

移动跨平台开发

REACT NATIVE

为什么用 REACT NATIVE

- 跨平台开发
 - 一次编写, 处处运行
 - 降低开发成本
- 开发效率高
 - 画界面很灵活
 - 调试方便
- 渲染速度快
 - Virtual DOM
 - 原生渲染
- 提升 Android/iOS 开发技能

演示

跨平台四大流派

- Web 流
- 代码转换流
- 渲染速度快
- 虚拟机流

by 百度前端团队

WEB 流

- Hybrid
- 内嵌WebView
- PhoneGap/Cordova

性能慢?

代码转换流

- 将某种语言转成 Objective-C、Java 或 C#，然后使用不同平台下的官方工具来开发
- Java 转 Objective-C
- Objective-C 转 Java
- Java 转 C#
- ...



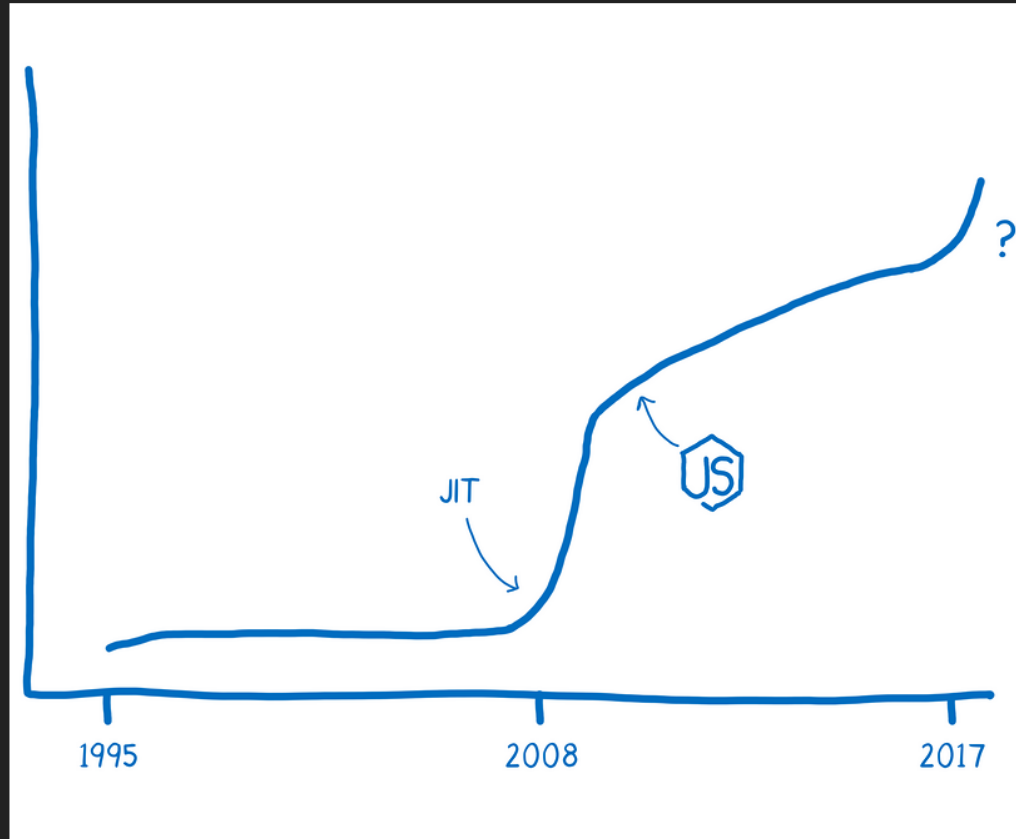
编译流

- 将某个语言编译为二进制文件，生成动态库或打包成 apk/ipa/xap 文件
- C/C++
 - 使用C++实现非界面部分, 如  [Mars](#)
 - 使用 OpenGL 来绘制界面, 常见于游戏中
 - 基于C++的跨平台UI库, 如QT
- Xamarin
 - C# 开发 Android/iOS
- RoboVM
 - Java 转为 iOS

虚拟机流

- Java VM
 - J2ME
- Unity3D/Cocos2d-js
- Adobe AIR
- NativeScript
-  React Native
-  Weex

JavaScript性能



引入JIT即时编译后, 性能大幅提升

解决方案

- 非界面/基础模块用C++
- 界面用JSVM系: React Native / Weex

命令式编程

=> HOW

声明式编程

=> WHAT

命令式

```
LinearLayout ll = new LinearLayout(this);
TextView tv = new TextView(this);
tv.setText("hello world");
ll.addView(tv, new LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.WRAP_CONTENT,
    LinearLayout.LayoutParams.WRAP_CONTENT));
setContentView(ll);
```

