# 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。

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。由于组件库是同一套代码、同一个渲染引擎，所以在不同平台，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；为了保证UI绘制性能，自绘UI系统一般都会采用 AOT 模式编译其发布包，所以应用发布后，不能像 Hybrid 和 RN 那些使用 JavaScript（JIT）作为开发语言的框架那样动态下发代码。
* 应用开发效率低：Qt 使用 C++ 作为其开发语言，而编程效率是直接会影响 App 开发效率的，C++ 作为一门静态语言，在 UI 开发方面灵活性不及 JavaScript 这样的动态语言，另外，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。2008年，Qt Company 科技被诺基亚公司收购，Qt 也因此成为诺基亚旗下的编程语言工具。2012年，Qt 被 Digia 收购。2014年4月，跨平台集成开发环境 Qt Creator 3.1.0 正式发布，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。

Qt 虽然在 PC 端获得了巨大成功，备受社区追捧，然而其在移动端却表现不佳，在近几年，虽然偶尔能听到 Qt 的声音，但一直很弱，无论 Qt 本身技术如何、设计思想如何，但事实上终究是败了，究其原因，笔者认为主要有四：

第一：Qt 移动开发社区太小，学习资料不足，生态不好。

第二：官方推广不利，支持不够。

第三：移动端发力较晚，市场已被其它动态化框架占领（ Hybrid 和 RN )。

第四：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。

基于此四点，尽管 Qt 是移动端开发跨平台自绘引擎的先驱，但却成为了烈士。

## Flutter出世

“千呼万唤始出来”，铺垫这么久，现在终于等到本书的主角出场了！

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。那么，我们会担心，Qt mobile 面对的问题Flutter是否也一样，Flutter会不会步入Qt mobile后尘，成为另一个烈士？要回到这个问题，我们先来看看Flutter诞生过程：从 2017 年 Google I/O 大会上，Google 首次发布 Flutter 到 2021年8月底，已经有 127K 的 Star，Star 数量 Github 上排名前 20 。经历了4年多的时间，Flutter 生态系统得以快速增长，国内外有非常基于 Flutter 的成功案例，国内的互联网公司基本都有专门的 Flutter 团队。总之，历时 4 年，Flutter 发展飞快，已在业界得到了广泛的关注和认可，在开发者中受到了热烈的欢迎，成为了移动跨端开发中最受欢迎的框架之一。

现在，我们来和 Qt mobile做一个对比：

1. 生态：Flutter 已经有了丰富的生态系统，无论是开发者数量还是第三方组件都已经非常可观。Flutter
2. 技术支持：现在 Google 正在大力推广Flutter，Flutter 的作者中很多人都是来自Chromium团队，并且 Github上活跃度很高。另一个角度，从 Flutter 诞生到现在，频繁的版本发布也可以看出 Google 对 Flutter的投入的资源不小，所以在官方技术支持这方面，大可不必担心。
3. 开发效率：一套代码，多端运行；并且在开发过程中 Flutter 的热重载可帮助开发者快速地进行测试、构建UI、添加功能并更快地修复错误。在 iOS 和 Android 模拟器或真机上可以实现毫秒级热重载，并且不会丢失状态。这真的很棒，相信我，如果你是一名原生开发者，体验了Flutter开发流后，很可能就不想重新回去做原生了，毕竟很少有人不吐槽原生开发的编译速度。

基于以上三点，相信读者和笔者一样，已经迫不及待的想要去了解一下 Flutter 了。到现在为止，我们已经对移动端开发技术有了一个全面的了解，接下来我们便要进入本书的主题，你准备好了吗！

小结

本章主要介绍了目前移动开发中三种跨平台技术，现在我们从框架角度对比一下它们，如表1-1所示：

| **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 |

表1-1: 跨平台技术对比

上表中开发语言主要指应用层的开发语言，而开发效率，是指整个开发周期的效率，包括编码时间、调试时间、以及排错、处理兼容性问题时间。动态化主要指是否支持动态下发代码和是否支持热更新。值得注意的是 Flutter 的Release 包默认是使用 Dart AOT 模式编译的，所以不支持动态化，但 Dart 还有 JIT 或 snapshot 运行方式，这些模式都是支持动态化的。