# 原生开发

原生应用程序是指某一个移动平台（比如iOS或安卓）所特有的应用，使用相应平台支持的开发工具和语言，并直接调用系统提供的SDK API。比如Android原生应用就是指使用Java或Kotlin语言直接调用Android SDK开发的应用程序；而iOS原生应用就是指通过Objective-C或Swift语言直接调用iOS SDK开发的应用程序。

原生开发有以下**主要优势**：

* 可访问平台全部功能（GPS、摄像头）；
* 速度快、性能高、可以实现复杂动画及绘制，整体用户体验好；

**主要缺点**：

* 平台特定，开发成本高；不同平台必须维护不同代码，人力成本随之变大；
* 内容固定，动态化弱，大多数情况下，有新功能更新时只能发版；

纯原生开发主要面临动态化和开发成本两个问题，而针对这两个问题，诞生了一些跨平台的动态化框架。

# 跨平台技术简介

主要分为三类：

* H5 + 原生（Cordova、Ionic、微信小程序）
* JavaScript 开发 + 原生渲染 （React Native、Weex）
* 自绘UI + 原生 (Qt for mobile、Flutter)

### H5 + 原生

这类框架主要原理就是将 App 中需要动态变动的内容通过HTML5来实现，通过原生的网页加载控件WebView （Android）或 WKWebView（iOS）来加载。这种方案中，H5 部分是可以随时改变而不用发版，动态化需求能满足；同时，由于 H5 代码只需要一次开发，就能同时在 Android 和 iOS 两个平台运行，这也可以减小开发成本，也就是说，H5 部分功能越多，开发成本就越小。

目前混合开发框架的典型代表有：Cordova、Ionic 。大多数 App 中都会有一些功能是 H5 开发的，至少目前为止，HTMLybrid App 仍然是最通用且最成熟的跨端解决方案。

混合开发技术点

如之前所述，原生开发可以访问平台所有功能，而混合开发中，H5代码是运行在 WebView 中，而 WebView 实质上就是一个浏览器内核，其 JavaScript 依然运行在一个权限受限的沙箱中，所以对于大多数系统能力都没有访问权限，如无法访问文件系统、不能使用蓝牙等。所以，对于 H5 不能实现的功能，就需要原生去做了。

混合框架一般都会在原生代码中预先实现一些访问系统能力的 API ， 然后暴露给 WebView 以供 JavaScript 调用。这样一来，WebView 中 JavaScript 与原生 API 之间就需要一个通信的桥梁，主要负责 JavaScript 与原生之间传递调用消息，而消息的传递必须遵守一个标准的协议，它规定了消息的格式与含义。

混合应用的**优点**：

* 动态内容是 H5，Web 技术栈
* 社区及资源丰富

**缺点**：

* 性能体验不佳
* 对于复杂用户界面或动画，WebView 有时会不堪重任。

### JavaScript 开发 + 原生渲染

React Native

React Native 实现了跨平台。 相对于混合应用，由于React Native是 原生控件渲染，所以性能会比混合应用中 H5 好一些，同时 React Native 提供了很多原生组件对应的 Web 组件，大多数情况下开发者只需要使用 Web 技术栈 就能开发出 App。

React Native的**优点**

* React Native可将标记元素转化为真实的原生UI元素，利用在任何平台上所呈现视图的现有方法
* React Native与主UI线程分开工作，所以应用程序可以在不牺牲功能的前提下保持最大的性能
* React Native创建的应用程序允许开发人员创建和构建跨平台应用程序，节省了跨平台应用程序开发时间，并且应用程序维护更便宜
* 在组件开发方面，Native移动应用程序使用大量代码和类来在UI中进行渲染，它将在UI中呈现这两个平台都为移动应用开发节省了时间。

React Native的**缺点**：

* 它仍处于改进阶段，应用程序在升级过程中也会遇到了很多问题。
* 除了仅编写代码之外，开发人员有时还需要为与React Native不兼容的组件编写额外的本机代码，有时候与本机组件相比，React Native组件变得更差
* 它缺乏安全稳健性，React Native是一个JavaScript库和开源框架，它在安全性方面存在重大问题
* React Native在最初渲染之前需要花费大量时间来初始化运行，因为JavaScript线程需要时间来初始化

Weex

Weex 是阿里巴巴于 2016 年发布的跨平台移动端开发框架，思想及原理和 React Native 类似，底层都是通过原生渲染的，不同是应用层开发语法 （即 DSL，Domain Specific Language）：Weex 支持 Vue 语法和 Rax 语法，Rax 的 DSL(Domain Specific Language) 语法是基于 React JSX 语法而创造，而 RN 的 DSL 是基于 React 的，不支持 Vue。

