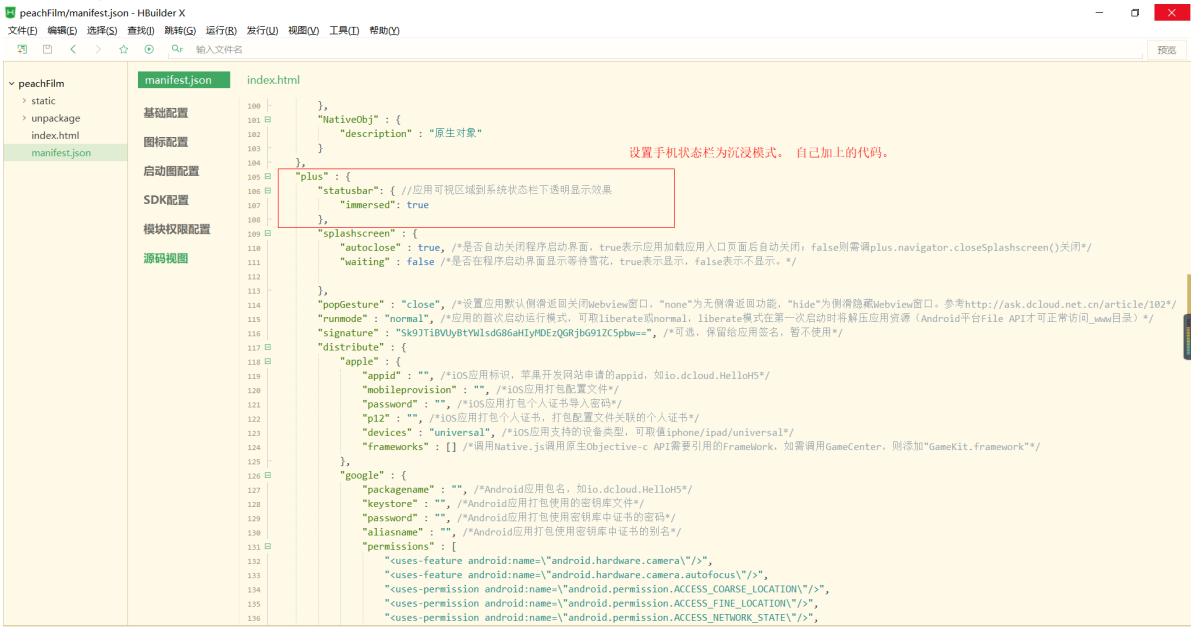

peachFilm/manifest.json - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
输入文件名
预览
peachFilm
static
unpackage
res
icons
20x20.png
29x29.png
40x40.png
48x48.png
58x58.png
60x60.png
72x72.png
76x76.png
80x80.png
87x87.png
96x96.png
120x120.png
144x144.png
152x152.png
167x167.png
180x180.png
192x192.png
index.html
manifest.json
这里就会自动生成了。
manifest.json
基础配置
图标配置
启动图配置
SDK配置
模块权限配置
源码视图
图标配置
iOS、Android等的应用图标配置，打包后生效
1. 选择一个 你想当做 App 图标的文件，png格式的。
提交app store使用的图标 1024x1024
app store需要1024大图标，可用于自动生成所有适用图标
F:/Appimg/LOGO.png
浏览...
不替换手动设置的图标
自动生成所有图标并替换
浏览生成图标所在目录...
2. 如果不想一个一个的设置图标。就点击这里
然后这下面的图标就自动生成了。
iPhone图标配置
120x120
iOS7-11程序图标（iPhone4S/5/6/7/8）
unpackage/res/icons/120x120.png
浏览...
180x180
iOS7-11程序图标（iPhone6plus/7plus/8plus/X）
unpackage/res/icons/180x180.png
浏览...
80x80
iOS7-11 Spotlight搜索图标（iPhone5/6/7/8）
unpackage/res/icons/80x80.png
浏览...
120x120
iOS7-11 Spotlight搜索图标（iPhone6plus/7plus/8plus/X）
unpackage/res/icons/120x120.png
浏览...
58x58
iOS5-11 Settings设置图标（iPhone5/6/7/8）
unpackage/res/icons/58x58.png
浏览...
87x87
iOS5-11 Settings设置图标（iPhone6plus/7plus/8plus/X）
unpackage/res/icons/87x87.png
浏览...


