

硕 士 研 究 生 读 书 报 告



题目 React Native在移动端开发中的应用研究

作者姓名 俞晨昊

作者学号 21651191

指导教师 李启雷

学科专业 软件工程

所在学院 软件学院

提交日期 二○一七年 一月

The Research on Application of React Native in mobile terminal development

Major Subject: Software Engineering

Advisor: Qilei Li

By

Chenhao Yu

Zhejiang University, P.R. China

2017

摘要

随着移动互联网的飞速发展，移动设备与平台也在不断更新迭代，多种平台与操作系统共存。对用户而言，多了一种选择，而对于开发者而言，则大大增加了开发成本。为了实现相同的功能，为每个平台单独开发应用无 疑是一种非常耗费人力资源的解决方案;而基于 Web 的跨平台开发方案并不能保证在不同平台提供统一流畅的用户体验。因此降低开发成本、保证应用体验是移动平台开发一直追求和关注的两个核心点。本文论述了 React Native 相比其他跨平台解决方案的优势与不同，讲述了 React Native 的由来以及它的运行原理，并阐述了它 的优缺点。

**关键词**：移动互联网;跨平台;React Native

Abstract

With the rapid evolution of internet，mobile devices and platforms are updating and iterating more fre- quently than ever，multiple platforms coexist. It benefits the common users but not the developers，because multiple platforms also mean even more resources to invest in development. In the meantime，web based development framework cannot assure a unified experience in different platforms. Reduce the cost of development and the assurance of user ex- perience is the two key points that developers are pursuing. This paper discusses the advantages and differences of React Native compared to other cross platform solutions, tells the origin of React Native and its operating principle, and expounds its advantages and disadvantages.

**Keywords：**Mobile internet;Multi-platform;React native;

1引言

移动端的操作系统目前呈现两极分化，Android 与 iOS 是目前最普及的两个移动操作系统。然而在为它们开发同一个应用程序时，通常都需要两套开发人 员来维护两个不同的代码仓库。除了 React Native 外比较流行的解决办法是使用 Html5等Web技术来实现“Write once, Run everywhere”。然而由于设备与操作系统的差异性，基于 Web 的界面与交互在两个平台上的体验并不一致且相差甚远。因此它只解决了开发成本的问题，并没有解决保证应用体验的问题。然而 React Native 则很好的同时解决了这两个问题。

另外值得一提的是在 Android 与 iOS 平台都可以实现代码的动态加载，与 Web 同样灵活。即在程序运行的过程中更新业务代码，可以用来实现 bug 修复或应用升级等功能。

Android 依托于 Java 的 Classloader，而ios依托于Objective-c的动态性。React Native 主要使用 JS 作为开发语言，也非常好的支持了动态加载机制，但缺点是打包后的 JS 代码过多，不利于在质量较低的移动网络下进行传输。

本文主要介绍了 React Native 的概念，并且详细介绍了它的运行机制。

**2 React Native 的概念**

React Native是 Facebook 在 2015 年发布的一套基于JavaScript的开源框架。React Native 结合了 Web 应用和 Native 应用的优势，可以使用 JavaScript 来开发 iOS 和 Android 原生应用。在 JavaScript 中用 React 抽象操作系统原生的 UI 组件，代替 DOM 元素来渲染界面。

2.1 理解React

前端界总是喜欢创造新的概念，仿佛谁说的名词更晦涩，谁的水平就越高。如果你听到 React 这个概念一脸困惑的话，只要记住以下定义即可：

React 是一套可以用简洁的语法高效绘制 DOM 的框架。

作为前端新手，我对前端的理解是这样的：

1. 用 HTML 创建 DOM，构建整个网页的布局、结构
2. 用 CSS 控制 DOM 的样式，比如字体、字号、颜色、居中等
3. 用 JavaScript 接受用户事件，动态的操控 DOM

在这三者的配合下，几乎所有页面上的功能都能实现。但也有比较不理想的地方，比如我想动态修改一个按钮的文字，我需要这样写：

<button type="button" id="button" onclick="onClick()">old button</button>

然后在 JavaScript 中操作 DOM：

<script>

function onClick() {

document.getElementById('button').innerHTML='new button';

}

</script>

可以看到，在 HTML 和 JavaScript 代码中，id 和 onclick 事件触发的函数必须完全对应，否则就无法正确的响应事件。如果想知道一个 HTML 标签会如何被响应，我们还得去 JavaScript 代码中查找，这种原始的配置方式非常不方便。

随着 FaceBook 推出了 React 框架，这个问题得到了大幅度改善。我们可以把一组相关的 HTML 标签，也就是 app 内的 UI 控件，封装进一个组件(Component)中，例如：

var MyComponent = React.createClass({

handleClick: function () {

this. refs. myTextInput.focus();},

render: function() {

return (

<div>

<input type="text" ref="myTextInput" />

<input type="button" value="Focus the text input" onClick={this.handleClick} />

</div>

);

