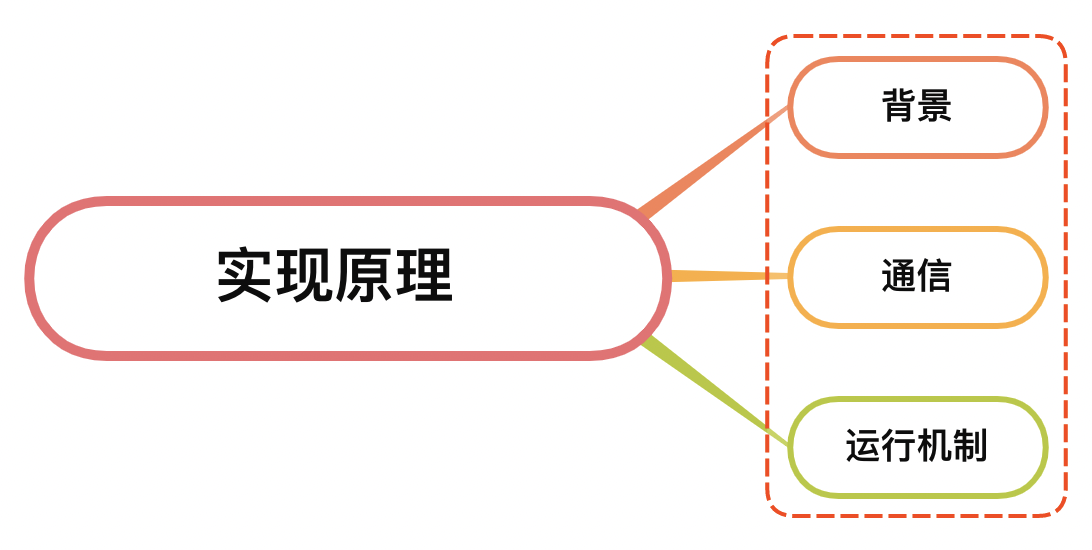
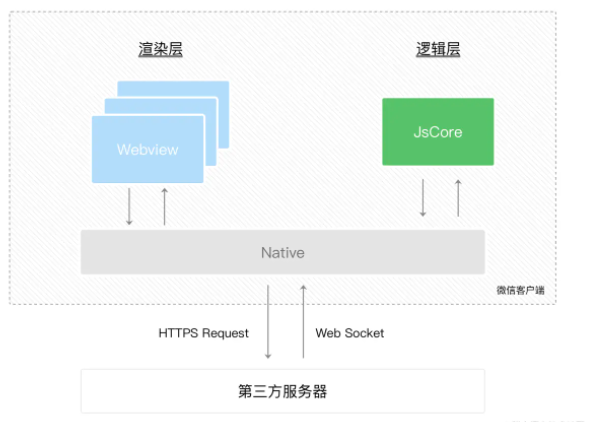
# 面试官：说说微信小程序的实现原理？



## 一、背景

网页开发，渲染线程和脚本是互斥的，这也是为什么长时间的脚本运行可能会导致页面失去响应的原因，本质就是我们常说的 JS 是单线程的

而在小程序中，选择了 Hybrid 的渲染方式，将视图层和逻辑层是分开的，双线程同时运行，视图层的界面使用 WebView 进行渲染，逻辑层运行在 JSCore 中

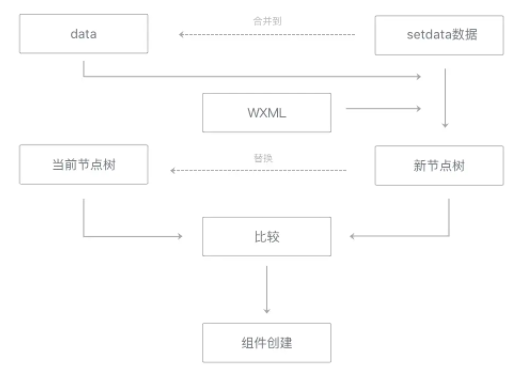


* 渲染层：界面渲染相关的任务全都在 WebView 线程里执行。一个小程序存在多个界面，所以渲染层存在多个 WebView 线程
* 逻辑层：采用 JsCore 线程运行 JS 脚本，在这个环境下执行的都是有关小程序业务逻辑的代码

