React Native 业务实践和性能优化

携程旅行网

赵辛贵

目录

目录

ReactNative - 简单介绍

- Facebook基于React开发
- 首次引入JavaScript to Native开发模式
- 2015/3/27 iOS开源，2015/9/15 Android开源
- 每2周迭代一个版本，最新稳定版V0.35

ReactNative - 谁在使用？

电商/O2O

- 手机京东
- 唯品会
- 乐视商城
- 百度糯米
- 波罗蜜全球购
- 全民一元夺宝
- 返利
- 千牛
- 京东到家
- 58同城
- 美甲帮

旅游

- Airbnb
- 携程旅行
- 去哪儿旅行
- 途牛旅游
- 艺龙旅行
- 去哪儿火车票
- 游谱旅行
- 就旅行
- 探途离线地图
- 快巴出行

金融

- 一账通
- 蚂蚁聚宝
- 速投盈
- 借贷宝
- 千牛
- 小方
- 快贷贷款
- 挖财记账理财

社交/工具

- QQ空间
- 新浪微博
- 手机百度
- 虾米音乐
- 网易有道词典
- 丁香医生
- 米家
- ee聊天交友
- 堆糖
- 洋葱数学
- 龙珠直播

基于2016年10月18日百度手机助手站点，Android App数据

ReactNative优势

Size优势

iOS App Size大于100MB，在移动网络下下载，会受到限制

支持动态发布

支持线上业务功快速迭代和随时更新发布

提升用户体验

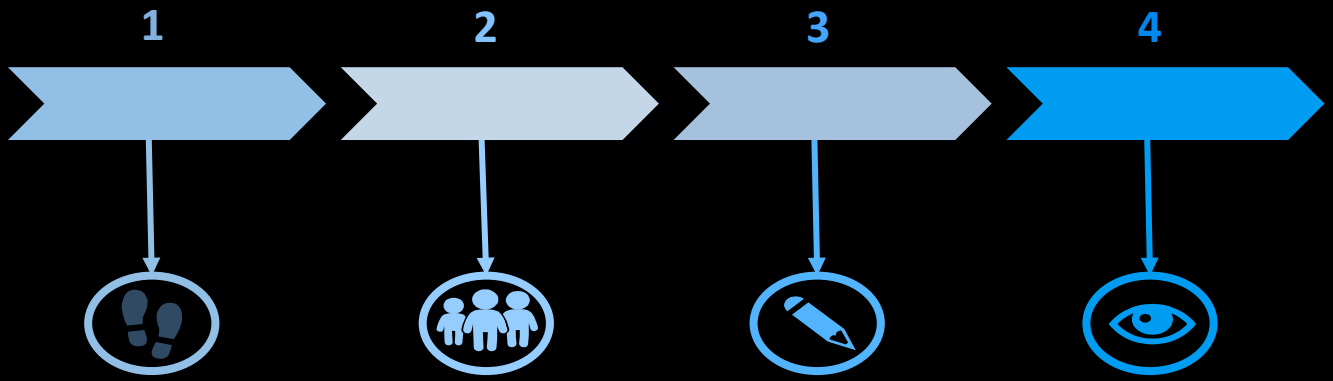
用户体验接近Native，远优于H5 Hybrid

相对成熟

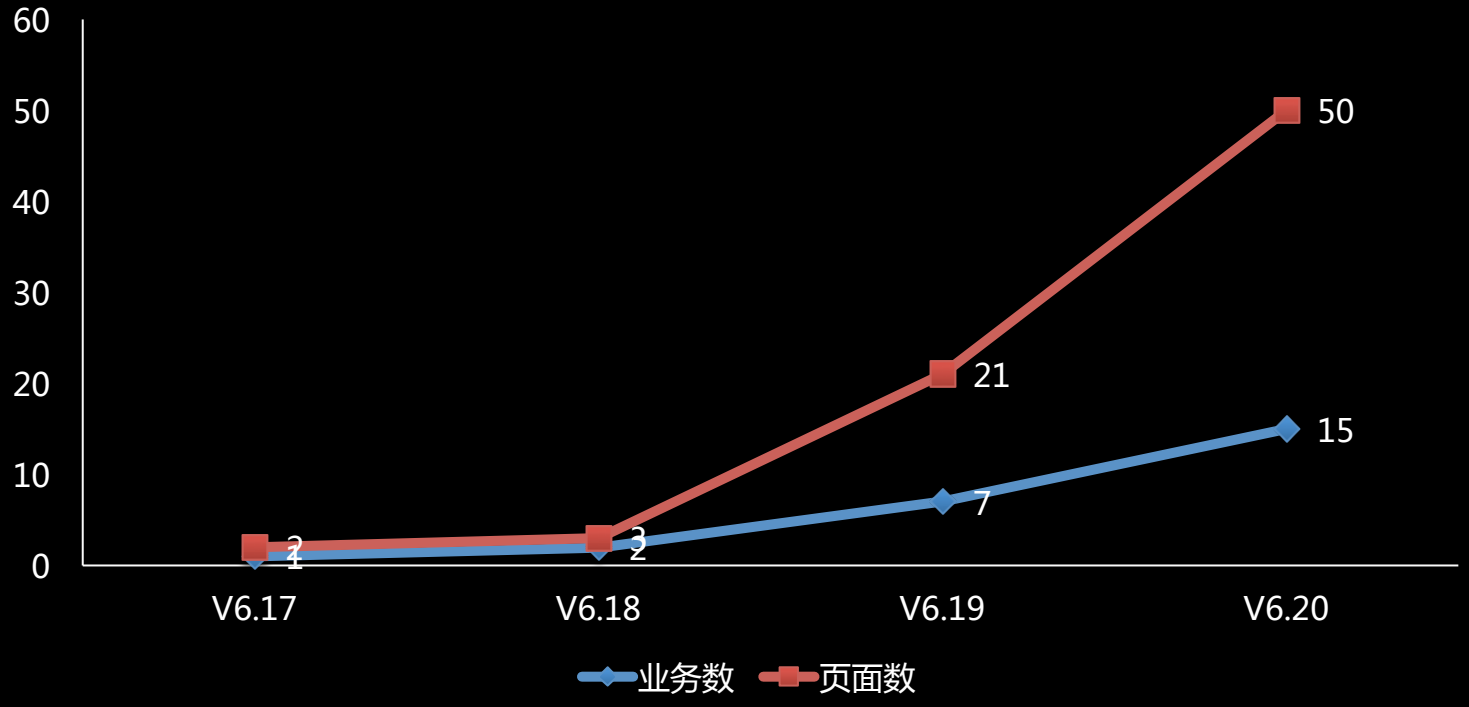
经过一年多的发展探索，RN接口和runtime趋于稳定，且支持跨平台，降低开发成本