}

});

在这里我们看到，JavaScript 代码里面出现了 HTML 的语法，这就是 React 的奥妙。这种语法被称为 JSX，它是一种 JavaScript 语法拓展。JSX 允许我们写 HTML 标签或 React 标签，它们终将被转换成原生的 JavaScript 并创建 DOM。在 React 框架中，除了可以用 JavaScript 写 HTML 以外，我们甚至可以写 CSS。

而我们所谓的React“高效”，是因为 React 独创了 Virtual DOM 机制。Virtual DOM 是一个存在于内存中的 JavaScript 对象，它与 DOM 是一一对应的关系，也就是说只要有 Virtual DOM，我们就能渲染出 DOM。

当界面发生变化时，得益于高效的 DOM Diff 算法，我们能够知道 Virtual DOM 的变化，从而高效的改动 DOM，避免了重新绘制 DOM。

当然，React 并不是前端开发的全部。从之前的描述也能看出，它专注于 UI 部分，对应到 MVC 结构中就是 View 层。要想实现完整的 MVC 架构，还需要 Model 和 Controller 的结构。

2.2 理解react native

由于 App审核周期的限制，如何动态的更改 app 成为了永恒的话题。无论采用何种方式，我们的流程总是可以归结为以下三部曲：“从 Server 获取配置 --> 解析 --> 执行native代码”。

很多时候，我们自觉或者不自觉的利用 JSON 文件实现动态配置的效果，它的核心流程是：

1. 通过 HTTP 请求获取 JSON 格式的配置文件。
2. 配置文件中标记了每一个元素的属性，比如位置，颜色，图片 URL 等。
3. 解析完 JSON 后，我们调用 Objective-C 的代码，完成 UI 控件的渲染。

通过这种方法，我们实现了在后台配置 app 的展示样式。从本质上来说，移动端和服务端约定了一套协议，但是协议内容严重依赖于应用内要展示的内容，不利于拓展。也就是说，如果业务要求频繁的增加或修改页面，这套协议很难应付。

最重要的是，JSON 只是一种数据交换的格式，说白了，我们就是在解析文本数据。这就意味着它只适合提供一些配置信息，而不方便提供逻辑信息。举个例子，我们从后台可以配置颜色，位置等信息，但如果想要控制 app 内的业务逻辑，就非常复杂了。我们只是在解析字符串，它完全不具备运行和调试的能力。

所以移动端通过 JSON 文件传递信息有它特有的不足之处：只能传递配置信息，无法表达逻辑。

而 React 在前端取得突破性成功以后，JavaScript 布道者们开始试图一统三端。他们利用了移动平台能够运行 JavaScript 代码的能力，并且发挥了 JavaScript 不仅仅可以传递配置信息，还可以表达逻辑信息的优点。

当痛点遇上特点，两者一拍即合，于是，一个基于 JavaScript，具备动态配置能力，面向前端开发者的移动端开发框架，React Native，诞生了！

看到了么，这是一个面向前端开发者的框架。它的宗旨是让前端开发者像用 React 写网页那样，用 React Native 写移动端应用。这就是为什么 React Native 自称：“Learn once，Write anywhere!”而非很多跨平台语言，项目所说的：“Write once, Run anywhere!”

React Native 希望前端开发者学习完 React 后，能够用同样的语法、工具等，分别开发安卓和 iOS 平台的应用并且不用一行原生代码。

如果用一个词概括 React Native，那就是：Native 版本的 React。

因此它同时解决了以下的两个问题:

一、 减少开发成本

无论是 Android 还是 iOS 平台，React Native 都使用 JavaScript 作为主要的开发语言，同时它也使用了 React 来简化和加速开发流程。由于 Android 与 iOS 中 的系统 UI 组件大部分都相似(如文字、图片、列表 等)，基础功能(如本地存储、动画、通知)也基本类似，因此 React Native 将这些组件统一抽象为一个 Module，开发者不用再关心到底当前编写的 Module 运行在哪个平台上，只需要关注本身的业务逻辑即可。

而对于Android 与iOS 所各自特有的概念，React Native进行单独定义，如Android 中的Toolbar、iOS 中的TabBar等等。这样开发者就可以针对不同平台的特性，在现有代码基础上稍作修改以适配不同的移动 操作系统。Android 软件的设计与其他系统中软件设计基本类似，可能在测试方法、测试用例的编写上有所不同，但是毕竟开发流程并不同意。

Facebook在提出React Native的时候提出“Learn once，write anywhere”，即不同的平台代码不会完全相同，但是会很相似。即也可以保证一套开发人员即可完成多平台业务场景的开发与实现，相较于传统的开发方式有很大改观。

