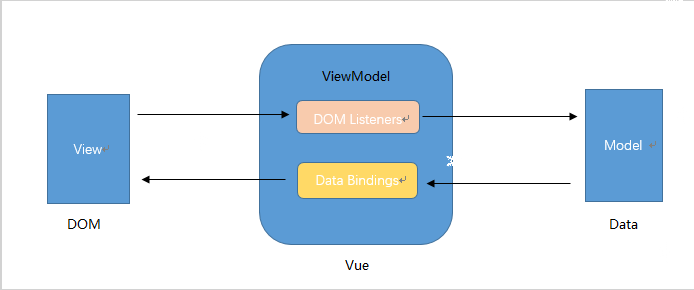
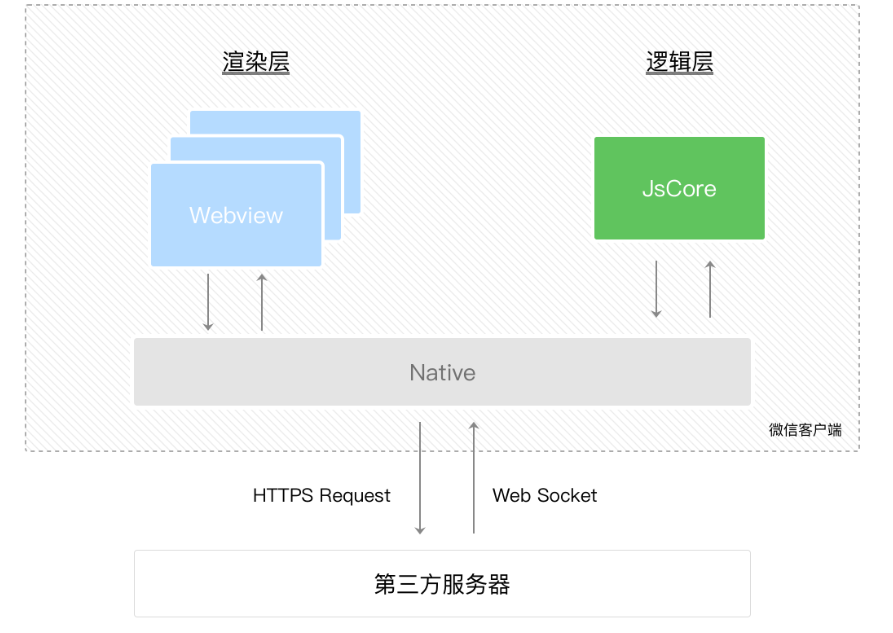
# 开发模式：

MVVM (Model-View-ViewModel)开发模式，mvvm即Model-View-ViewModel，mvvm的设计原理是基于mvc的，所以说mvvm不算是一种创新，充其量是一种改造，



（例如 React, Vue），提倡把渲染和逻辑分离。简单来说就是不要再让 JS 直接操控 DOM，JS 只需要管理状态即可，然后再通过一种模板语法来描述状态和界面结构的关系即可。

# 开发环境说明：



小程序也使用MVVM设计模式，不同于常见MVVM模式的是，它的每一个渲染层都是一个线程（类似于Android的界面开发），Native代表微信客户端，既保证各多个webview线程间的数据通信也保证网络请求的转发。

Webview线程渲染过程： 通过app.json的pages字段，获取所有页面路径，等待渲染 （pages字段的第一个路径代表小程序打开的第一个界面）-> 通过文件路径找到文件渲染界面