目录

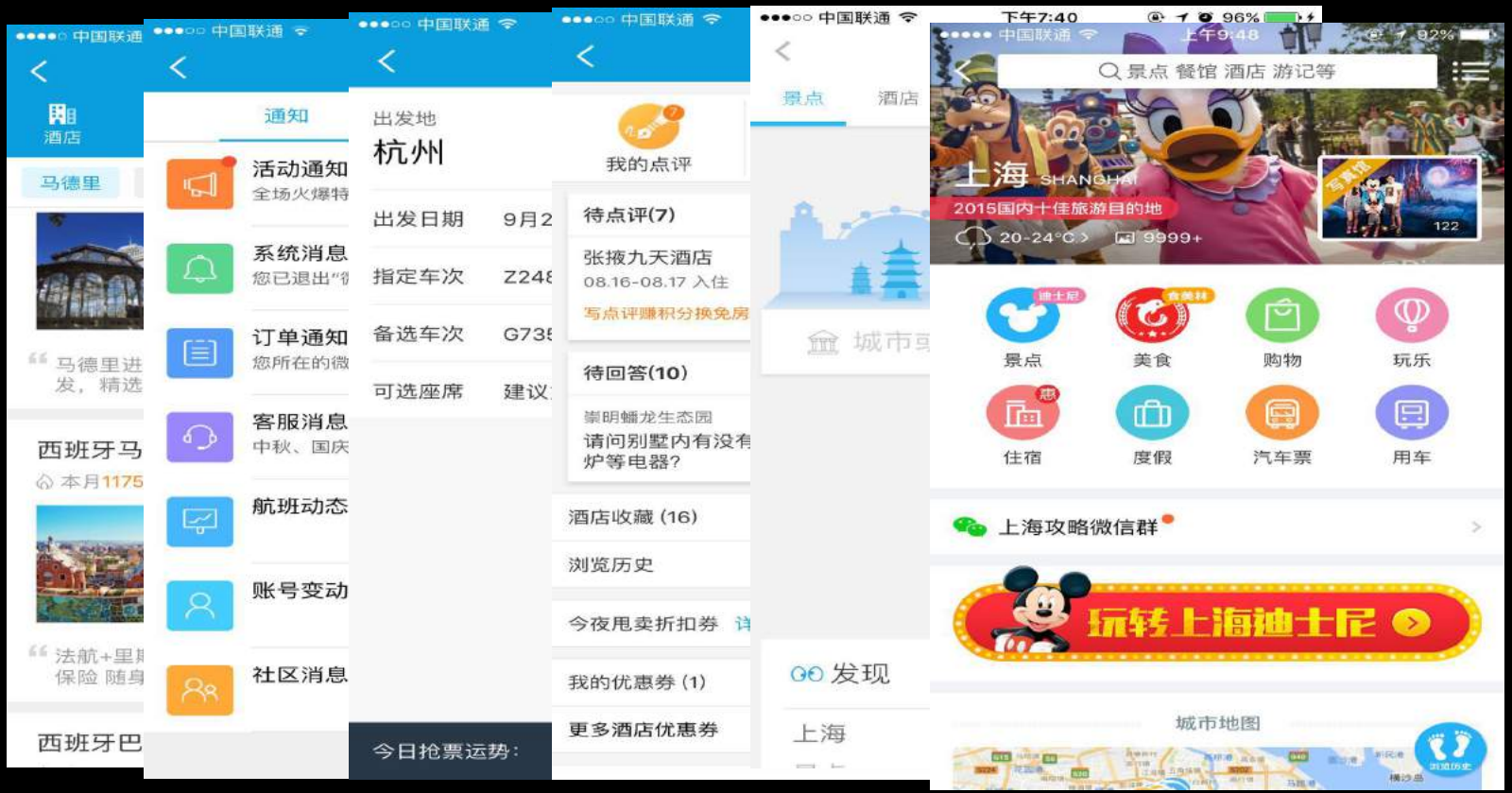
RN在携程的发展


1
2
3
4
体验阶段
2015年5月，基于RN 0.4版本，在iOS App V6.5版本开发发票抬头
探索阶段
2015年10月，各业务团队开始研究探索RN在业务上的使用
试用/准备
2016年4月，基于RN 0.28 定制CRN，App V6.17-V6.18，站内信、机票低价订阅
规模化使用
升级RN 0.30，App V6.19，15个业务，50+页面。

