移动开发技术简介 · 《Flutter实战》

book.flutterchina.club/chapter1/mobile_development_intro.html

移动开发技术简介

原生开发与跨平台技术

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；

在移动互联网发展初期，业务场景并不复杂，原生开发还可以应对产品需求迭代。 但近几年，随着物联网时代到来、移动互联网高歌猛进，日新月异，在很多业务场景中，传统的纯原生开发已经不能满足日益增长的业务需求。主要表现在：

- 动态化内容需求增大；当需求发生变化时，纯原生应用需要通过版本升级来更新内容，但应用上架、审核是需要周期的，这对高速变化的互联网时代来说是很难接受的，所以，对应用动态化(不发版也可以更新应用内容)的需求就变的迫在眉睫。
- 业务需求变化快，开发成本变大；由于原生开发一般都要维护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。

总结一下，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

针对原生开发面临问题，人们一直都在努力寻找好的解决方案，而时至今日，已经有很多跨平台框架(注意，本书中所指的“跨平台”若无特殊说明，即特指Android和iOS两个平台)，根据其原理，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、快应用）
- 自绘UI+原生(QT for mobile、Flutter)

在接下来的章节中我们逐个来看看这三类框架的原理及优缺点。

Hybrid技术简介

H5+原生混合开发

这类框架主要原理就是将APP的一部分需要动态变动的内容通过H5来实现，通过原生的网页加载控件WebView (Android)或WKWebView（iOS）来加载（以后若无特殊说明，我们用WebView来统一指代android和iOS中的网页加载控件）。这样一来，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。我们称这种h5+原生的开发模式为**混合开发** ，采用混合模式开发的APP我们称之为**混合应用**或**Hybrid APP** ，如果一个应用的大多数功能都是H5实现的话，我们称其为**Web APP** 。

目前混合开发框架的典型代表有：Cordova、Ionic 和微信小程序，值得一提的是微信小程序目前是在webview中渲染的，并非原生渲染，但将来有可能会采用原生渲染。

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，都需要原生去做。而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就成为了JavaScript与原生API之间通信的桥梁，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，我们把依赖于WebView的用于在JavaScript与原生之间通信并实现了某种消息传输协议的工具称之为**WebView JavaScript Bridge**, 简称 **JsBridge**，它也是混合开发框架的核心。

示例：JavaScript调用原生API获取手机型号

下面我们以Android为例，实现一个获取手机型号的原生API供JavaScript调用。在这个示例中将展示JavaScript调用原生API的流程，读者可以直观的感受一下调用流程。我们选用笔者在Github上开源的dsBridge作为JsBridge来进行通信。dsBridge是一个支持同步调用的跨平台的JsBridge，此示例中只使用其同步调用功能。

1. 首先在原生中实现获取手机型号的API `getPhoneModel`

```
class JSAPI {
 @JavascriptInterface
 public Object getPhoneModel(Object msg) {
  return Build.MODEL;
 }
}
```

2. 将原生API通过WebView注册到JsBridge中

```
import wendu.dsbridge.DWebView
...

DWebView dwebView = (DWebView) findViewById(R.id.dwebview);

dwebView.addJavascriptObject(new JsAPI(), null);
```

3. 在JavaScript中调用原生API

```
var dsBridge = require("dsbridge")

var model = dsBridge.call("getPhoneModel");

console.log(model);
```

上面示例演示了JavaScript调用原生API的过程，同样的，一般来说优秀的JsBridge也支持原生调用JavaScript，dsBridge也是支持的，如果您感兴趣，可以去github dsBridge项目主页查看。

现在，我们回头来看一下，混合应用无非就是在第一步中预先实现一系列API供JavaScript调用，让JavaScript有访问系统的能力，看到这里，我相信你也可以自己实现一个混合开发框架了。

总结

混合应用的优点是动态内容是H5，web技术栈，社区及资源丰富，缺点是性能不好，对于复杂用户界面或动画，WebView不堪重任。

React Native和Weex

本篇主要介绍一下 **JavaScript开发+原生渲染**的跨平台框架原理。

React Native (简称RN)是Facebook于2015年4月开源的跨平台移动应用开发框架，是Facebook早先开源的JS框架 React 在原生移动应用平台的衍生产物，目前支持iOS和Android两个平台。RN使用Javascript语言，类似于HTML的JSX，以及CSS来开发移动应用，因此熟悉Web前端开发的技术人员只需很少的学习就可以进入移动应用开发领域。

由于RN和React原理相通，并且Flutter也是受React启发，很多思想也都是相通的，万丈高楼平地起，我们有必要深入了解一下React原理。React是一个响应式的Web框架，我们先了解一下两个重要的概念：DOM树与响应式编程。

DOM树与控件树

文档对象模型（Document Object Model，简称DOM），是W3C组织推荐的处理可扩展标志语言的标准编程接口，一种独立于平台和语言的方式访问和修改一个文档的内容和结构。换句话说，这是表示和处理一个HTML或XML文档的标准接口。简单来说，DOM就是文档树，与用户界面控件树对应，在前端开发中通常指HTML对应的渲染树，但广义的DOM也可以指Android中的XML布局文件对应的控件树，而术语**DOM操作**就是指直接来操作渲染树（或控件树）， 因此，可以看到其实DOM树和控件树是等价的概念，只不过前者常用于Web开发中，而后者常用于原生开发中。

响应式编程

React中提出一个重要思想：状态改变则UI随之自动改变，而React框架本身就是响应用户状态改变的事件而执行重新构建用户界面的工作，这就是典型的**响应式**编程范式，下面我们总结一下React中响应式原理：

