浙江大學

# 硕 士 研 究 生 读 书 报 告


题目 基于 ReactNative 开发移动应用的研究

作者姓名 李 超

作者学号 21651107

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 二〇一七 年 一 月

# Research on Developing Mobile Application Based on ReactNative

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software Engineering

Advisor: Li Qilei

By

Superlee

Zhejiang University, P.R. China

2017

# 目录

# 摘要

本文重点讨论如何使用 Facebook 的框架 React Native 进行移动应用的开发，主要技术为 JavaScript，React Native 需要一个 JS 运行环境，在 iOS 中直接使用内置的 javascriptcore,在 Android 使用 webkit.org 官方开源的 jsc.so。React Native 能够在 JavaScript 和 React 的基础上获得完全一致的开发体验，构建一流的原生 APP,着力于提高多平台开的效率，并且仅需学习一次，编写任何平台(Learn once,write anywhere)。

相比于 Hybird App 或 Webapp,React Native(以下简称 RN)开发移动应用所带来的优势是更快的交互与更流畅的界面效果，贴近于原生的速度，它不使用 webview,摆脱了 webview 所带来的交互瓶颈和性能问题。同时，RN 有很强的扩展性，使用 js 可以自由组合 native 端提供的基本控件，并且不需要担心平台的问题，给移动开发者带来了很大的便利。

**关键词：移动开发,ReactNative，React,JavaScript,CSS**

# Abstract

This article focuses on the use of Facebook's framework React Native for mobile application development, whose main tecknology is JavaScript. React Native requires a JS runtime environment, and use the build-in javascriptcore in iOS, while use the open-source jsc.so of webkit.org in android, getting the same development experience in both JavaScript and React, and RN aims to build the better app that works well in different platform, which means that learning once and writing anywhere.

Compare to Hybird App or Webapp, react native(rn) can bring the app moreFaster interaction and smoother interface effects, similar to native app. Also, RN has a very strong scalability by combining js with the native, no more should the developers worry about the platform, which is a big gift for developer.

**Keywords: Mobile development, ReactNative, React, JavaScript, CSS**

# 1. 引言

手机在人们的生活中日益占据重要的位置，构成手机使用环境的 APP 更是离不开人们的生活，其中的两位主角便是目前市场上引领风潮的 android 和苹果，与两者分别相关的 JAVA 语言和 Object-C 语言目前在编程语言使用范围也相对广泛，对于 app 开发人员来说，同样的逻辑和业务，为了满足多平台，需要重新基于不同平台进行开发，极大花费了开发人员的时间和精力，也增加了软件的开发成。而能够支持跨平台的 hybridapp 和 webapp 虽然在一定程度上减小平台的影响，但是带来的却是性能的流失，比如列表 listview，滑动列表进行浏览时，在后两者的表现上有明显的卡顿和延迟，严重影响用户体验。

在这种环境下，一个基于 JS 的 React Native 框架问世，同时支持 android 端和 iOS 端，这个开源框架为前端工作人员转战移动端开发打开了一扇大门，从未学习过移动应用开发的网页开发开程师也可以用 React Native 构建出一个流畅的 android 和 iOS 应用，而进行过移动开发的工程师则更加得心应手，因为他不再需要特别去关注不同平台的影响。并且 RN 开发社区活跃度日趋增长，各种 NPM 包的完善、第三方插件的增多使得 React Native 的前途一片光明。

# 2. 什么是 React Native

## 2.1 React Native 起源

React Native(以下简称为 RN) 是 facebook 公司发布的开源框架[1]，面向移动开发，包括 iOS 和 Android，React Native 使得 JavaScript 能够开发真正的 APP 应用，它不仅有着与原生应用相媲美的体验，同时拥有着 Web 应用的优势和开发效率。React Native 提倡的是 Learn once, Write anywhere,也就是说，只需要一次的学习成本，就能够在各大平台上开发应用，不得不说这是一个大胆且有意义的框架。React Native 鲜明的特点就是组件化，一个应用由多个自定义组件构成；同时为了更高的效率，React Native 采用了内存 Dom tree Diff 计算，生成虚拟 dom，然后再渲染生成真正的 dom 结构，优化了 view 的渲染效率和体验。使用 JavaScript 开发跨终端的应用是未来的趋势。

