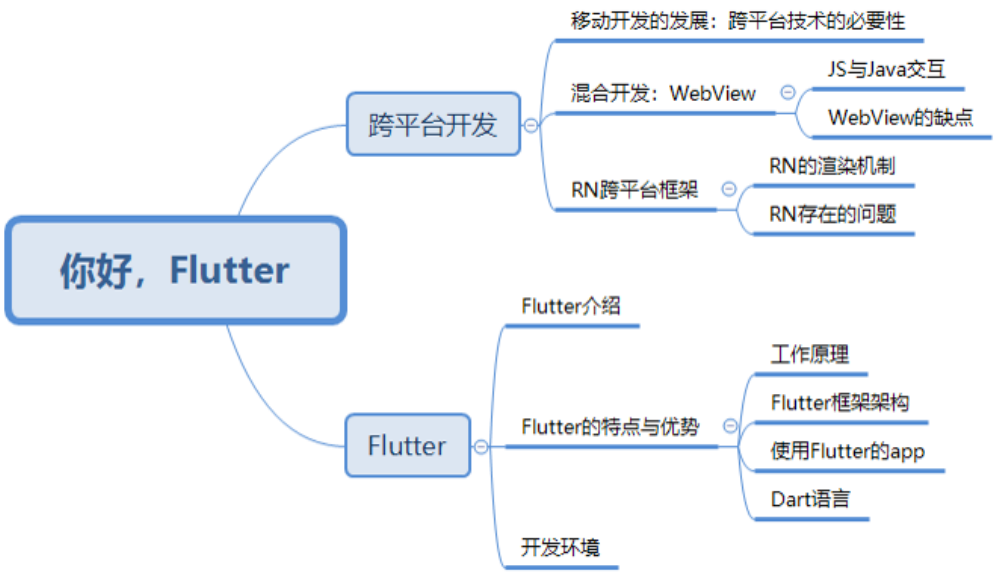
移动开发新利器Flutter



跨平台开发

在传统的原生开发中，一般都要维护Android、iOS两个开发团队，版本迭代时，无论人力成本，还是测试成本都会变大。这一点可能对于我们普通的android程序员感受并不深，或者说我们不关心这一点。但是站在公司的角度上，如果能够有一套代码，直接就能够开发出android的apk和ios的ipa，是不是意味着我只需要拥有开发维护这套代码的一个团队就可以了？从公司人力成本上来说，可以让公司少发一些工资。而在开发的角度，一套代码也能够很好的完成复用、测试以及UI风格的统一。而Flutter正是这样一个能够让我们跨平台开发的框架。

Flutter作为一门相对年轻的技术，对于原生开发的岗位可能并不会那么多，但是对于原生开发的工程师，大家对于这么年轻的Flutter技术的起点并不会差距太大。所以这是不是我们的一个机会呢?

移动开发发展

2011年第一季度，Android在全球的市场份额首次超过塞班系统，跃居全球第一。2011年，移动互联网各种应用开始普及，用户习惯慢慢养成。从2011年到如今，移动互联网实实在在的影响着我们日常生活中的点点滴滴。在互联网草莽兴起的年代，只要能用Android提供的SDK完成一个简单界面的编写就能找到一份不错的工作。但是移动互联网慢慢进入下半场，已经没有了高速发展期的红利，在如今我们移动互联网从业人员更应该从各个方面不断强化自己的技能。