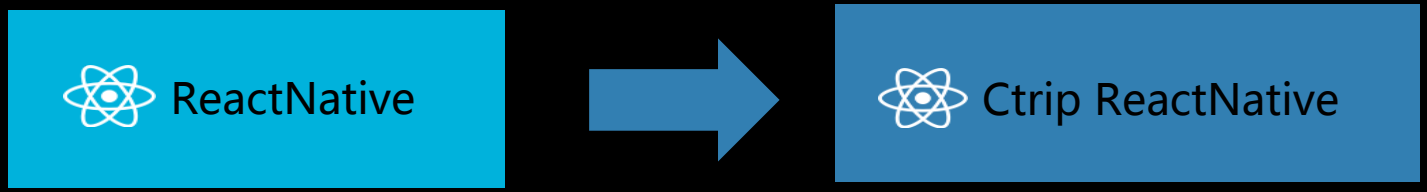
RN在携程App中的使用情况



RN在携程App中的使用场景



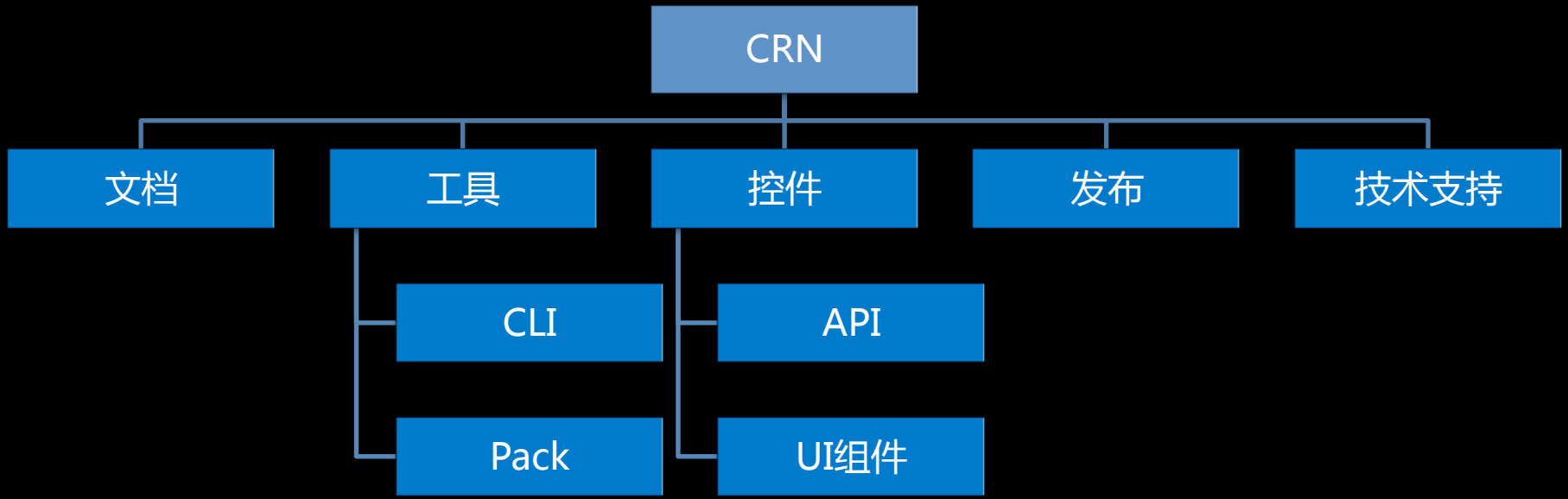
携程App如何引入RN



关注点：

1. 方便业务接入；
2. 功能增强；
3. 性能/稳定性优化；
4. 跨平台<iOS/Android>使用;

CRN组成



CRN扩展组件

RN原生

- ActivityIndicator
- Image
- ListView
- ListView.DataSource
- MapView
- Modal
- Navigator
- RefreshControl
- ScrollView
- Slider
- StatusBar
- Switch
- Text
- TextInput
- TouchableXXXX
- View

CRN新增

- PulldownRefreshView
- LoadMoreRefreshView
- HeaderView
- NavigationBar
- Button
- NavigationBarButton
- HtmlText
- SpriteImage
- SwipeoutView
- LoadingView
- DatePicker
- ProgressView
- SegmentedControl
- CRNListView
- CRNListViewDataSource

CRN扩展API

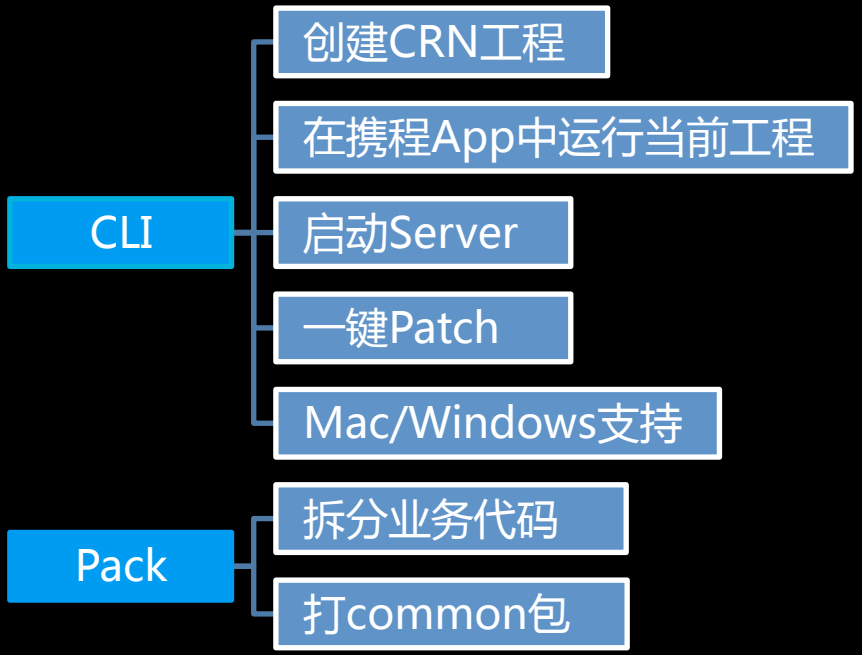
RN原生

- Alert
- Animated
- AppRegistry
- AppState
- CameraRoll
- Clipboard
- Dimensions
- InteractionManager
- LayoutAnimation
- Linking
- NativeMethodsMixin
- NetInfo
- PanResponder
- PixelRatio
- StyleSheet
- Vibration