## 2.2 为什么需要 react native

RN 的兴起需要从 app 应用开始探讨，目前主流的应用大体分为三类：Native App,Web App,Hybird App.它们之间的关系如图所示：

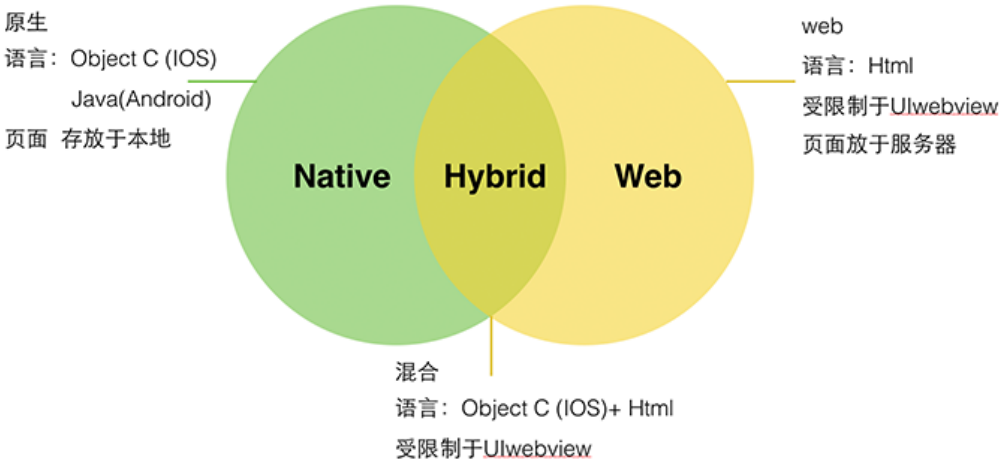


图 2.1 app 的分类

对于 Native App，它的优缺点如下：

① 性能好（优点）

② 开发成本高（缺点）

③ 升级困难（缺点）

对于 Web APP，它的优缺点如下:

① 跨平台（优点）

② 版本升级容易（优点）

③ 无法调用系统级 API(缺点)

④ 用户留存度低，只作为临时入口（缺点）

⑤ 性能差（缺点）

对于 Hybird App,它选取了前两者的折中方案，但也有它自身的缺点，因为它的实现机制是借用 Webview，在其中绘制界面，实现动画，资源消耗比较大

① 保留了 Native APP 和 Web App 的优点（优点）

② 页面渲染效率低，资源消耗大（缺点）

所以，RN 诞生了，它的设计初衷是既保留流畅的用户体验，又保留 React 开发效率，这里我们只作概述，详细实现原理后文讨论。

① 丢弃了 WebView

② 复用 React,将 Dom 结构的改变通过 diff 算法处理后，由 js 传递给 native 进行底层视图布局

③ 对 js 暴露底层常用的 UI 组件，js 层可直接对这些组件进行布局

④ css-layout 引擎，前端开发者可以继续写熟悉的 css 语法，引擎进行底层层布局的转化。

通过这样的四个步骤，RN 实现了其初衷，在获得高性能的同时，不以牺牲资源为代价，同时降低了 app 的开发学习路线。

# 3. React Native 和 React

## 3.1 区别和联系

说起 RN(React Native),不得不说 react，因为 RN 是在 react 基础上设计诞生的，都起源于 facebook 内部，最开始只有 react,当时 facebook 公司工程师对市场上所有 JavaScript MVC 框架都不满意，就决定自己开发写一套，用于 Instagram 的网站开发，并命名为 react,在实际使用中发现 react 很好用，于是开源了该框架，并构建了一个完整的社区用于该框架的维护和完善，这个项目本身也越来越大，从最早的 UI 引擎变成了一整套前后端通吃的 Web App 解决方案。衍生的 React Native 项目，目标更是宏伟，希望用写 Web App 的方式去写 Native App，于是诞生了 react native，并独立了出来，并且就语法成面而言，两者都基于 virtual dom 渲染生成，所以，接下来我们介绍 react 的特性，也即是 react native 的主要特性