peachFilm/manifest.json - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
输入文件名
预览
peachFilm
static
unpackage
index.html
manifest.json
manifest.json
基础配置
图标配置
启动图配置
SDK配置
模块权限配置
源码视图
启动图配置
应用启动及splash相关配置，图片变更需打包后生效
启动界面选项
启动界面显示等待雪花
这里是 在启动界面有转圈圈的效果，我这里不要，只要启动图片。你们自己选
自动关闭启动界面
不自动关闭时，需要在应用中显式调用plus.navigator.closeSplashscreen方法来关闭
自动关闭启动界面的延迟时间
启动界面在应用的首页面加载完毕后延迟关闭的时间，单位为毫秒，必须是整数
默认不延迟，即首页面加载完毕后立即关闭
iPhone启动图片设置
因为我要生成 安卓应用，
所以其他设备的启动图我就
不选了
640x960
3.5英寸设备(iPhone4)启动图片
浏览...
640x1136
4.0英寸设备(iPhone5)启动图片
浏览...
1136x640
4.0英寸设备(iPhone5)横屏启动图片
浏览...
750x1334
4.7英寸设备（iPhone6）启动图片
浏览...
1334x750
Android启动图片设置
选择本地启动图片
480x762
hdpi高分屏启动图片
F:/Appimg/QD.png
浏览...
720x1242
xhdpi 720P高分屏启动图片
F:/Appimg/QD.png
浏览...
1080x1882
xxhdpi 1080P高分屏启动图片
F:/Appimg/QD.png
浏览...


peachFilm/manifest.json - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
输入文件名
预览
peachFilm
static
unpackage
index.html
manifest.json
manifest.json
基础配置
图标配置
启动图配置
SDK配置
模块权限配置
源码视图
SDK配置
使用三方SDK，需在这些SDK的官网申请key等信息填回到此处，打包后生效
地图(只能选择一个)
地图配置指南
百度定位或地图
appkey_ios
iOS平台百度地图应用密钥
appkey_android
Android平台百度地图应用密钥
高德定位地图
因为我的项目中 使用了 高德定位 所以这里我填上了 自己项目中的 key
下面的 其它配置， 根据自己的需求搞。
appkey_ios
iOS平台高德地图应用KEY
4a7452f7896f5c32565b66a045bb5cde
appkey_android
Android平台高德地图应用KEY
4a7452f7896f5c32565b66a045bb5cde
登录鉴权
第三方登录鉴权配置指南
微信登录
appid
微信开放平台申请应用的AppID值
appsecret
微信开放平台申请应用的AppSecret值