CRN新增

- Application
- **Storage**
- Device
- File
- Encrypt
- Env
- Notification
- PhotoBrowser
- AddressBook
- Calendar
- Location
- **Fetch**
- Toast
- Log
- Call
- ABTest
- QRCode
- ImagePicker
- BirthdayPicker
- User
- Pay
- Share

CRN组成工具


CLI
创建CRN工程
在携程App中运行当前工程
启动Server
一键Patch
Mac/Windows支持
Pack
拆分业务代码
打common包

目录

RN性能和稳定性优化

- 打包Bundle拆分
- 页面加载提速
- Crash优化
- List View性能优化
- 版本管理和发布

RN性能和稳定性优化

- 打包Bundle拆分
- 页面加载提速
- 版本管理和发布
- Crash优化
- List View性能优化

背景介绍 - RN代码

```
'use strict';

/* 1.模块引用申明部分 */
import React, { Component } from 'react';
import { AppRegistry } from 'react-native';
import { HelloWorldView } from './HelloWorldView'


/* 2.入口模块定义部分 */
class Index extends Component {
  function render() {
    return (<HelloWorldView/>)
  }
}


/* 3.入口模块注册部分*/
AppRegistry.registerComponent('RNMessage', () => Index);
```

背景介绍 - RN打包的Bundle

```
/* 1. 头部--全局定义部分 */
(function(global) {
global.__DEV__=true;
global.__BUNDLE_START_TIME__=Date.now();
})(typeof global !== 'undefined' ? global : typeof self !== 'undefined' ? self : this);


/* 2. 中间--各模块定义部分 */
__d(0 /* RNMessage/index.android.js */, function(global, require, module, exports) {
  /*...code...*/
  module.exports=require(12 /* ./src/index */);
}, "RNMessage/index.android.js");
__d(188 /* InitializeJavaScriptAppEngine */ , function(global, require, module, exports) {
  /*...code...*/
  require(80 /* RCTDeviceEventEmitter */ );
}, "InitializeJavaScriptAppEngine");
__d(473 /* BorderBox */ , function(global, require, module, exports) {
  /*...code...*/
  module.exports = BorderBox;
}, "BorderBox");
__d(474 /* resolveBoxStyle */ , function(global, require, module, exports) {
  /*...code...*/
  module.exports = resolveBoxStyle;
}, "resolveBoxStyle");


/* 3. 尾部--引擎初始化+执行入口模块 */
;require(188);//InitializeJavaScriptAppEngine
;require(0);//入口模块
```