## 3.2 react 主要特性

react 有四大特性，也即是 react native 的主要特性（语言成面），特性如下：

① 在 JS 里声明描述 UI,使用 JSX 语法取代 HTML 模板

为了方面构建 UI 和自定义组件，Facebook 创造 JSX 语法，用来代替常规的 HTML 模板，我们可以直接在 js 文件中来使用 JSX，这种语法结合了 HTML 和 JavaScript 的优势，既能像平常一样使用 HTML，同时又能在 HTML 中使用强大的 JavaScript 语言。相当于我们可以把 View 和 JavaScript 逻辑写在同一个文件里面,如下代码所示

```
var myDivElement = <div className="foo" />;

ReactDOM.render(myDivElement,
document.getElementById('example'));
```

② 虚拟 DOM 取代物理 DOM 作为操作对象，封装了 DOM 的事件系统

前端的 dom 一直以来就是一个讨论比较多的话题，因为 html 页面渲染速度的快慢、交互的流畅性等都与 dom 元素有关，比如 jquery,在 dom 元素上做了一层封装，开发者直接调用 jquery 提供的 api 饥渴直观的

操作 dom 元素，但是 Facebook 认为开发者直接操作 DOM 还不够好，影响交互的效率，所以创造了一个虚拟 DOM 的概念，当使用 react 时，开发者不再需要去直接操作 dom 元素，它用一种更快的内置仿造的 DOM 来计算差异，计算出效率最高的 DOM 改变，然后自动去更新 DOM。而且还封装了事件系统，React 的高明之处就是这个事件系统对于开发者而言，并没有新的接口或者其他，在加快了速度的同时，还减低了学习成果。

③ 单向数据流动

不同于 angular 和 vue 的双向数据绑定，在 React 中，数据的流向是单向的，即从父节点传递到子节点，因为组件是简单而且易于把握的，他们只需从父节点获取 props 渲染即可，如果顶层组件的某个 prop 改变了，React 会递归的向下遍历整棵组件树，重新渲染所有使用这个属性的组件，并且，在 react 组件内部还有自己的状态，这些状态只能在组件内修改（调用 setState 方法）

④ 组件和基于组件的设计过程

在 react 中，组件的概念就是状态机，React 将用户界面看做简单的状态机，当组件处于某个状态时，那么就输出这个状态对应的界面。比如有数据时是一种状态，没有数据又是另一种状态，react 会根据不同状态进行 dom 元素的渲染，更新某个组件的状态，然后输出基于新状态的整个界面。React 负责以最高效的方式去比较两个界面并更新 DOM 树。通过这种方式，就很容易去保证界面的一致性，代码样例如下：

```
var LikeButton = React.createClass({
  getInitialState: function() {
    return {liked: false};
  },
  handleClick: function(event) {
    this.setState({liked: !this.state.liked});
  },
  render: function() {
    var text = this.state.liked ? 'like' : 'haven\'t liked';
    return (
      <p onClick={this.handleClick}>
        You {text} this. Click to toggle.
```

```
      </p>
    );
  }
});

ReactDOM.render(
  <LikeButton />,
  document.getElementById('example')
);
```

所有的组件必须返回 render 方法，进行组件的渲染。

## 3.3 组件生命周期及方法调用

在 React 和 React Native 中，组件的生命周期分为三个状态

- Mounting:已插入真实 DOM
- Updating:正在被重新渲染
- Unmounting:已移出真实 DOM

为方便进行 dom 元素的处理，react 为每个状态都提供了两种处理函数，带 will 的函数在进入状态之前调用，带 did 的函数在进入状态之后调用，共计如下 5 中处理函数[4]

- ✓ componentWillMount():组件插入之前
- ✓ componentDidMount()：组件插入之后
- ✓ componentWillUpdate(object nextProps, object nextState)：组件重新渲染之前
- ✓ componentDidUpdate(object prevProps, object prevState)：组件重新渲染之后
- ✓ componentWillUnmount()：组件被移出真实 DOM 之前

同时，React 又提供两种处理特殊状态的函数

- ✓ componentWillReceiveProps(object nextProps)：已加载组件收到新的参数时调用
- ✓ shouldComponentUpdate(object nextProps, object nextState)：组件判断是否重新渲染时调用