声明式

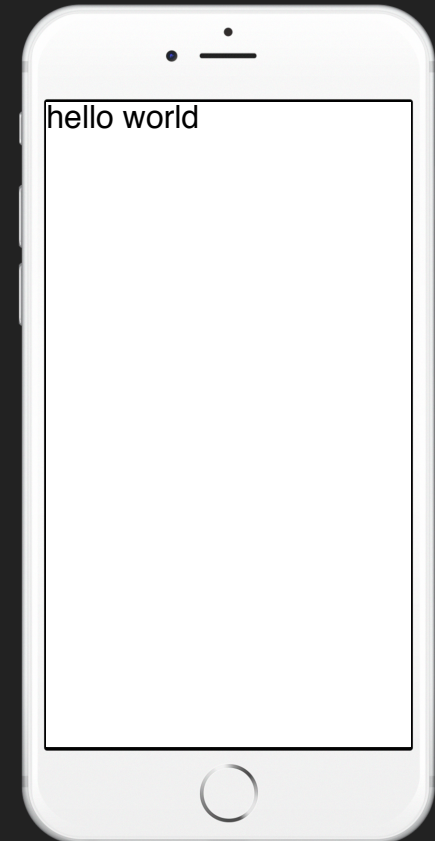
```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    <TextView>
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="hello world!" />
</LinearLayout>
```

Hello World

```
1 import React, { Component } from 'react';
2 import { AppRegistry, Text } from 'react-native';
3
4 class HelloWorldApp extends Component {
5   render() {
6     return (
7       <Text>
8         hello world
9       </Text>
10     );
11   }
12 }
13
14 AppRegistry.registerComponent('HelloWorldApp', () => HelloWorldApp);
```

No Errors

Show Details

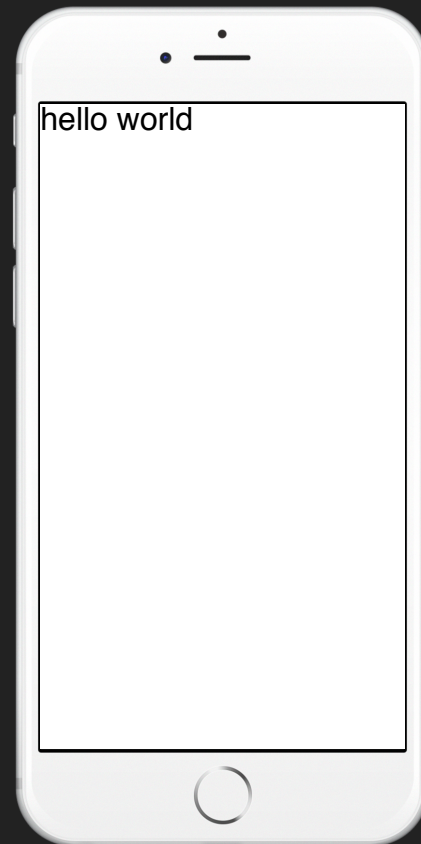


声明式 Demo

```
1 import React, { Component } from 'react';
2 import { AppRegistry, Text, TouchableOpacity } from 'react-native';
3 class HelloWorldApp extends Component {
4   state = {
5     description: 'hello world',
6   }
7   render() {
8     return (
9       <TouchableOpacity onPress={this.press}>
10         <Text>
11           {this.state.description}
12         </Text>
13       </TouchableOpacity>
14     );
15   }
16   press = () => {
17     this.setState({
18       description: '你好, 世界',
19     })
20   }
21 }
```

No Errors

Show Details



界面随状态的变化而变化 => Android DataBinding

遇到各种坑...

- 动画卡顿
- 反应迟钝
- 数据如何存储
- 事件无法传递
- ...

- 1) 只要必要时重新渲染
- 2) 直接设置控件样式
- 3) 引入Redux...