## 二、通信

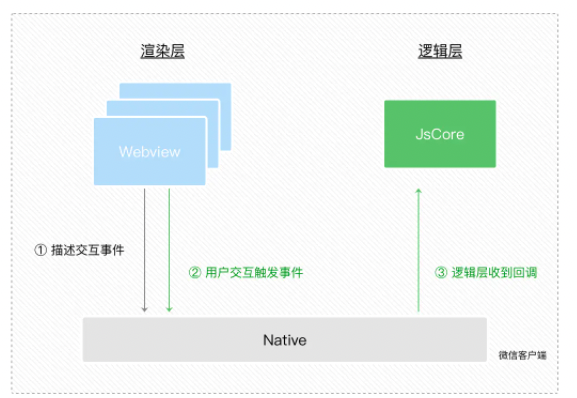
小程序在渲染层，宿主环境会把wxml转化成对应的JS对象

在逻辑层发生数据变更的时候，通过宿主环境提供的setData方法把数据从逻辑层传递到渲染层，再经过对比前后差异，把差异应用在原来的Dom树上，渲染出正确的视图



当视图存在交互的时候，例如用户点击你界面上某个按钮，这类反馈应该通知给开发者的逻辑层，需要将对应的处理状态呈现给用户

对于事件的分发处理，微信进行了特殊的处理，将所有的事件拦截后，丢到逻辑层交给JavaScript进行处理

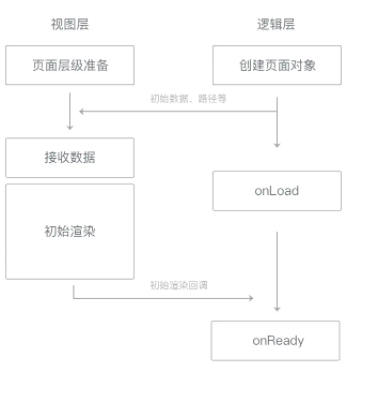


由于小程序是基于双线程的，也就是任何在视图层和逻辑层之间的数据传递都是线程间的通信，会有一定的延时，因此在小程序中，页面更新成了异步操作

异步会使得各部分的运行时序变得复杂一些，比如在渲染首屏的时候，逻辑层与渲染层会同时开始初始化工作，但是渲染层需要有逻辑层的数据才能把界面渲染出来

如果渲染层初始化工作较快完成，就要等逻辑层的指令才能进行下一步工作

因此逻辑层与渲染层需要有一定的机制保证时序正确，在每个小程序页面的生命周期中，存在着若干次页面数据通信

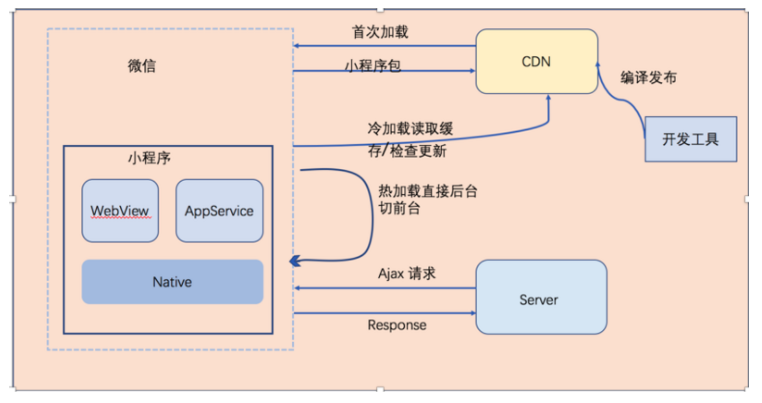


## 三、运行机制

小程序启动运行两种情况：

* 冷启动（重新开始）：用户首次打开或者小程序被微信主动销毁后再次打开的情况，此时小程序需要重新加载启动，即为冷启动
* 热启动：用户已经打开过小程序，然后在一定时间内再次打开该小程序，此时无需重新启动，只需要将后台态的小程序切换到前台，这个过程就是热启动

#### 需要注意： > 1.小程序没有重启的概念  
> 2.当小程序进入后台，客户端会维持一段时间的运行状态，超过一定时间后会被微信主动销毁  
> 3.短时间内收到系统两次以上内存警告，也会对小程序进行销毁，这也就为什么一旦页面内存溢出，页面会奔溃的本质原因了



开发者在后台发布新版本之后，无法立刻影响到所有现网用户，但最差情况下，也在发布之后 24 小时之内下发新版本信息到用户

每次冷启动时，都会检查是否有更新版本，如果发现有新版本，将会异步下载新版本的代码包，并同时用客户端本地的包进行启动，即新版本的小程序需要等下一次冷启动才会应用上

## 参考文献

* https://developers.weixin.qq.com/community/develop/article/doc/0008a4c4f28f30fe3eb863b2750813
* https://juejin.cn/post/6976805521407868958#heading-5
* https://juejin.cn/post/6844903805675388942
* https://juejin.cn/post/6844903999863259144#heading-1