状态及对应常用函数如下图所示：

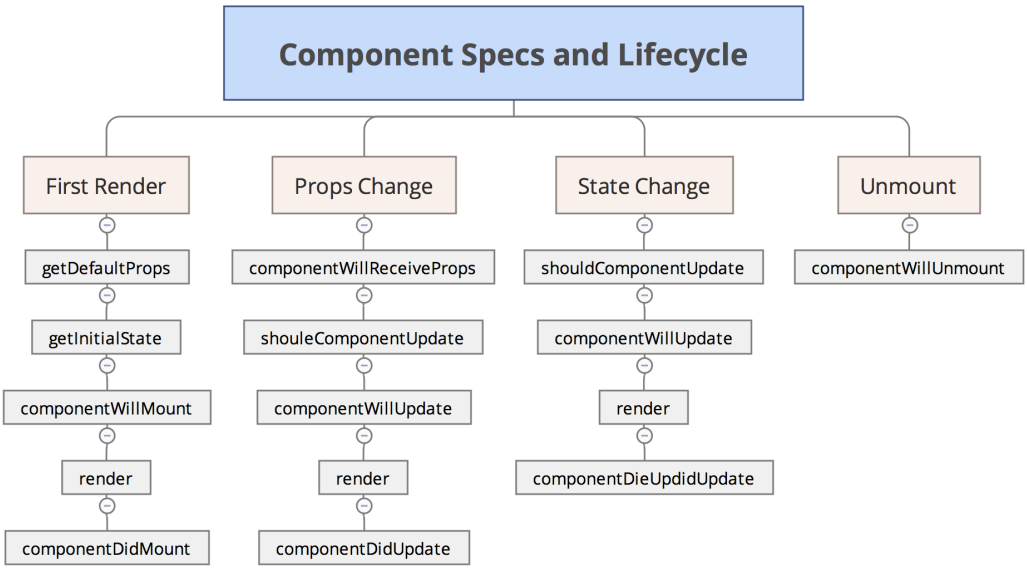


图 3.1 组件状态及常用 API

自定义组件时根据需要在组件生命周期的不同阶段实现不同的逻辑，除了必须实现的 Render 方法外，其他的常用方法包括：

- ✓ getDefaultProps：在组件挂载之前调用一次。返回值将会作为 this.props 的初始值。
- ✓ getInitialState：在组件挂载之前调用一次。返回值将会作为 this.state 的初始值。
- ✓ componentDidMount：在组件第一次 render 之后调用，这时组件对应的 DOM 节点已被加入到浏览器。在这个方法里可以去实现一些初始化逻辑。
- ✓ componentWillUnmount：在 DOM 节点移除之后被调用，进行相关的处理工作。

