## Uni-app自主学习

### **hbuild制作流程（详见：**<https://uniapp.dcloud.io/quickstart>）

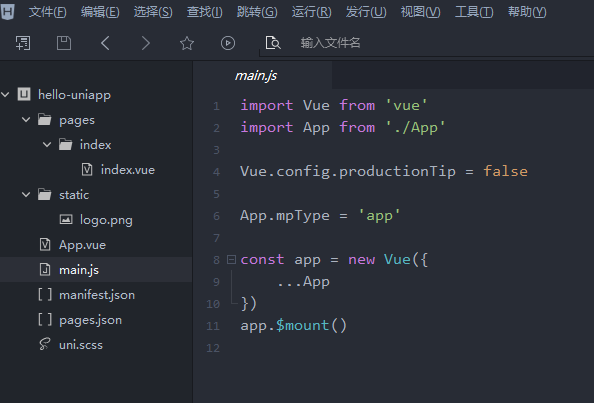
（1）下载Hbuilder X 安装



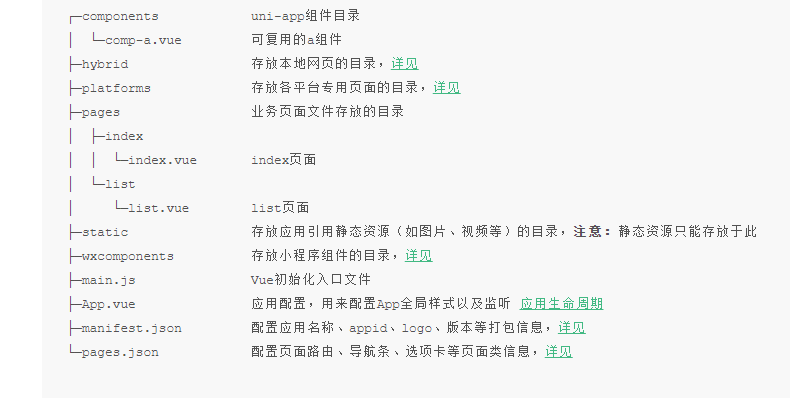
（2）新建项目，选择uni-app模板



（3）生成一个初始结构：



（4）官方Demo目录解释：



（5）Package.json的配置：



### **vue/cli 方式创建项目**

1. 全局安装@vue/cli
2. 命令行创建：vue create -p dcloudio/uni-preset-vue my-project
3. 进入到my-project里面，就可以运行，进行修改页面了
4. 在VS code编辑器也可以做

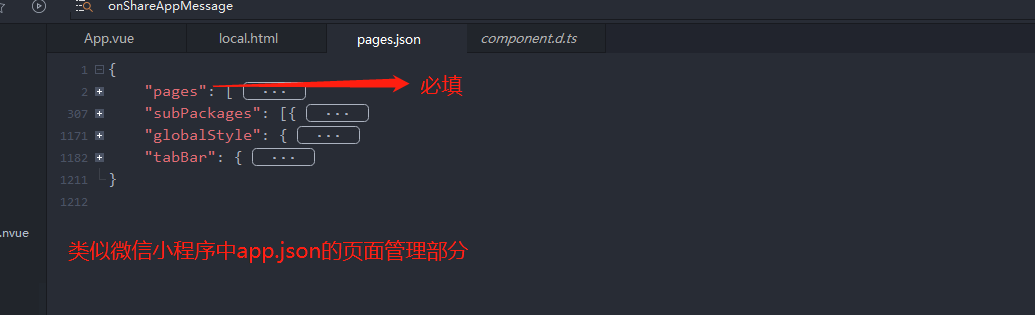
Uni-app使用vue语法和小程序的标签、api；为多端使用，建议flex布局