Js线程处理过程： 小程序启动首先进入app.js执行onLunch回调 ->根据当前渲染界面获取对应js文件进行数据处理

# 开发文件说明：

1. JSON配置文件

项目的根目录有一个 app.json 和 project.config.json,此外在每个页面文件下都有一个对应各自界面的json文件。三个文件数据格式都严格采用JSON格式，格式错误会报错。三种文件各有用处。

app . json : 当前小程序的全局配置，包括了小程序的所有页面路径、界面表现、网络超时时间、底部 tab 等。

project.config.json : 对开发工具的配置，导入项目后会根据配置文件恢复到自己的个性化设置。

page.json : 与app.json类似但是是针对不同页面做出的不同配置

1. WXML文件

类似于web开发的html界面。标签名字和if，for语法采用wx自己特有格式。

1. WXSS文件

具有CSS文件的大部分特性，又做了一定的扩充。

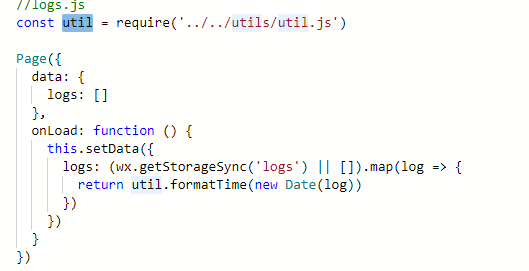
1. 层支持新的尺寸单位 rpx ，开发者可以免去换算的烦恼，只要交给小程序底层来换算即可，由于换算采用的浮点数运算，所以运算结果会和预期结果有一点点偏差。
2. 提供了全局的样式和局部样式。和前边 app.json, page.json 的概念相同，你可以写一个 app.wxss 作为全局样式，会作用于当前小程序的所有页面，局部页面样式 page.wxss 仅对当前页面生效。
3. 此外 WXSS 仅支持部分 CSS 选择器
4. JS文件

完成与用户的交互工作。Js文件格式：



整个小程序只有一个 App 实例，是全部页面共享的

onLaunch： // 小程序启动之后 触发



Page 是一个页面构造器，这个构造器就生成了一个页面。//每个单独的页面都有自己的页面构造器。在渲染完界面之后，页面实例就会收到一个 onLoad 的回调，你可以在这个回调处理你的逻辑。

# 运行机制：

前台/后台：小程序启动后，界面展示给用户时处于前台状态，当通过右上角胶囊或者home键离开小程序时，小程序并没有立即终止，而是处于后台状态。只有当用户很久没有进入小程序或者系统资源紧张时，小程序才可能被销毁，完全终止运行。

销毁：在 iOS 上，当微信客户端在一定时间间隔内（目前是 5 秒）连续收到两次及以上系统内存告警时，会主动进行小程序的销毁，并提示用户 「该小程序可能导致微信响应变慢被终止」。

建议小程序在必要时使用 wx.onMemoryWarning 监听内存告警事件，进行必要的内存清理

# 启动场景：

冷启动：当用户第一次进入小程序，或小程序被销毁后再次进入，小程序会重新加载，属于冷启动。

热启动：小程序还存在没有被销毁的情况下再次打开属于热启动。

启动场景分类：

A:

B:打开小程序的某个页面，除A场景外的其他。

热启动会遇到一些情况，按照场景值来分类：



A类场景 重新启动策略：小程序如果被销毁，B类场景的情况下，下次启动将会进入制定的界面，如果是A类场景启动，一般默认是进入首页，但是我们也可以通过在配置文件中配置“restartStrategy”字段进行修改，可以选择两个选项：‘homePage’，‘homePageAndLatestPage’（下次冷启动恢复到上次关闭时的页面，但是页面中的状态不会保存，比如输入框内容，如需还原或部分还原，则需要使用到退出状态来设置），\*\*小程序退出太久，当前默认为一天，restartStrategy也不会起作用，而是会打开首页。

//\*\*\*\*\*\*\*\*\*退出状态：小程序在销毁前，页面的回调函数onSaveExitState会被调用， 可以在这里保存一些数据。下次启动时通过exitState来获得这些数据



onSaveExitState包含两个返回值，data为保存的数据，exprieTimeStamp，为超时时刻，超过这个时间后保存的数据将会被删除，并且下次冷启动也会打开首页。有时候小程序因为异常被系统强制杀死时不会调用这个方法，下次启动时也会进入首页。