JS与Native交互原理简介

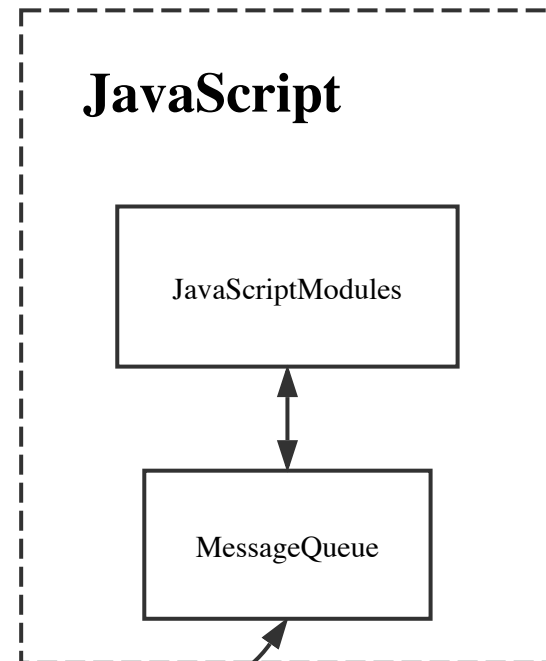
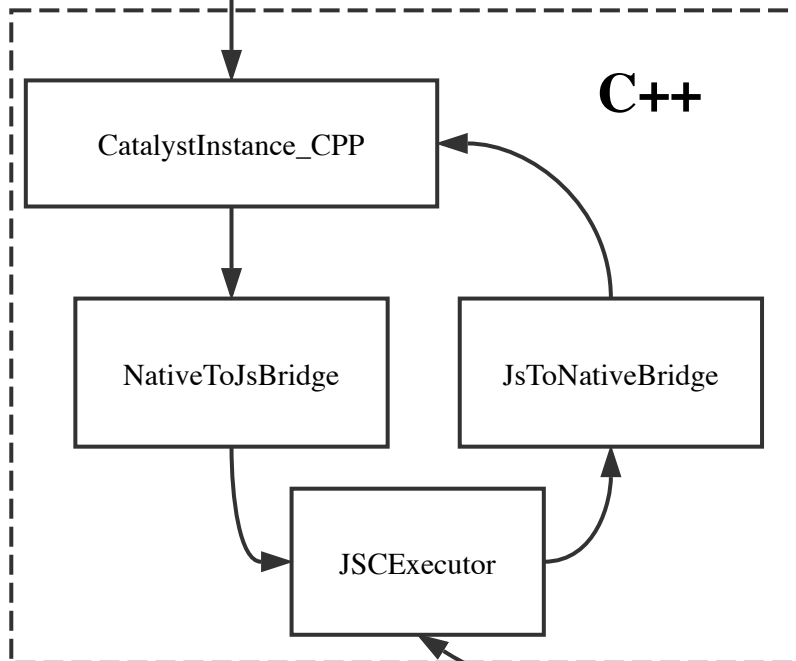
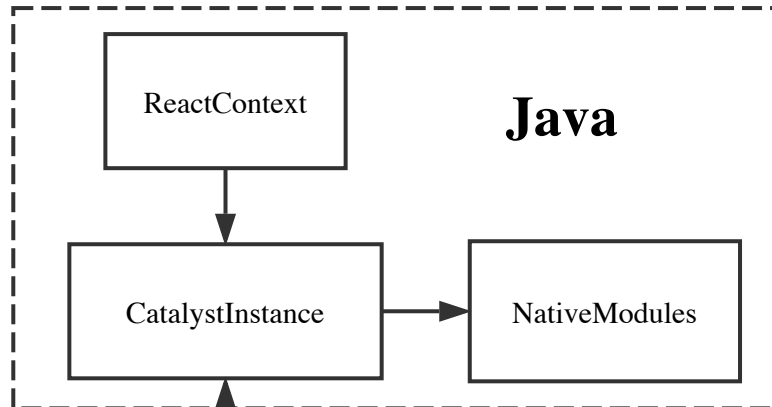
Android WebView

Java调JS

```
webview.loadUrl("javascript: alert('haha')");
```

JS调Java

```
webview.addJavascriptInterface(new Object() {  
    @JavascriptInterface  
    void foo() {  
        Log.e("Demo", "foo");  
    }  
}, "bridge");  
  
webview.loadUrl("javascript: window.bridge.foo()");  
// 链接跳转: shouldOverrideUrlLoading()
```



热更新

果爹发飙

今天收到Apple的警告邮件。

应用中使用了JSPatch一段时间了，之前的版本是没有问题的。

而且这个通知邮件也不是在提交更新版本审核过程中收到，而是苹果主动发出的。

- 是否和JSPatch有关?
- 如果有关请问是否有解决方案。
- 其它开发者最近有收到类似邮件的话，也请分享一下。请不要灌水。
- 相关信息 / Related information
 - [可能用到patch相关的第三方SDK——处理进度汇总](#)
 - [Issue in Apple Developer Forums](#)
 - [Issue in react-native](#)
 - [Issue in AFNetworking](#)
 - [Issue in alibaba/weex](#)
 - [issue in alibaba/LuaViewSDK](#)
 - [discussion in Hack News thanks @yong-chen04](#)
 - [文章: Apple向热更新下达最后通牒](#)

- 热更新很方便
- RN最大的优势不是热更新, 而是其跨平台技术
- 没有热更新, 照样可以上RN

结论

- 技术栈要与时俱进
- 不仅要有深度, 还要有广度

结束