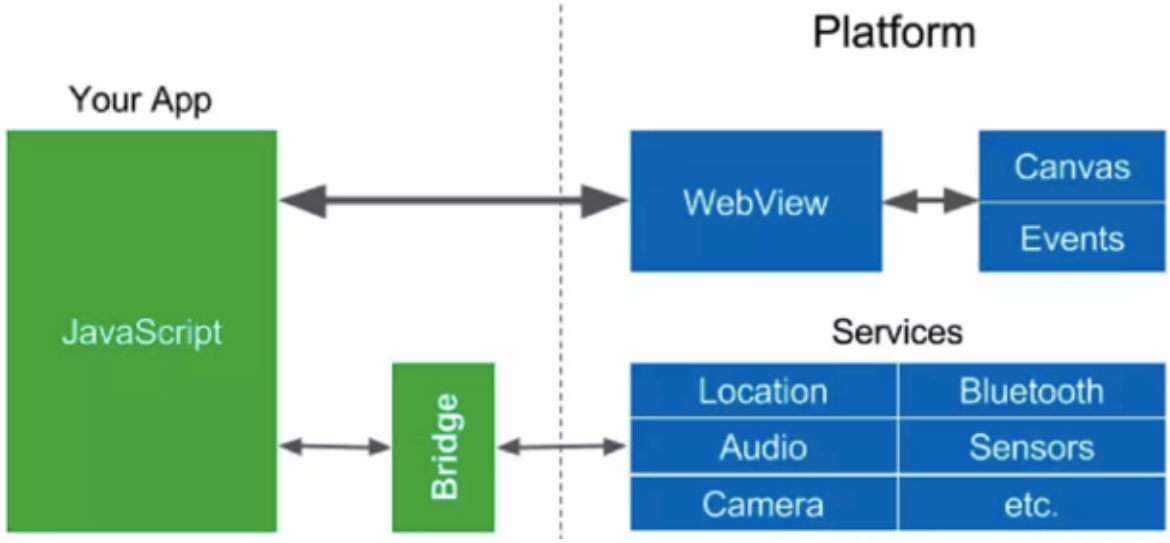
第一阶段：原生开发

在早期的移动开发中，面对这个崭新的方向，人们都处于摸索节点，那时候大家都在进行原生的开发。但是谁都不想放过移动互联网这块蛋糕。Web开发人员的数量远远多于移动互联网开发人员，同时由于原生开发的成本与动态化渐渐无法满足要求。同时原生开发维护成本大、开发周期长，动态化能力弱，所以人们都在寻找跨平台的方案，希望能够只需要一个团队维护一份代码，就能够完成Android与IOS的开发需求。

第二阶段：H5

早在2008年，就有一款叫做**“PhoneGap”**的框架获了奖并开始支持Android平台。现在我们说的PhoneGap一般指的是**"Cordova"**。它是PhoneGap贡献给**Apache**后的开源项目，是从PhoneGap中抽离出的核心代码，是驱动PhoneGap的核心引擎。两者维护的是共同的一份源代码组件，只有名字和包名不一样。PhoneGap是一个采用`HTML，CSS和JavaScript`来完成跨平台开发的技术，当时PhoneGap宣称接近原生性能。然而它的工作原理是基于`WebView`，然后利用`JavaInterface`来完成与原生代码的交互。我们称这种工具为**WebView JavaScript Bridge(JsBridge)**。这种方式，能够很好的解决跨平台与动态更新的需求但是，我们都知道android `WebView`的渲染效率很差，同时`JavaScript`是解释型语言，它不需要编译，在运行时候解释执行，这就导致JavaScript的执行性能太低了。同时因为android自身的问题，使用`WebView`过程中消耗的内存，没有办法在不需要使用的时候进行及时的回收，这样会导致我们的可用内存越来越少，最终OOM。

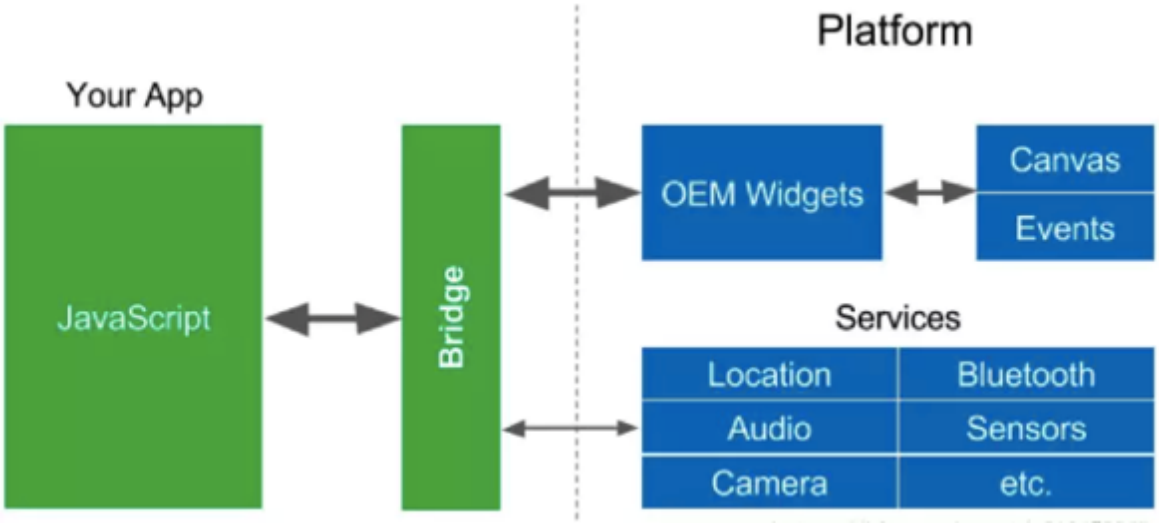
Cordova工作机制图如下：



第三阶段：跨平台框架

为了突破使用`WebView`进行渲染的性能问题，2015年4月，Facebook开源了JS框架`React`在原生移动应用平台的衍生产物——`React Native`。**RN将渲染交给原生，而不是直接在HTML中完成。**