一个 react 组件完整的生命周期如图 3.2 所示：

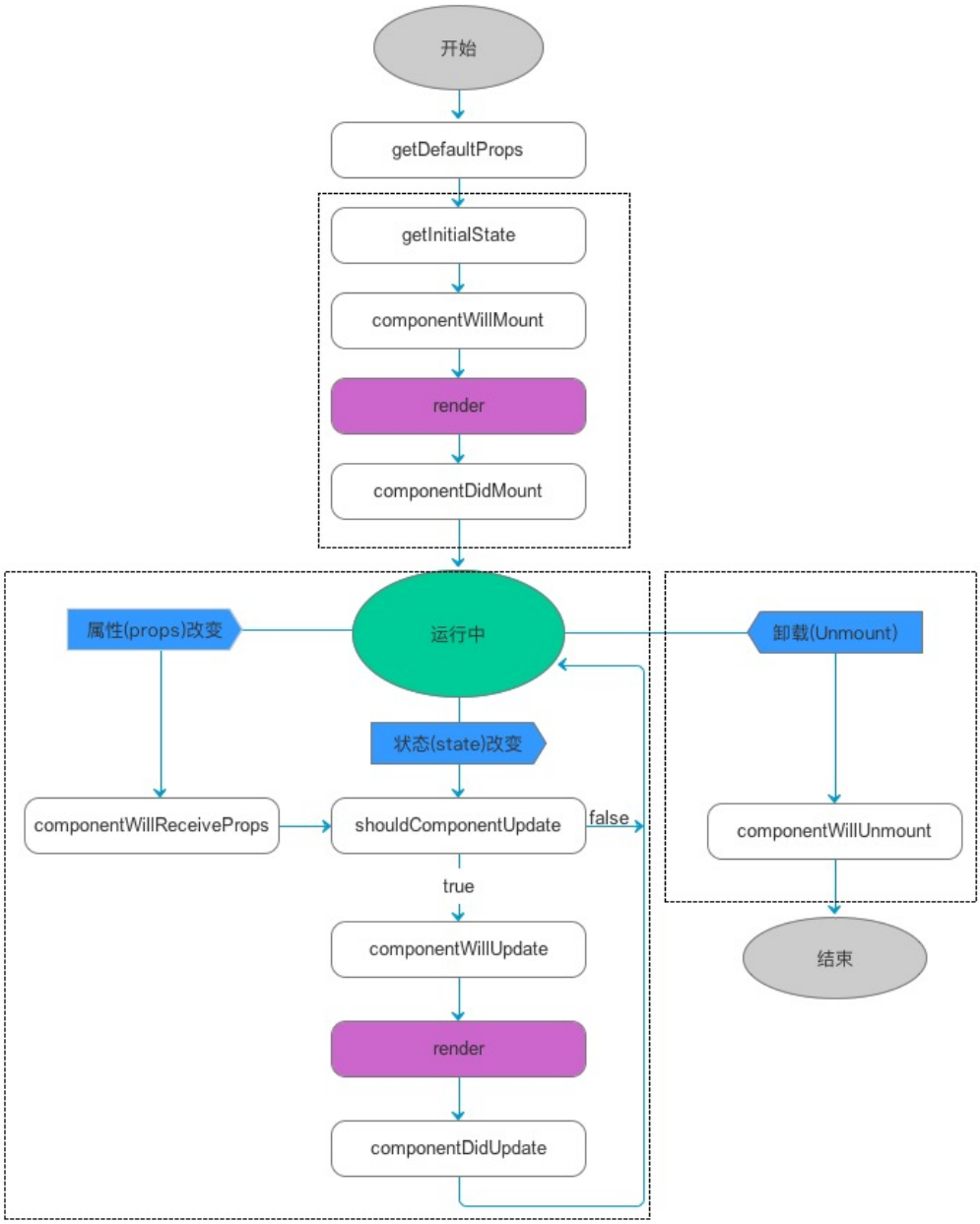


图 3.2 React Native 组件生命周期

前面讨论过生命周期的状态，那么，一个完整的 RN 组件的生命周期可以分为三个阶段：

- 第一阶段：是组件第一次绘制阶段，如图中的上面虚线框内，在这里完成了组件的加载和初始化；
- 第二阶段：是组件在运行和交互阶段，如图中左下角虚线框，这个阶段组件可以处理用户交互，或者接收事件更新界面；
- 第三阶段：是组件卸载消亡的阶段，如图中右下角的虚线框中，这里做一些组件的清理工作。

# 4. React Native 关键技术及解析

使用 RN 开发最大的难题就是设计思想的转变。以前的设计方法论已经不太适用，并且 RN 只是提供了 view 的框架，完整的 app 架构并没有直接提供，需要开发者自己实现。下文我们讨论 RN 中关键的设计理念。

## 4.1 组件化

由于 RN 是基于 React 的，React 是采用 virtual dom 进行页面构造，所以，虚拟 dom 不仅带来了简单的 UI 开发逻辑，也带来了组件化开发的思想。我们先了解一下组件的概念，所谓组件，即封装起来的具有独立功能的 UI 部件。RN 推荐以组件化的方式思考 UI 的构成，将 UI 上每一个功能相对独立的模板定义成组件，然后将小的组件通过组合或者嵌套的方式构成大组件，最终完成整体 UI 的构建。

相比于 MVC 的视图-数据-控制器的分离，组件化的思考方式带来了 UI 功能木块之间的相对分析，开发者从功能角度出发，将 UI 分成不同组件，每个组件被独立分装，只关心自己部分的逻辑，彼此独立。并且，RN（或者说 react）提供的组件封装方式和单向数据流动极大的简化了 app 架构的理解难度。