**优点**

* 更加轻量。Weex 采用了 Vue 作为上层框架，相较于 React 更加轻量，Vue 的官网宣传就是非常轻量，体积小巧，语法简单。
* 学习成本小。
* 跨平台支持Web端

**缺点**

* 学习资料少，从问世的时间上来看，Weex 的学习资料比较少
* 社区活跃度不够，Weex 相较于 RN 起步比较晚，Weex目前社区主要由阿里人员在维护, 相较于已经成熟的RN社区目前还有很多不足
* Weex 现在存在的 BUG 相较于 RN 还比较多，对于使用来说会有一些影响。

总结

JavaScript 开发 + 原生渲染 的方式**主要优点**如下：

* 采用 Web 开发技术栈，社区庞大、上手快、开发成本相对较低。
* 原生渲染，性能相比 H5 提高很多。
* 动态化较好，支持热更新。

**不足**：

* 渲染时需要 JavaScript 和原生之间通信，在有些场景如拖动可能会因为通信频繁导致卡顿。
* JavaScript 为脚本语言，执行时需要解释执行，执行效率和编译类语言仍有差距。
* 由于渲染依赖原生控件，不同平台的控件需要单独维护，并且当系统更新时，社区控件可能会滞后；如果需要自定义原生渲染组件时，开发和维护成本过高。

### 自绘UI + 原生

自绘UI + 原生。这种技术的思路是：通过在不同平台实现一个统一接口的渲染引擎来绘制UI，而不依赖系统原生控件，所以可以做到不同平台UI的一致性。

这种平台技术的**优点**如下：

* 性能高；由于自绘引擎是直接调用系统API来绘制UI，所以性能和原生控件接近。
* 灵活、组件库易维护、UI外观保真度和一致性高
* 由于UI渲染不依赖原生控件，也就不需要根据不同平台的控件单独维护一套组件库，所以代码容易维护。
* 在不同平台，组件显示外观可以做到高保真和高一致性
* 另外，由于不依赖原生控件，也就不会受原生布局系统的限制，这样布局系统会非常灵活。

**不足**：

* 动态性不足
* 应用开发效率低： C++需要开发者手动去管理内存分配，没有 JavaScript 及Java中垃圾回收（GC）的机制。

Qt 简介

Qt 是一个1991年由 Qt Company 开发的跨平台 C++ 图形用户界面应用程序开发框架，实现了对于 iOS 的完全支持，新增 WinRT、Beautifier 等插件，废弃了无 Python 接口的 GDB 调试支持，集成了基于 Clang 的 C/C++ 代码模块，并对 Android 支持做出了调整，至此实现了全面支持 iOS、Android、WP，它提供给应用程序开发者构建图形用户界面所需的所有功能。

不足：

* Qt 移动开发社区太小，学习资料不足，生态不好。
* 官方推广不利，支持不够。
* 移动端发力较晚，市场已被其它动态化框架占领（ Hybrid 和 RN )。
* 在移动开发中，C++ 开发和Web开发栈相比有着先天的劣势，直接结果就是 Qt 开发效率太低。

Flutter

Flutter 是 Google 发布的一个用于创建跨平台、高性能移动应用的框架。Flutter 和 Qt mobile 一样，都没有使用原生控件，相反都实现了一个自绘引擎，使用自身的布局、绘制系统。

**Flutter优点**

* 混合开发中，最接近原生开发的框架；
* 性能强大，流畅；
* 优秀的路由设计；
* 优秀的动画设计；
* 简单易学，Dart语言更具优势；
* 跨多种平台，减少开发成本；支持插件，可以访问原生系统的调用。

**Flutter缺点**

* 脱离不开原生，开发人员需要具备原生（Android、iOS）基础开发能力；
* 适配问题，开发工具版本升级后，修改量大；
* 原生集成第三方SDK后，兼容性适配是个令人头痛的问题；
* 代码可读性较差，对代码质量和管理要求较高；
* Widget的类型难以选择，糟糕的UI控件API；
* Flutter packages和Dart packages上第三方sdk繁杂，适配性差，不可乱用；
* 目前几乎没有第三方开发者平台开发Flutter能力的SDK，需要原生去集成；

小结

| **技术类型** | **UI渲染方式** | **性能** | **开发效率** | **动态化** | **框架代表** |
| --- | --- | --- | --- | --- | --- |
| H5 + 原生 | WebView渲染 | 一般 | 高 | 支持 | Cordova、Ionic |
| JavaScript + 原生渲染 | 原生控件渲染 | 好 | 中 | 支持 | RN、Weex |
| 自绘UI + 原生 | 调用系统API渲染 | 好 | Flutter高, Qt低 | 默认不支持 | Qt、Flutter |