RN工作机制图如下：



我们编写的依旧是`JavaScript`代码。这时候，我们使用JS写好UI后，会组成一个**Virtual DOM**。然后通过Bridge将VD发给原生层去进行UI的创建。这种方式看起来与原生开发非常相似，我们在进行原生开发的时候，写好XML布局再由Java对这个布局XML进行解析，然后通过反射创建对应的View，这个XML就是一个界面的配置文件。而使用RN，我们的配置则由XML变成了JS编写。不同的是我们在原生开发中，xml是静态的配置，而RN中要更新UI，又需要经历一次Bridge传递的过程。同时如果需要在RN中调用原生API也需要通过bridge来进行传递。
这样导致需要频繁的跨桥调用，bridge 的成本太高了。

虽然RN底层是使用原生进行绘制，但是毕竟中间多了一层Bridge，也就是多了一层中介，你租房是希望能直接与房东进行交流还是和中介公司？很显然，去中介公司进行租房，你将花费一些代价。所以RN多了一层中介——Bridge，它绝无可能与原生的性能相媲美，但是为了跨平台，这点代价不是致命的。你会因为中介需要多付中介费，就完全靠自己去找房东个人租房子吗？

Flutter

之前我们了解到，现有的跨平台技术的工作机制。根据之前的了解，我们是不是设想如果拥有一个框架，我们编写的代码直接会根据打包的平台编译为此平台本身的原生代码，在运行时期直接执行这些编译后的原生代码，就和我们进行原生开发一样，不再需要Bridge来担任中介的角色，是不是能够拥有最优秀的性能？

Flutter是什么？

> Flutter是谷歌的移动UI框架，可以快速在iOS和Android上构建高质量的原生用户界面。Flutter可以与现有的代码一起工作。在全世界，Flutter正在被越来越多的开发者和组织使用，并且Flutter是完全免费、开源的。

从官方的介绍中，能看出三个关键点：

- 跨平台移动UI框架
- 与现有的代码一起工作
- 完全免费、开源

第一点：移动端跨平台，一般指的就是Android与IOS两大平台。值得我们讨论的是除了Android、IOS之外Flutter它还是一个叫做**"Fuchsia"**的操作系统主要的UI开发框架。

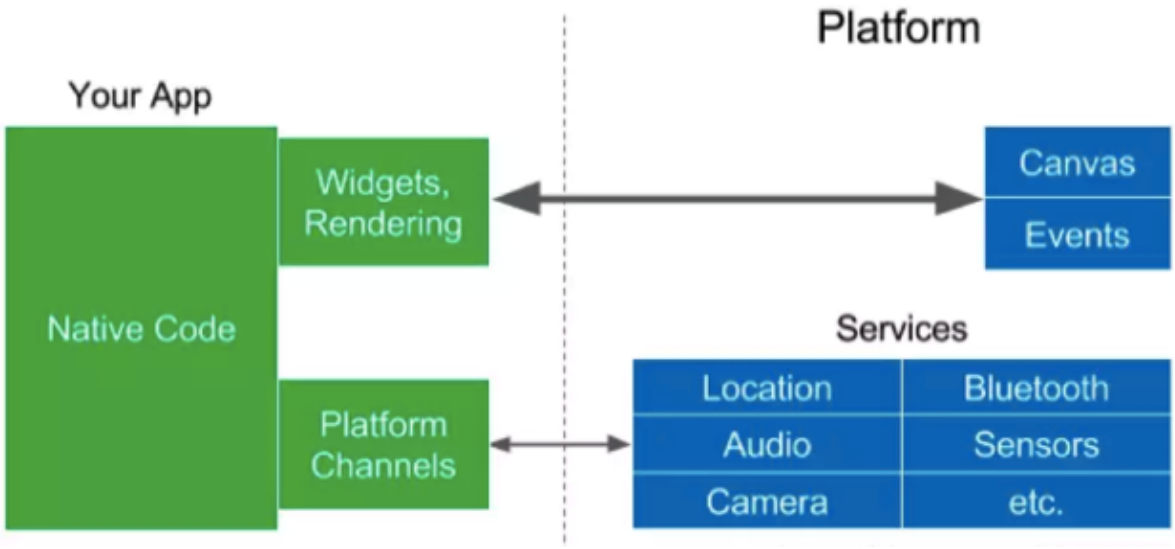
> Fuchsia是Google开发的继Android和Chrome OS之后的第三个系统。外界都在猜测它会取代Android，但是也只是猜测而已。

第二点：与现有的代码一起工作。这意味着，Flutter可以支持进行**"混合开发"**，我们的项目如果一开始是使用原生的形式开发的，在后续业务功能的实现上，我们希望能够利用到跨平台的优势，那么Flutter它运行我们在原有代码的基础上嵌入Flutter来进行开发。