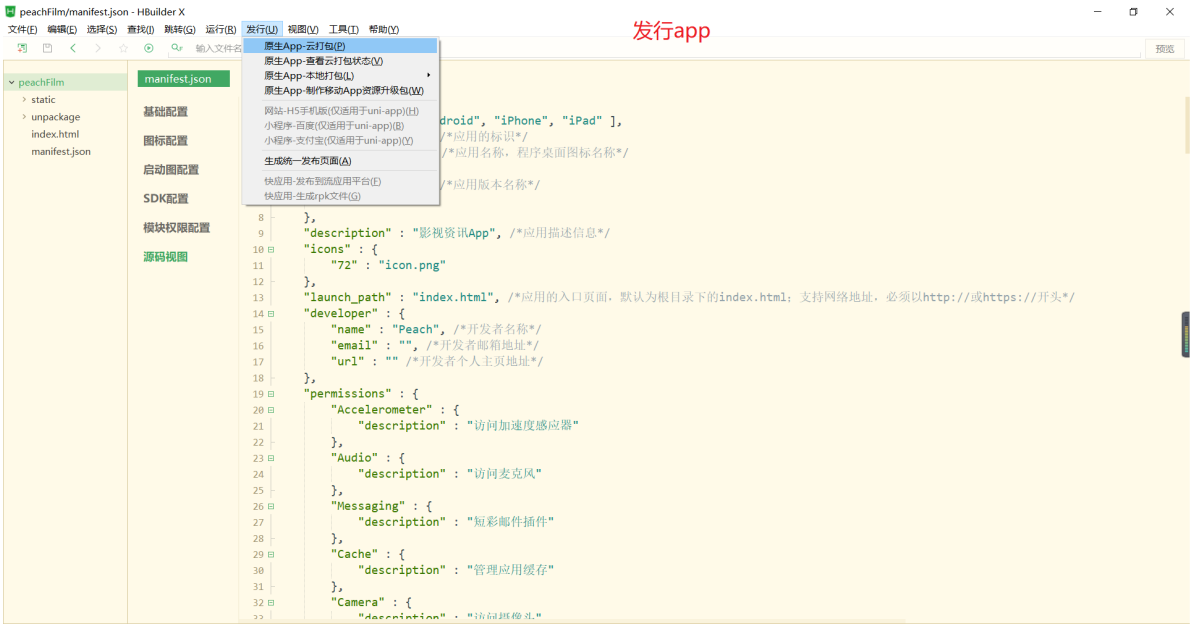


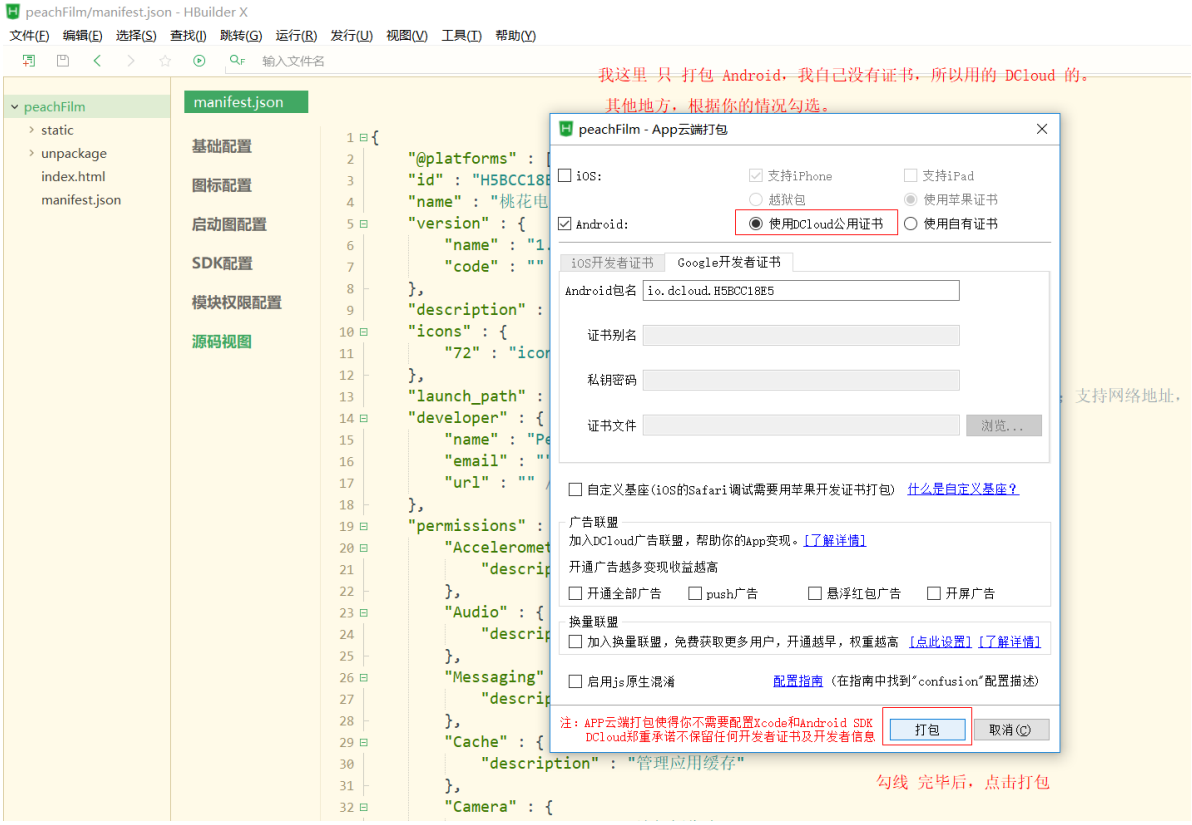

设置沉浸状态栏：（什么是沉浸状态栏和设置方法 请移步： http://ask.dcloud.net.cn/article/32 地址里的 http://ask.dcloud.net.cn/article/1150）




上图中添加位置的代码：

```
"statusbar": { //应用可视区域到系统状态栏下透明显示效果
    "immersed": true
},
```


发行app
peachFilm/manifest.json - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
原生App-云打包(P)
原生App-查看云打包状态(V)
原生App-本地打包(L)
原生App-制作移动App资源升级包(W)
网站-H5手机版(仅适用于uni-app)(H)
小程序-百度(仅适用于uni-app)(B)
小程序-支付宝(仅适用于uni-app)(Y)
生成统一发布页面(A)
快应用-发布到流应用平台(F)
快应用-生成rpk文件(G)
peachFilm
static
unpackage
index.html
manifest.json
manifest.json
基础配置
图标配置
启动图配置
SDK配置
模块权限配置
源码视图


peachFilm/manifest.json - HBuilder X
文件(F) 编辑(E) 选择(S) 查找(I) 跳转(G) 运行(R) 发行(U) 视图(V) 工具(T) 帮助(Y)
我这里 只 打包 Android，我自己没有证书，所以用的 DCloud 的。
其他地方，根据你的情况勾选。
peachFilm
static
unpackage
index.html
manifest.json
manifest.json
基础配置
图标配置
启动图配置
SDK配置
模块权限配置
源码视图
peachFilm - App云端打包
iOS:
支持iPhone
支持iPad
越狱包
使用苹果证书
Android:
使用DCloud公用证书
使用自有证书
iOS开发者证书
Google开发者证书
Android包名
io.dcloud.H5BCC18E5
证书别名
私钥密码
证书文件
浏览...
自定义基座(iOS的Safari调试需要用苹果开发证书打包)
什么是自定义基座？
广告联盟
加入DCloud广告联盟，帮助你的App变现。[了解详情]
开通广告越多变现收益越高
开通全部广告
push广告
悬浮红包广告
开屏广告
换量联盟
加入换量联盟，免费获取更多用户，开通越早，权重越高
[点此设置]
[了解详情]
启用js原生混淆
配置指南
(在指南中找到"confusion"配置描述)
注：APP云端打包使得你不需要配置Xcode和Android SDK
DCloud郑重承诺不保留任何开发者证书及开发者信息
打包
取消(C)
支持网络地址，
勾线 完毕后，点击打包