官方demo链接：

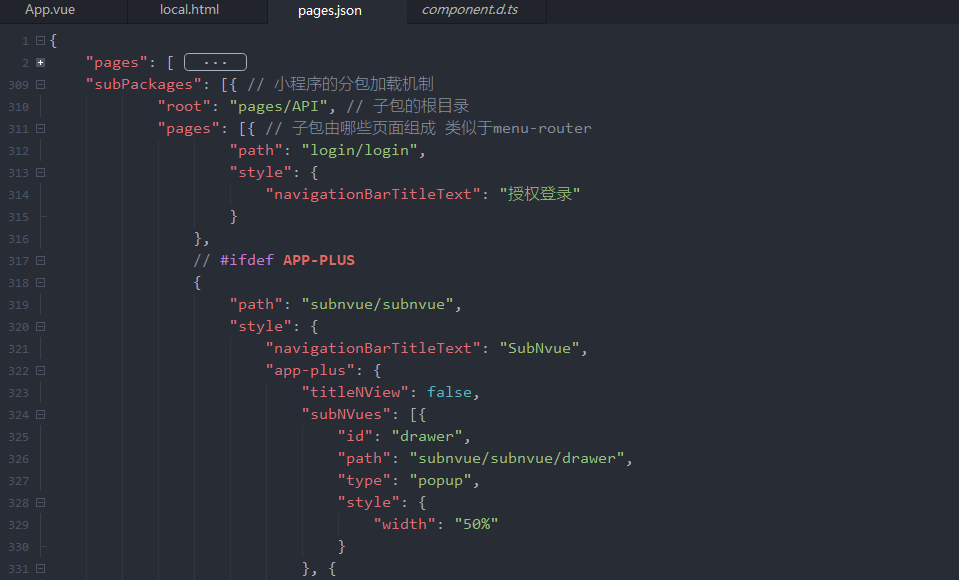
<https://uniapp.dcloud.io/README?id=%e5%bf%ab%e9%80%9f%e4%bd%93%e9%aa%8c>