组件有如下特征：

- ✧ 可组合（Composeable）：一个组件能够很简单的和其它组件一起使用，或者嵌套在另一个组件内部。如果一个组件内部创建了另一个组件，那么说父组件拥有（own）它创建的子组件，通过这个特性，一个复杂的 UI 可以拆分成多个简单的 UI 组件；
- ✧ 可重用（Reusable）：每个组件都是具有独立功能的，只负责自己相关的逻辑，它可以被使用在多个 UI 场景
- ✧ 可维护（Maintainable）：每个最小单位的组件仅仅包含自身的逻辑，更容易被理解和维护；
- ✧ 可测试（Testable）：因为每个组件都是独立的，所以对于各个组件分别测试要比整体 UI 测试简单的多。

在使用 RN 的时候，实现组件化，主要关注三个属性，props、state 和 ref，详细介绍如下：

① props：大多数组件在创建时就可以使用各种参数来进行定制。用于定

制的这些参数就称为 props（属性）,this.props 对象的熟属性与组件的属性意义对应（除 cha 了 this.props.children,它表示组件的所有子节点）,表示那些一旦定义，就不再改变的特性

② state: 组件免不了和用户互动，RN 中将组件看成状态机，一开始有一个初始状态，然后用户交互，导致状态改变，从而重新渲染 UI。this.state 表示那些随着用户交互而产生变化的特性

③ ref: 组件并不是正式的 DOM 节点，而是存在于内存中的一种数据结构，只有当它插入到文档中之后，通过 UI 渲染，才会变成真实的 DOM,通过使用 ref，可以从组件中获取真实的 DOM 节点。需要注意的是，由于 this.refs.[refName]获取到的是真实 DOM，所以必须等到虚拟 DOM 插入文档之后，才能使用这个属性

组件化的思想擅长在负责场景中保证高性能，并且 UI 中最为负责的局部更新部分由 RN 接管，这种方式不仅提高开发效率，而且可以让代码更容易理解，维护和测试。

## 4.2 虚拟 DOM

React 之所以快，是因为它不直接操作 DOM。React 将 DOM 结构存储在内存中，在内存中维护一个快速响应的 DOM 描述，这个 dom 描述其实就是一个 js 数据对象，react 把它称为 virtual dom(虚拟 dom),然后同 render()的返回内容进行比较，计算出需要改动的地方，最后才反映到 DOM 中[4][5]。

举个实际例子，朋友圈的评论点赞系统，用户点赞时，需要更新该评论下的赞，赞的时候需要更新用户赞数量信息及显示用户昵称，我们把它叫做视图重渲染，这时候，react 需要做的是构建虚拟 dom，通过内部属性的改变通知虚拟 dom 进行内存运算，急与旧属性的虚拟 dom 进行一次比较，将计算后的结果直接更新在界面上，不需要我们手动管理视图。

我们提到了 dom 比较，这个是 rn 中核心的算法--DOM Diff 算法。当其中某一部分发生变化时，其实就是对应的某个 DOM 节点发生了变化，在 RN 的界面展示中，构建 UI 界面的思路是由当前状态决定界面，不同的状态对应不同的及诶按，然后由 RN 比较界面的区别，这就需要对 DOM 树进行 Diff 算法分析。

分析的过程如下，即给定任意两棵树，找到最少的转换步骤，但是标准的 Diff 算法复杂度需要 O(n^3),如果按这个复杂度设计算法，显然无法满足性能要求。RN 实现了这一点，它结合界面的特点做出了两个简单的假设，使得 Diff 算法复杂度直接降低到 O(n)[3]。