- 开发者只需关注状态转移（数据），当状态发生变化，React框架会自动根据新的状态重新构建UI。
- React框架在接收到用户状态改变通知后，会根据当前渲染树，结合最新的状态改变，通过Diff算法，计算出树中变化的部分，然后只更新变化的部分（DOM操作），从而避免整棵树重构，提高性能。

值得注意的是，在第二步中，状态变化后React框架并不会立即去计算并渲染DOM树的变化部分，相反，React会在DOM的基础上建立一个抽象层，即**虚拟DOM**树，对数据和状态所做的任何改动，都会被自动且高效的同步到虚拟DOM，最后再批量同步到真实DOM中，而不是每次改变都去操作一下DOM。为什么不能每次改变都直接去操作DOM树？这是因为在浏览器中每一次DOM操作都有可能引起浏览器的重绘或回流：

1. 如果DOM只是外观风格发生变化，如颜色变化，会导致浏览器重绘界面。
2. 如果DOM树的结构发生变化，如尺寸、布局、节点隐藏等导致，浏览器就需要回流（及重新排版布局）。

而浏览器的重绘和回流都是比较昂贵的操作，如果每一次改变都直接对DOM进行操作，这会带来性能问题，而批量操作只会触发一次DOM更新。

> 思考题：Diff操作和DOM批量更新难道不应该是浏览器的职责吗？第三方框架中去做合不合适？
>
> 此处需要有一张插图

React Native

上文已经提到React Native 是React 在原生移动应用平台的衍生产物，那两者主要的区别是什么呢？其实，主要的区别在于虚拟DOM映射的对象是什么？React中虚拟DOM最终会映射为浏览器DOM树，而RN中虚拟DOM会通过 JavaScriptCore 映射为原生控件树。

JavaScriptCore 是一个JavaScript解释器，它在React Native中主要有两个作用：

1. 为JavaScript提供运行环境。
2. 是JavaScript与原生应用之间通信的桥梁，作用和JsBridge一样，事实上，在iOS中，很多JsBridge的实现都是基于 JavaScriptCore 。

而RN中将虚拟DOM映射为原生控件的过程中分两步：

1. 布局消息传递；将虚拟DOM布局信息传递给原生；
2. 原生根据布局信息通过对应的原生控件渲染控件树；

至此，React Native 便实现了跨平台。 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好很多，同时React Native是Web开发技术栈，也只需维护一份代码，同样是跨平台框架。

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，最大的不同是语法层面，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。与 React 不同，在 Rax 中 JSX 是必选的，它不支持通过其它方式创建组件，所以学习 JSX 是使用 Rax 的必要基础。而React Native只支持JSX语法。

快应用

快应用是华为、小米、OPPO、魅族等国内9大主流手机厂商共同制定的轻量级应用标准，目标直指微信小程序。它也是采用JavaScript语言开发，原生控件渲染，与React Native和Weex相比主要有两点不同：

1. 快应用自身不支持Vue或React语法，其采用原生JavaScript开发，其开发框架和微信小程序很像，值得一提的是小程序目前已经可以使用Vue语法开发（mpvue），从原理上来讲，Vue的语法也可以移植到快应用上。
2. React Native和Weex的渲染/排版引擎是集成到框架中的，每一个APP都需要打包一份，安装包体积较大；而快应用渲染/排版引擎是集成到ROM中的，应用中无需打包，安装包体积小，正因如此，快应用才能在保证性能的同时做到快速分发。

总结

JavaScript开发+原生渲染的方式主要优点如下：

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

不足：

1. 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。
2. JavaScript为脚本语言，执行时需要JIT(Just In Time)，执行效率和AOT(Ahead Of Time)代码仍有差距。
3. 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；除此之外，其控件系统也会受到原生UI系统限制，例如，在Android中，手势冲突消歧规则是固定的，这在使用不同人写的控件嵌套时，手势冲突问题将会变得非常棘手。

QT Moblie与Flutter

在本篇中，我们看看最后一种跨平台技术：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。注意，自绘引擎解决的是UI的跨平台问题，如果涉及其它系统能力调用，依然要涉及原生开发。这种平台技术的优点如下：

1. 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。

2. 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：

1. 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。

也许你已经猜到Flutter就属于这一类跨平台技术，没错，Flutter正是实现一套自绘引擎，并拥有一套自己的UI布局系统。不过，自绘制引擎的思路并不是什么新概念，Flutter并不是第一个尝试这么做的，在它之前有一个典型的代表，即大名鼎鼎的QT。

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。但是，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简介

“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：

- 2018年2月，Flutter发布了第一个Beta版本，同年五月， 在2018年Google I/O 大会上，Flutter 更新到了 beta 3 版本。
- 2018年6月，Flutter发布了首个预览版本，这意味着 Flutter 进入了正式版（1.0）发布前的最后阶段。

观其发展，在2018年5月份，Flutter 进入了 GitHub stars 排行榜前 100 名，已有 27k star。而今天(2019年5月29日)，已经有65K的Star。经历了短短2年多的时间，Flutter 生态系统得以快速增长，由此可见，Flutter在开发者中受到了热烈的欢迎，其未来发展值得期待！

现在，我们来和QT mobile做一个对比：

1. 生态；从Github上来看，目前Flutter活跃用户正在高速增长。从Stackoverflow上提问来看，Flutter社区现在已经很庞大。Flutter的文档、资源也越来越丰富，开发过程中遇到的很多问题都可以在Stackoverflow或其github issue中找到答案。
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，从今年上半年Flutter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；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，Flutter未来如何，心中自有定论。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！

本章总结

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：

H5+原生	WebView渲染	一般	高	✔	Cordova、Ionic
JavaScript+原生渲染	原生控件渲染	好	高	✔	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

上表中**开发语言**主要指UI的开发语言，**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，后续会介绍。