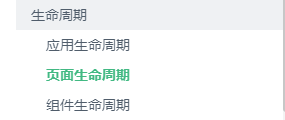
注：图片小于 40kb，uni-app 会自动将其转化为 base64 格式；



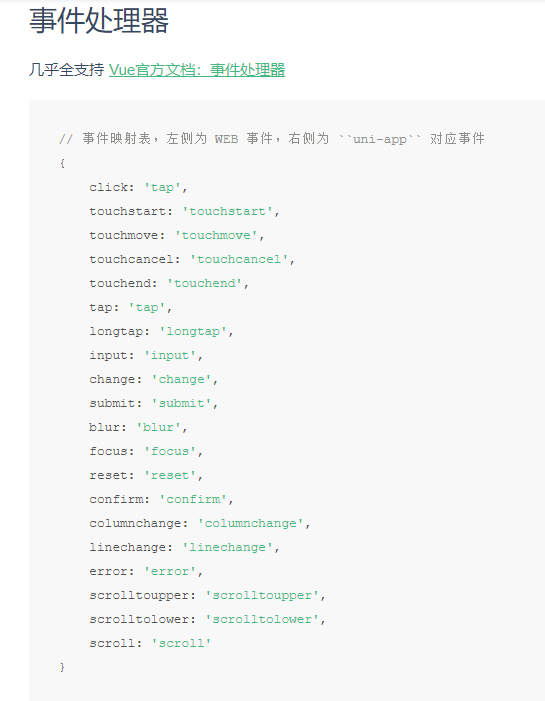