- 两个相同组件产生类似的 DOM 结构，不同的组件产生不同的 DOM 结构
- 对于同一层次的一组子节点，它们可以通过唯一的 id 进行区分

基于这两点假设，节点不同分为两种情况：

① 节点类型不同

② 节点类型相同，但属性不同

不同于数据结构中树的比较，在 RN 中，树的算法非常简单，两棵树只会对同一层次的节点进行比较，如图 4.1 所示

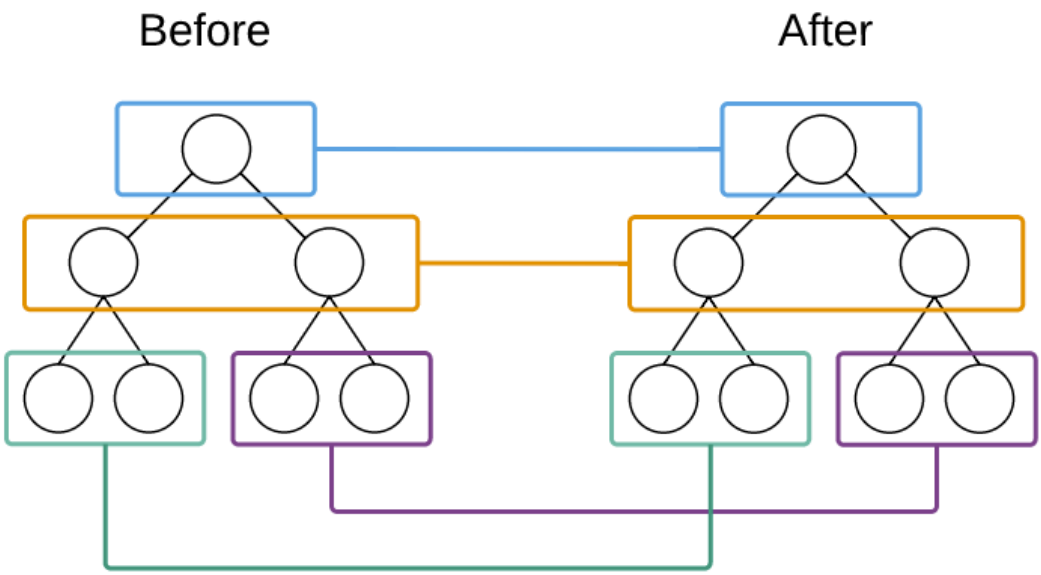


图 4.1 节点比较

RN 只会对相同颜色方框内的 DOM 节点进行比较，即同一个父节点下的所有子节点。当发现节点已经不存在，则该节点及其子节点会被完全删除掉，不会用于进一步的比较。这样只需要对树进行一次遍历，便能完成整个 DOM 树的比较。

虚拟 dom 主要用于解决 app 界面的性能瓶颈问题，类似 cordova 之类的跨平台解决方案性能瓶颈就在 DOM 上，虚拟 dom 就是当 DOM 有更改的时候进行

DOM 渲染，可以有效的减少 DOM 渲染次数，解决这个性能问题。这也是 React Native 的核心思想。

## 4.3 通信机制

以 Android 开发为例，RN 框架主要实现了一套 JAVA 和 JS 通信的方案，实现的是单向调用，Native 线程定期向 JS 线程大区数据，然后转成 JS 的调用预期，最后转交给 Native 对应的调用模块，达到 Java 和 JS 定义的 Module 相互调用的目的

### 4.3.1 JS 调用 Java

JS 通过扩展模块 require('NativeModules')获取 native 模块，然后调用 native 公开的方法。require('NativeModules')实际是获取 MessageQueue 里的 native 模块列表属性，然后使用_genModules 加载所有 native module 到 RemoteModules 数组，其中每一项都是一个映射到 native module 的 JS 对象。JS 端调用完毕后，queue 中数据等待 Native 层通过 bridge 来获取。

### 4.3.2 Java 调用 JS

Java 通过调用 catalystInstance.getJSModule 方法获取 JS 对象(实际是 js 对象在 java 层的映射对象)，然后访问对象方法 runAPPlication，并在 CoreModulesPackage.createJSModules 方法里配置，如果调用 JSModules 对象的方法，则会动态代理跳转到(mBridge).callFunction(moduleId, methodId, arguments);接着调用 ReactBridge 中声明的 JNI 函数,通过 JS 的 require 和 apply 函数拼接一段 JS 代码， 然后用 javascriptCore 的脚本运行接口执行，并得到返回值.