二、 提供一致的用户体验

一般来讲，在各自平台使用原生的 API 开发的应用是运行效率最高的，即用户体验最好的。而基于 Web 形式的应用既需要加载时间，又需要有较强的处 理器进行 DOM 渲染，因此在不同平台、不同设备上的体验非常不一致。虽然说 JQuery 等开源方案经过优化可以达到比较好的效果，但是离原生应用差距还是 较大。举例来说 PhoneGap属于较为成熟的基于 Web 的跨平台解决方案，它在 iOS 上的流畅度和加载时间要远远好于 Android，或 Android 中只能处理一些简单的静态逻辑。如果仅仅使用 PhoneGap为 iOS 开发， 就失去了跨平台解决方案的意义。

React Native 虽然使用 JavaScript 作为开发语言， 但是它并不将界面 UI 渲染到 Web 上，而是使用 JavaScript 解释器，将 JS 代码解释为原生的组件再渲染到界面上。因此，从用户体验上来讲，React Native 的渲染效率应该与原生应用是一致的。

**3 React Native 运行原理**

这里选取 iOS 作为分析对象。

首先要明白的一点是，即使使用了 React Native，我们依然需要 UIKit 等框架，调用的是 Objective-C 代码。总之，JavaScript 只是辅助，它只是提供了配置信息和逻辑的处理结果。React Native 与 Hybrid 完全没有关系，它只不过是以 JavaScript 的形式告诉 Objective-C 该执行什么代码。

其次，React Native 能够运行起来，全靠 Objective-C 和 JavaScript 的交互。对于没有接触过 JavaScript 的人来说，非常有必要理解 JavaScript 代码如何被执行。

我们知道 C 系列的语言，经过编译，链接等操作后，会得到一个二进制格式的可执行文，所谓的运行程序，其实是运行这个二进制程序。

而 JavaScript 是一种脚本语言，它不会经过编译、链接等操作，而是在运行时才动态的进行词法、语法分析，生成抽象语法树(AST)和字节码，然后由解释器负责执行或者使用 JIT 将字节码转化为机器码再执行。整个流程由 JavaScript 引擎负责完成。

3.1 JS 与原生模块相互调用

在 iOS 中，React Native 使用自带的 JavaScriptCore 作为 JS 的解析引擎，但并没有用到 JavaScriptCore 提供的一些可以让 JS 与 OC 互调的特性，而是自己实现了一套机制，这套机制可以通用于所有 JS 引擎上，这么做应该也是考虑到了项目的灵活性与 iOS 安全机制的问题。

实际上普通的JS与原生代码的调用非常简单，无论是Android还是iOS，原生代码可以向JS传递消息，如iOS中webview提供的 stringByEvaluatingJava ScriptFromString方法可以直接执行一段 JS 脚本，并且可以获取它的返回值。这个返回值其实就相当于 JS 向原生代码传递消息。React Native 就是以此为基础，通过模块配置表和响应流程实现了原生代码与 JS 代码的无缝衔接调用。

3.2 模块配置表

在 React Native 初始化的时候，为了使JS 知道可调用的原生模块 Module 方法与命名，需要将一份 Module 的配置表传递给JS，配置表里包含了所有的模块和模块中的方法信息。

React Native 框架中提供了一个叫做 RCTBridge Module 的接口。在 React Native 初始化的时候，可以通过runtime接口objc\_getClassList获取所有的类，然后逐个判断是否实现了RCTBridgeModule接口，就可以找到所有的模块类了。一个模块中有很多方法，一些可以暴露给 JS 调用，另一些则不需要。需要暴露的方法使用 RCT\_EXPORT\_METHOD 宏定义包裹，其实是将方法名在编译时重新定义为有固定前缀的方法，方便查找并记录。

因此在原生代码中定义好了接口与函数，React Native 就会根据运行时环境获取所有的模块与其方法生成模块配置表。这个配置表在原生代码与JS代码之间都维持一份相同的配置，便于原生代码与JS代码相互调用。

3.3 事件传递响应流程

React Native 的主要核心是自己实现了一套 JS 与原生代码相互调用的机制，这也是与其他基于Web的跨平台方案不同的地方。使用 JS 写好业务代码后，由 React Native 进行解释，偏向效率的逻辑执行、UI 渲染都会在原生阶段进行处理，因此 React Native 的执行效率要比其他的跨平台方案高很多。

JS 与原生代码的调用流程比较负责，如下所示:

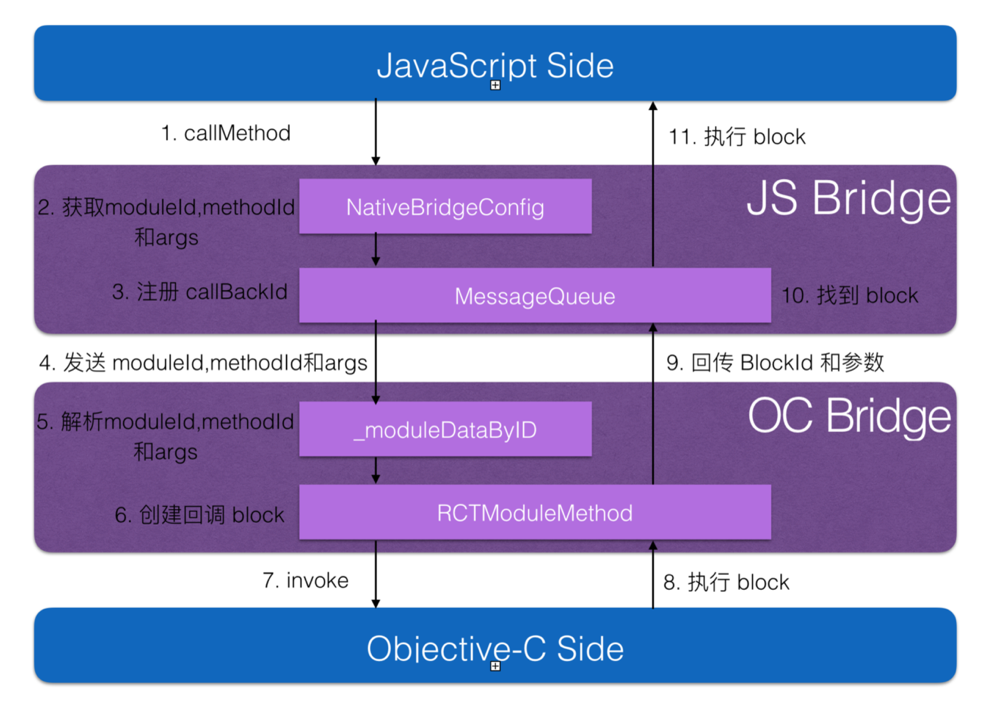


图1 交互图

1、JS 端调用原生模块中的某个方法

2、将调用分解为模块名、方法名、参数，传递给MessageQueue 进行处理，生成 callbackId

3、将模块名、方法名、参数对应模块配置表，转换成模块 id、方法 id、参数 id

4、原生模块根据各个 id 确定被调用的方法与参数

5、执行原生代码逻辑，将执行结果进行回调

6、JS 端根据 callbackId 接收执行结果

其中需要特别注意的是，JS 并不会主动传递数据给 OC。在调用 OC 方法时，会把模块 id、方法 id 放入到一个队列中。等原生代码过来调用 JS 的任意方法时，将这个队列返回再执行这个队列中需要执行的方法。所以 React Native 是一种基于事件响应机制的框架，在没有操作时不会进行无效的循环执行。

**4 小结**

我们在本文中主要介绍了React native发展由来，以及它的运行原理。这里略微总结一下它的优缺点。

React native的优点：

1. 复用了 React 的思想，有利于前端开发者涉足移动端。
2. 能够利用 JavaScript 动态更新的特性，快速迭代。
3. 相比于原生平台，开发速度更快，相比于 Hybrid 框架，性能更好。

缺点：

1. 做不到 Write once, Run everywhere，也就是说开发者依然需要为 iOS 和 Android 平台提供两套不同的代码，比如参考官方文档可以发现不少组件和API都区分了 Android 和 iOS 版本。即使是共用组件，也会有平台独享的函数。
2. 不能做到完全屏蔽 iOS 端或 Android 的细节，前端开发者必须对原生平台有所了解。加重了学习成本。对于移动端开发者来说，完全不具备用 React Native 开发的能力。
3. 由于 Objective-C 与 JavaScript 之间切换存在固定的时间开销，所以性能必定不及原生。比如目前的官方版本无法做到 UItableview(ListView) 的视图重用，因为滑动过程中，视图重用需要在异步线程中执行，速度太慢。这也就导致随着 Cell 数量的增加，占用的内存也线性增加。

可以说，React native是利用脚本语言进行原生平台开发的一次成功尝试，降低了前端开发者入门移动端的门槛，在一定业务场景下具有独特的优势，但是能不能取代原生平台开发还是一个未知数。

参考文献

[1] Facebook.React Native[OL].[2015-12-19] http://facebook.github.io/react-native.

[2] Facebook.React JS[OL].[2014-07-23].http://facebook.github.io/react/.

[3]潘婷婷.React Native在APP开发中的应用研究[J]，无线互联科技，2016-10-1

[4]戴炳荣,王宇飞;,王凌,陆奇峰,徐承宇.基于PhoneGap跨平台移动开发框架的研究与应用[J],现代计算机(专业版),2016-09-25

[5]金昱东. 一种基于 ReactNative 的代码压缩与动态加载的实现方法[J].软件，2016，37(02):81-84