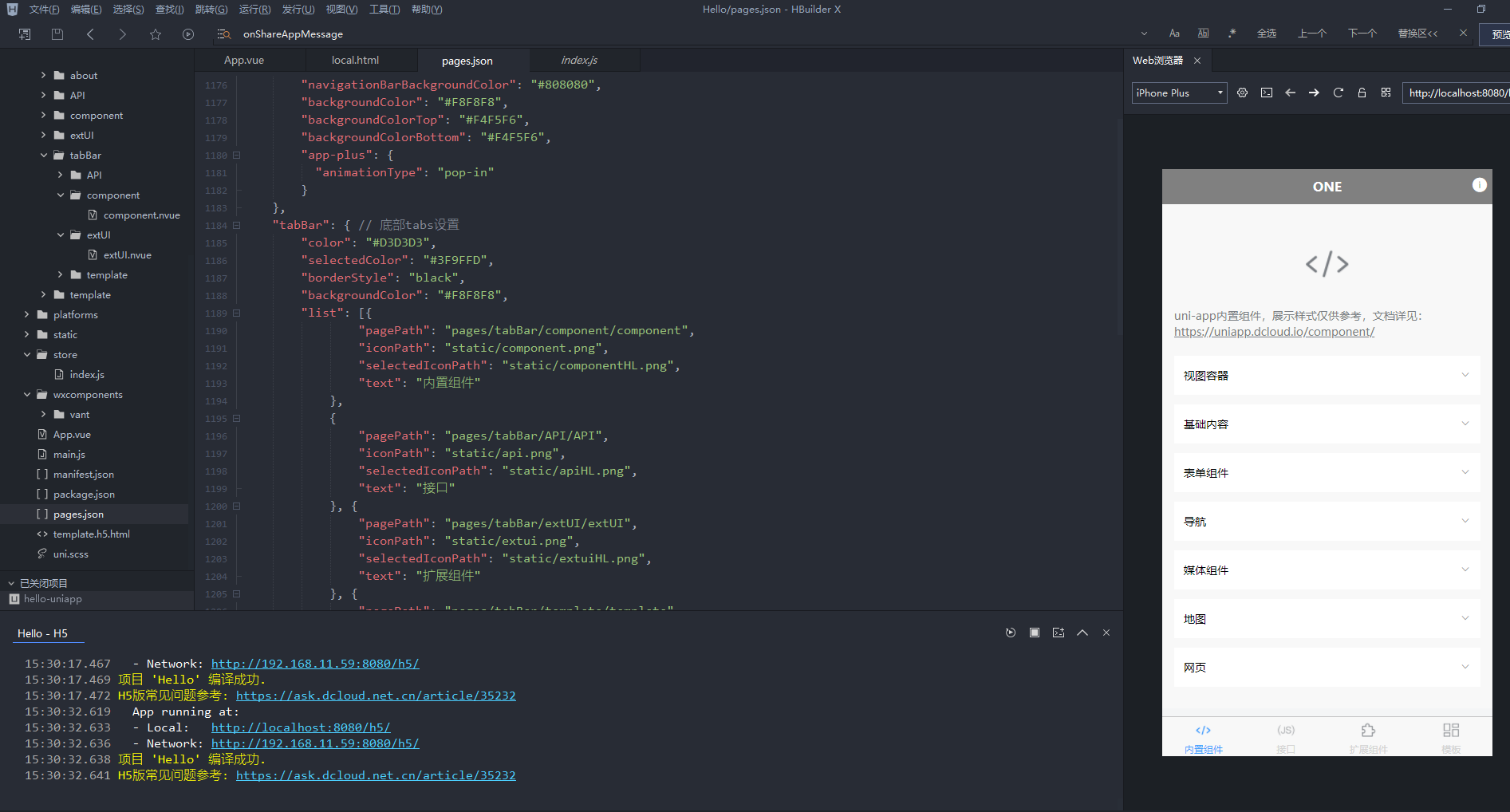
注意：

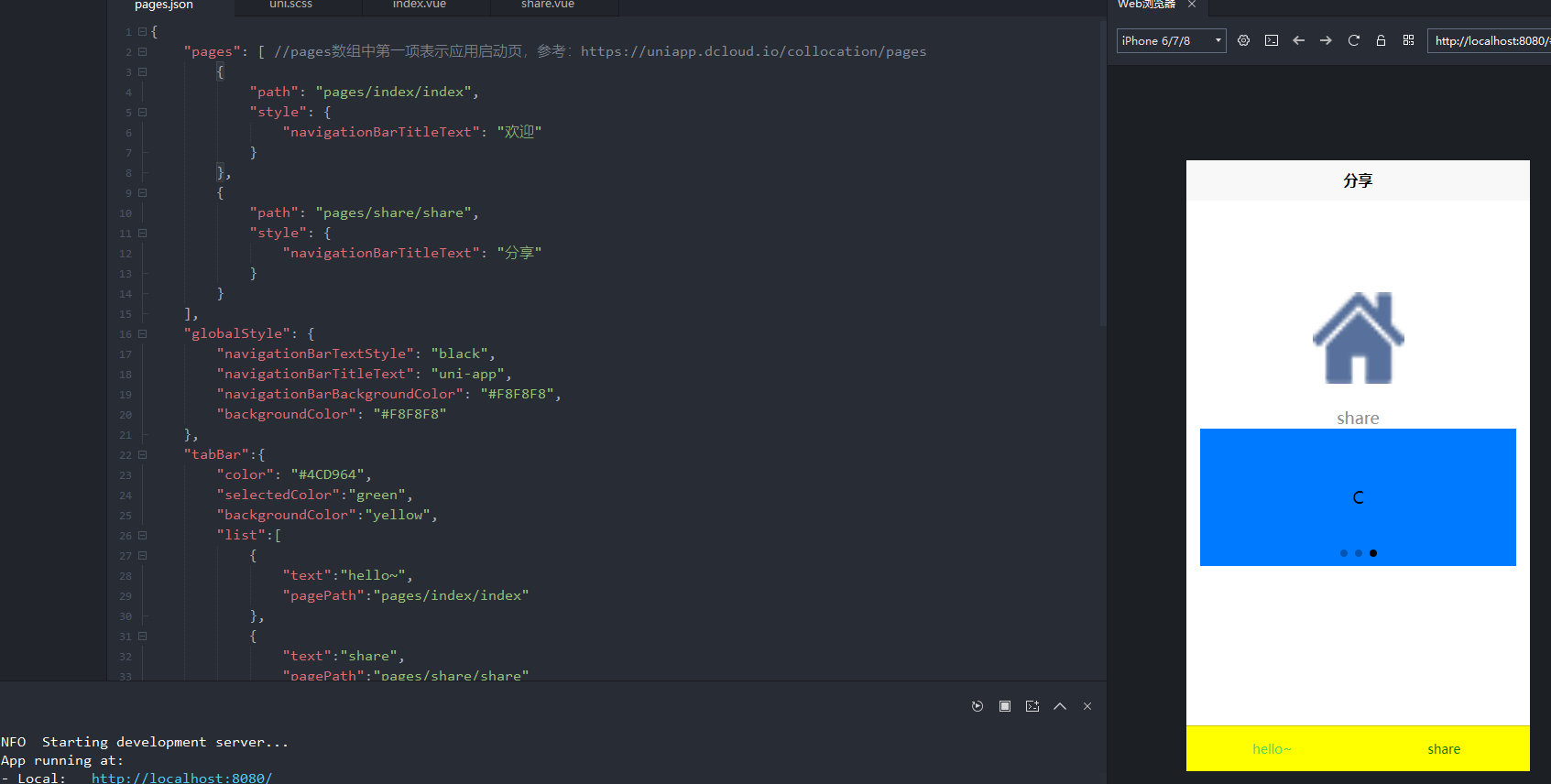


具有三个生命周期



可以利用编辑器自带的预览功能，随时进行查看和调试：

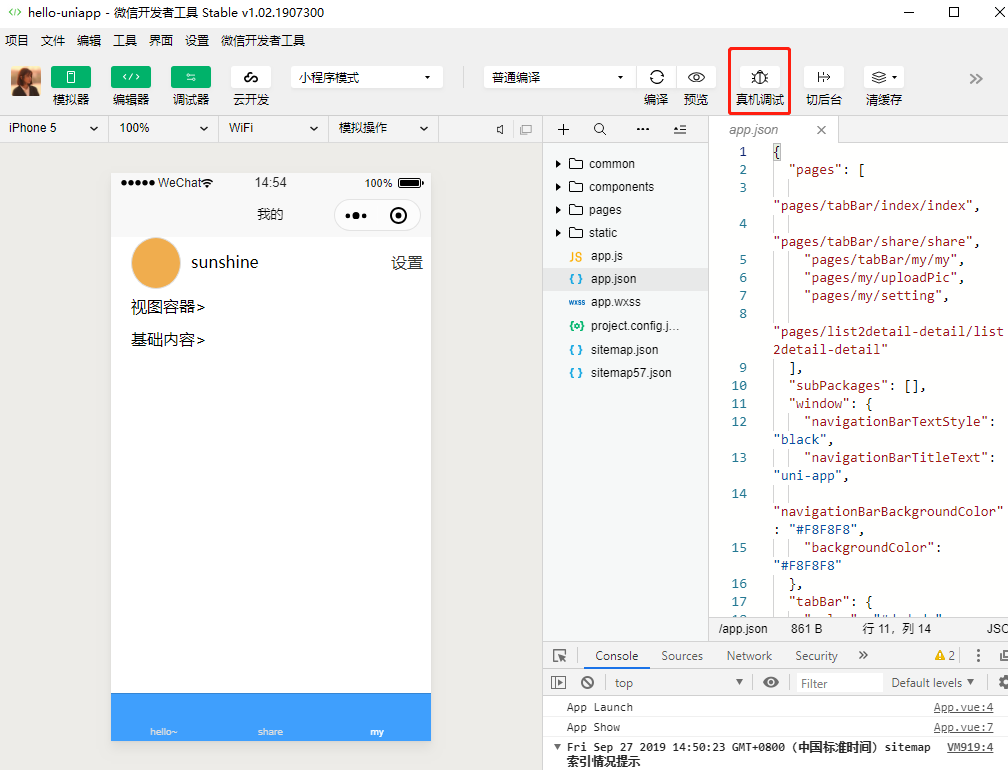