## 4.4 RN 与 Android/iOS 代码交互

讨论完组件化和虚拟 dom 这两大核心思想以及通信机制，这一节讨论 React Native 如何通过 js 与底层 android、iOS 平台进行交互。

RN 需要一个 JS 的运行环境，在 iOS 上直接使用系统内置的 JavaScriptCore,在 Android 平台则使用 webkit.org 开源的 jsc.so.RN 会把 JS 代码编译生成一个 js 文件（如 index.android.js/index.ios.js）,然后 RN 解

释运行这个 js 脚本文件，分两种情况

- js 扩展 API：通过 bridge 调用 native 方法
- UI 界面：映射到 virtual dom 这个虚拟 js 数据结构中，通过 bridge 传递到 native,根据数据属性设置各个对应的真实 native 的 view

以 Android 开发者为例，使用 React Native 进行开发时，RN 是一个普通的 android 程序加上事件响应，事件的触发源主要是 js 命令，然后有如下两个线程进行相关界面的渲染和事件响应：

① UI main thread:这个线程用来渲染 UI 界面，UI thread 在创建一个 APP 的事件循环后，就挂在 looper 等待事件，时间驱动各自对象执行命令。

② JS thread:这个线程相当于底层数据采集器，不断上传数据，转化为 UI 事件，通过 bridge 转发到 UI thread,UI thread 进行 dom diff 算法，从而改变真实的 dom。

在线程协同工作时，UI main thread 和 JS thread 类似于 CS 模型，UI main thead 类似于客户端，JS thread 是服务端，由客户端（UI main thread）不断询问服务端（JS thread）并且请求数据，如果数据有变，则更新 UI 界面。

# 5. ReactNative 组件架构设计

在实际使用 RN 开发 APP 时，一个独立完整的组件可以如下设计：

① root 组件 1 个

- 负责初始化 state
- 负责提供对外的 props 列表
- 负责组合子 view 组件形成页面
- 负责注册业务逻辑对象提供业务逻辑方法
- 负责管理业务逻辑对象

② view 子组件多个

- 包含 props（共有属性）
- 根据 props 进行视图的渲染

③ 业务逻辑对象

- 包含 root 组件对象的引用 this.root
- 构筑器，初始化 root 对象和室友属性
- 提供业务逻辑方法

以上是一种面向对象的实现方式，当然也可以采取目前比较火的 RN 框架，实现更快的 APP 开发，比如 Flux,Reflux,Redux,Relay,Marty 等。

# 6. 总结与展望

ReactNative 给 app 的开发带来了新的希望与期待，虽然目前还做不到“一次编写到处编译”的真正的跨平台 app，但它的出现，它的比较前卫的思想，给开发者带来了新的曙光，降低了 app 的开发门槛，方便了前端开发者和 app 开发者进行 app 的开发。它有它的优点，相比于当前的 HybirdApp 和 Webapp，它摆脱了 webview（不需要再考虑卡顿的交互和性能问题）；有较强的扩展性（js 直接调用 native 端提供的基本控件）；可以直接使用原生动画效果等等令人兴奋的优点，但也有它的缺点，在 guthub 上也能看到很多 react-native 源码库里开发者提出的 bug 未修复，它有它的局限性，相比如 native，它的动画性能还是比不上原生，所以，目前的 RN 可能更适合做偏 UI 方面相对简单的一些需求和功能，但是我们有理由相信，RN 会引领，或者正在引领新的 APP 开发方式，也期待 RN 会越来越完善，真正实现它的初衷—既保留良好的用户体验，又加快 app 的开发效率。

# 参考文献

[1].王建波. 论移动应用编程的新方向[J]. 科技展望, 2016, 26(33).

[2].潘婷婷. React Native 在 APP 开发中的应用研究[J]. 无线互联科技, 2016(19):142-143.

[3].王沛. 虚拟 DOM Diff 算法解析.
http://www.infoq.com/cn/articles/react-dom-diff.

[4].阮一峰. React 入门实例教程.
http://www.ruanyifeng.com/blog/2015/03/ react.html,2015-03-31

[5].shane_xu. react-native 操作 dom.
https://segmentfault.com/a/1190000003975609,2015-11-10