而第三点完全免费开源，这意味着我们可以完全免费的借助Flutter进行商业App的开发，不用担心任何版权问题。

> 17年知乎热帖：[如何看待百度要求内部全面停止使用 React / React Native?](#)，就是由版权引发的全球风暴，导致BAT等已准备弃用RN。

Flutter工作机制图如下：



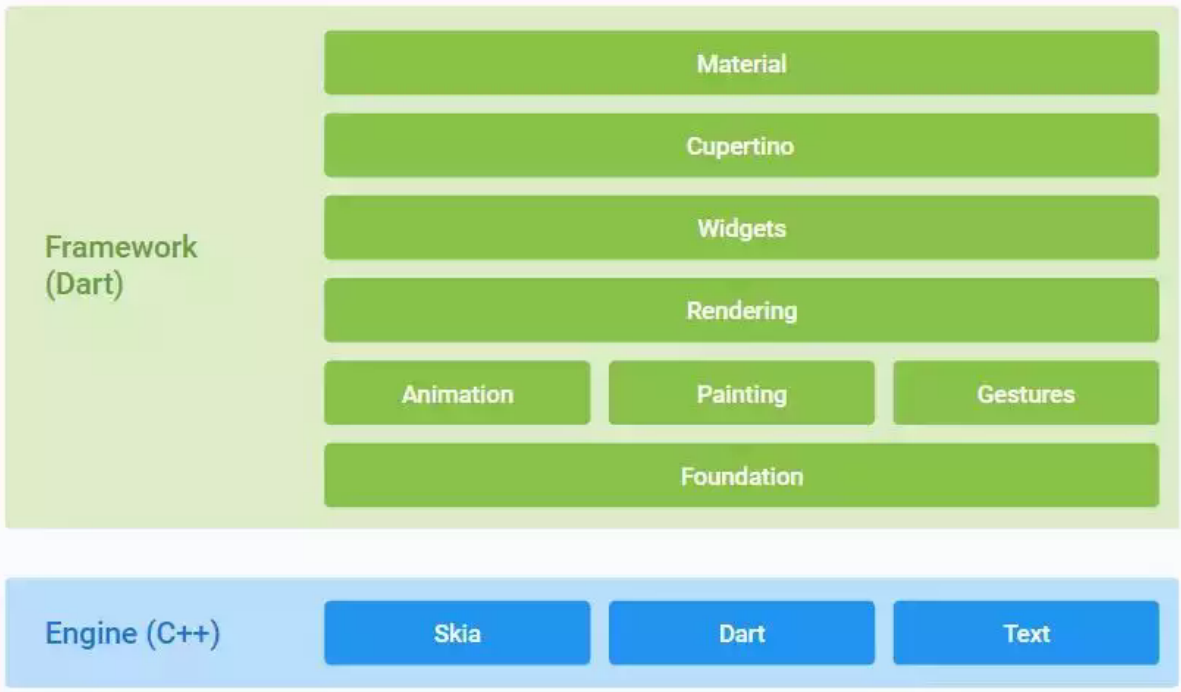
Flutter的工作机制和我们预想的一样，他正是通过将它的代码根据不同的平台编译为对应平台的机器码，这样就不需要Bridge的存在了。Flutter自己嵌入了一个 Dart VM，我们编写的代码会被编译成ARM的机器码，在Android中就是通过Dart VM所在的libflutter.so完成我们自己编写的代码生成的机器码的载入执行。

> Android真机一般都是ARM架构的，模拟器是X86或者x86_64
>
> IOS真机一般也是ARM，模拟器早期都是I386，在5s的模拟器后好像都是x86_64了。

Flutter架构

Flutter框架整体拥有两层架构，由上往下，第一层是Framework类库层，提供给我们在开发时所使用的各种Widget、动画等。而第二层则是Engine引擎层，我们上面所说的Dart VM也就处于这一层。

Flutter架构图如下：



Flutter的特点

> 你觉得Flutter前景怎么样？为什么你会选择使用Flutter?

类似Flutter这种框架的出现是必然的，现有的跨平台框架，比如RN都是基于JS，由于JS的执行性能，导致跨平台应用性能一直无法突破瓶颈。而Flutter在Debug使用JIT编译，支持热重载，能够提高我们的开发效率，而Release中利用AOT直接编译成机器码，能够达到更好的性能。

从设计角度而言，Flutter提供了非常丰富的Widget组件，能够让我们非常轻松的实现Android或者IOS风格的UI效果。

Flutter作为Google官方孵化的项目，拥有更加规范与完善的生态圈，依托官方就是它最大的优势之一。像IOS中的热更新，被苹果官方一句话就让你的努力白费了，但是如果是IOS官方推出的热更新还用担心这样的问题吗?