还可以用微信开发者工具（需要把提前在编辑器的“运行”里面配置好工具的安装路径）：

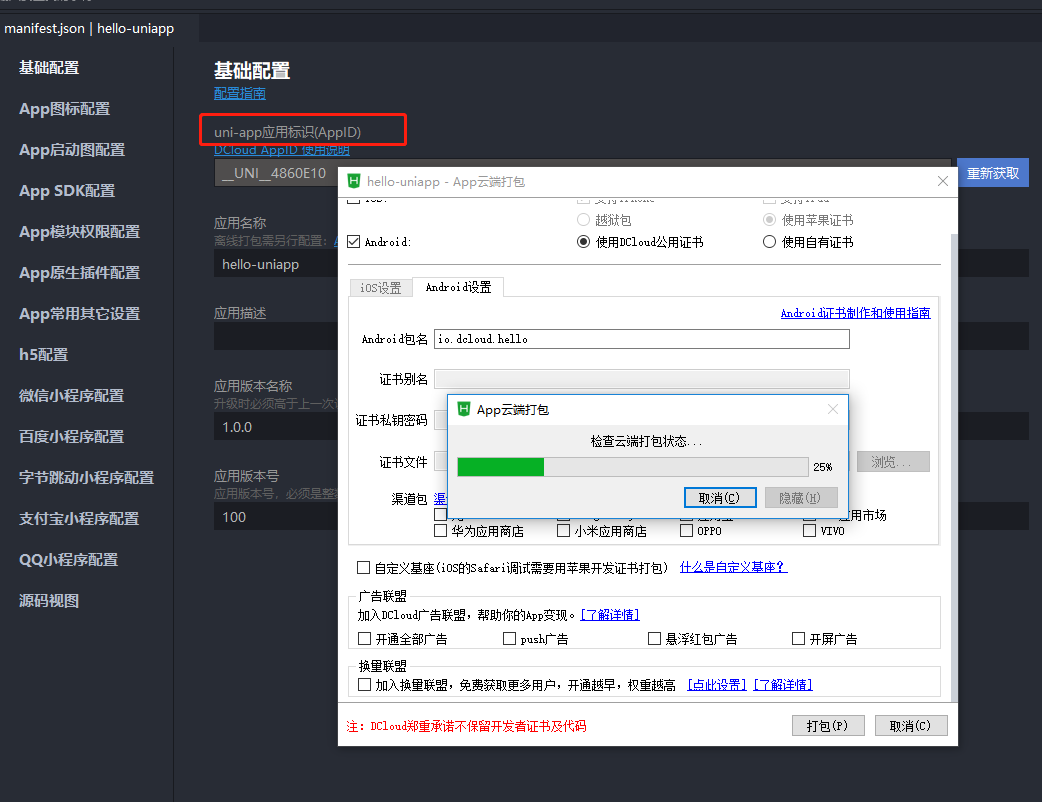
想尝试真机调试，将自己的appId放到project.config.json中





打包与发布app:

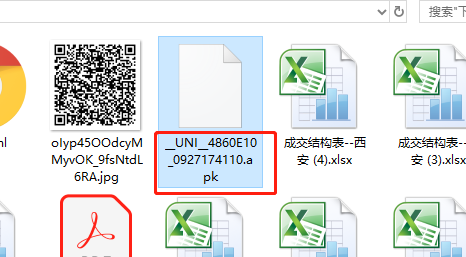
**使用“hbuilder的编辑器的“发行” -> “云打包”，**



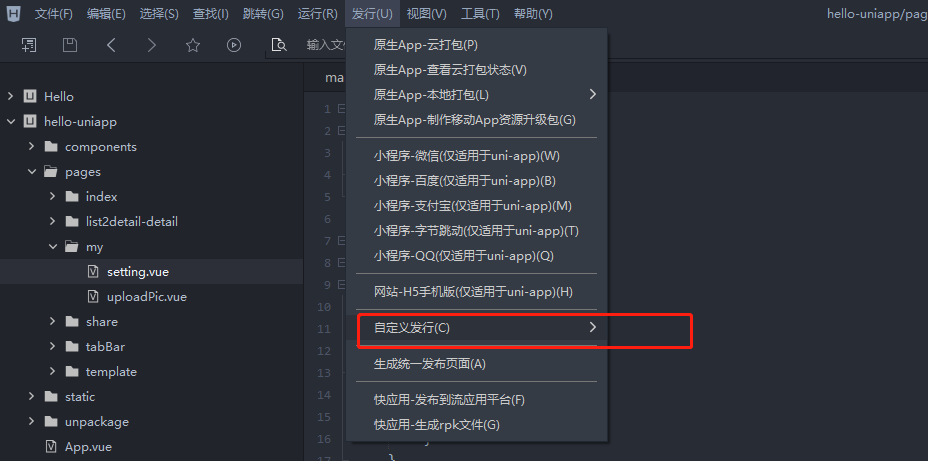
等待几分钟：



点击网址下载：

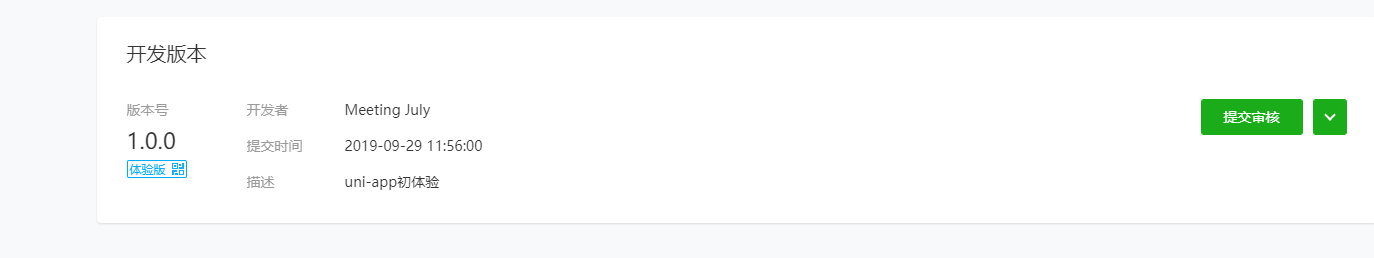


**微信发行：**



再到微信开发者工具->工具->上传，上传成功后,再打开微信公众平台：





这样就成功啦！

个人demo：



1. **开发优缺点**
2. **优点**：

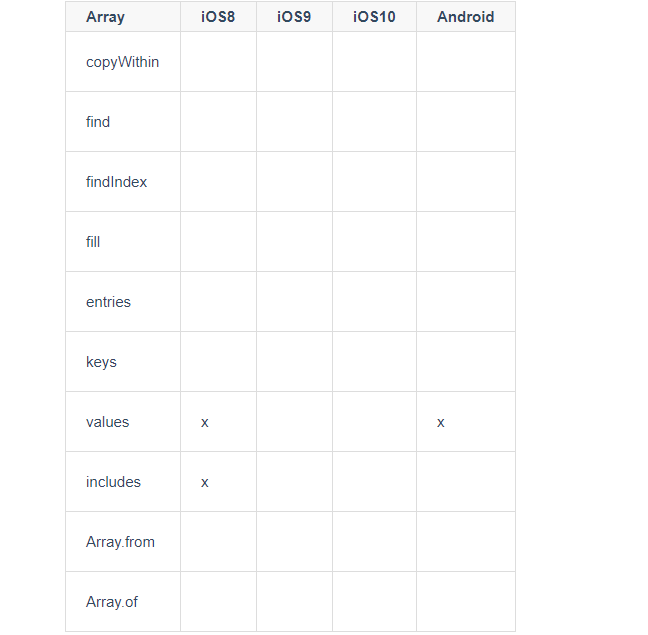
语法和**Vue**差不多，组件和**微信小程序**的组件很像；所以如果都了解这两个的学习和上手的时间比较快。

支持**多端**，weex的原生渲染；**基本组件多**；

通用ui风格（只需要写一套界面ui，就可以适应不同手机的用户）；

HbuilderX里面具备立即使用的插件环境和调试、预览的能力，

支持**绝大多数的ES6**语法，少数不支持，比如：



1. **缺点：**

官方回应不是很及时；

在app端动画不太灵活；

每一端都有些许差异，需要仔细 **研读文档**，上手容易，但是坑不少，需要适配多端，熟悉文档很重要。如：tabbar 的默认高度，在不同平台不一样。

注：uni-app的获取节点的方式: uni.createSelectorQuery().select(className)

1. **开发成本**

由于uni-app是基于vue2.0实现的，所以和vue的规则相似、和微信小程序组件存在很多相似之处，所以学习成本低，开发上手较快。而且文档和社区都相对完整。存在很多现成的组件。

App端内置了weex的渲染引擎，提供原生的渲染能力；不过如果不开发App，那么不太需要nvue。如果不熟悉 weex，那么建议仍然以使用 vue 为主，在App端某些 vue 表现不佳的场景下使用 nvue 作为强化补充即可。

由于多端之间存在差异性，所以需要考虑api差异、样式适配问题。



Vue & uni-app差异性：

1. 标签：div -> view,

span、font –> text,

a-> navigator,

img -> image,

select-> picker,

iframe -> web-view,

1. Js代码：ajax -> uni.request,

Cookie、seesion.storage -> uni.storage,

Local.storage -> uni.storage,

Alert,confirm -> uni.showmodel,

Window.resize -> uni.onWindowResize

Geolocation -> uni.getLocation 定位

Useragent -> uni.getSystemInfo

1. 样式的单位问题：
2. Px：固定的值；App端，在 pages.json 里的 titleNView 或页面里写的 plus api 中涉及的单位，只支持 px，不支持 rpx。
3. Upx：早期的，现在推荐为rpx
4. rpx：动态的值





1. uni-app 中，没有 document：

可以使用 plus.globalEvent.addEventListener 来实现（注意manifest中需开启新编译器，即自定义组件模式"usingComponents":true）。

1. 条件编译：

由于大量写 if else，会造成代码执行性能低下和管理混乱；

编译到不同的工程后二次修改，会让后续升级变的很麻烦。

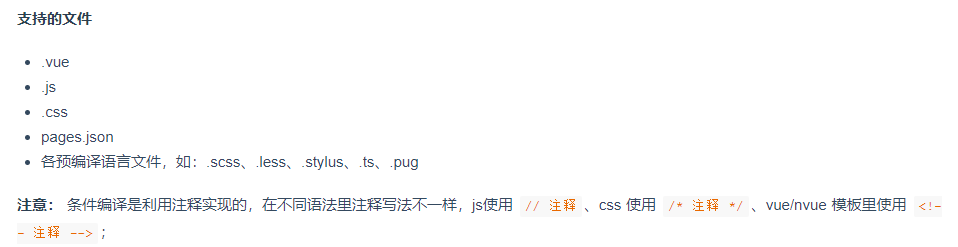
**写法：以 #ifdef 或 #ifndef 加 %PLATFORM% 开头，以 #endif 结尾。**

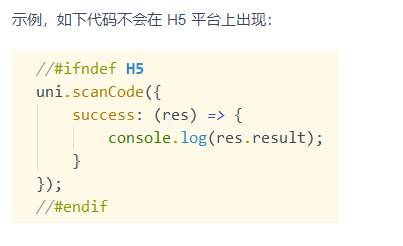
**#ifdef：if defined 仅在某平台存在**

**#ifndef：if not defined 除了某平台均存在**

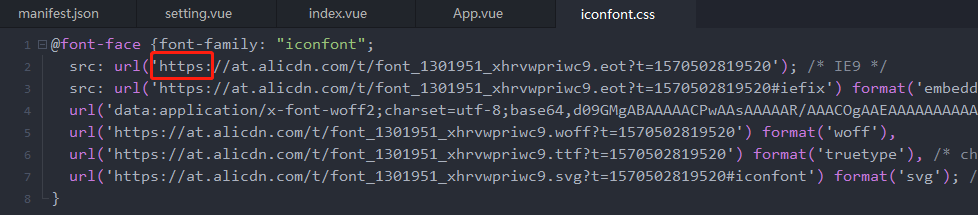
如**：#ifdef APP-PLUS**

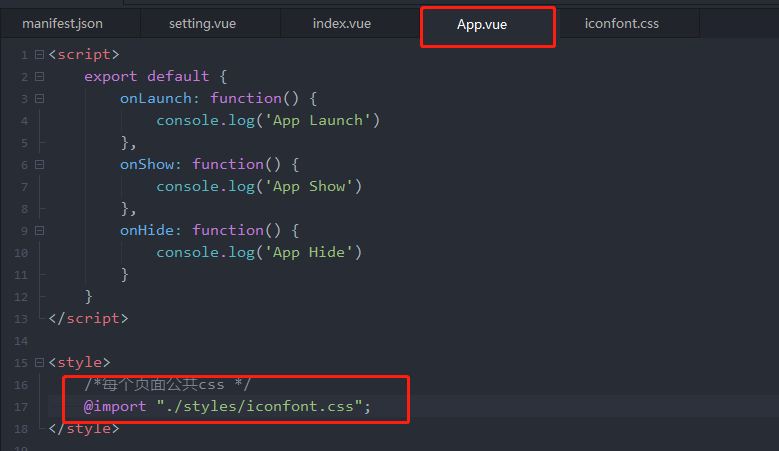


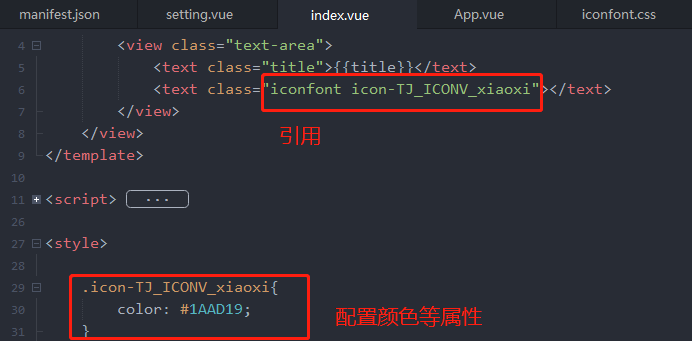




1. 在条件编译区，建议每个平台使用专有api
2. 推荐使用flex布局模型，和rpx单位
3. 使用icon font时，需要注意：在.css 文件中的url里面需要添加https:







1. 全局事件：

触发 uni.$emit(eventName,Obj);

监听 uni.$on(eventName,obj);