背景介绍 - 几个概念介绍

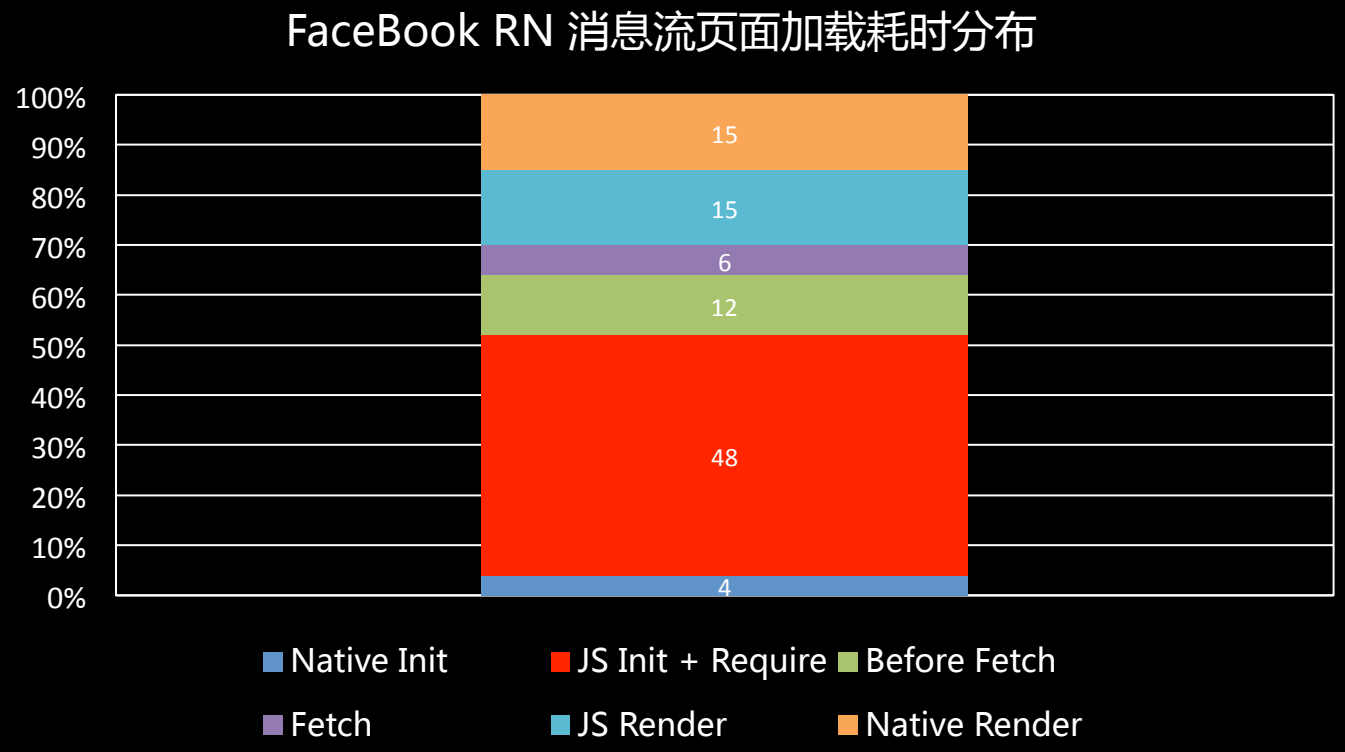
1. CommonJS规范；

- 每个文件都是一个module，命名空间独立；
- Module通过export导出对外暴露的接口；

2. Define和Require

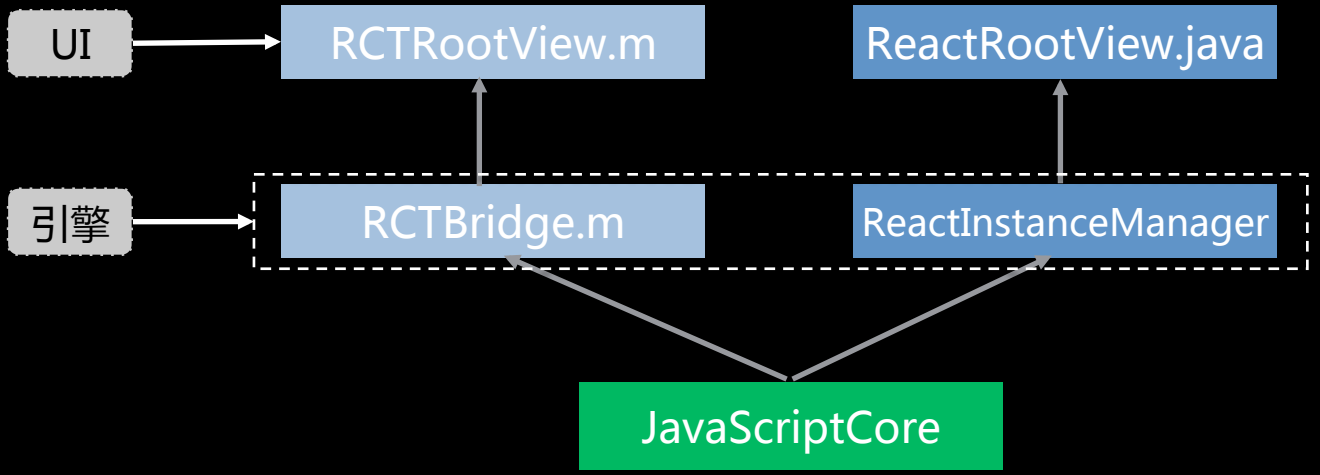
- Define给模块定义一个标识符，RN里面用自增长的数字表示；
- Require根据标识符去获取define好的模块；
- Require时，同步执行模块js代码

背景介绍 - 性能瓶颈



Note:图表来源Dive into React Native performance

背景介绍 - 页面渲染核心类

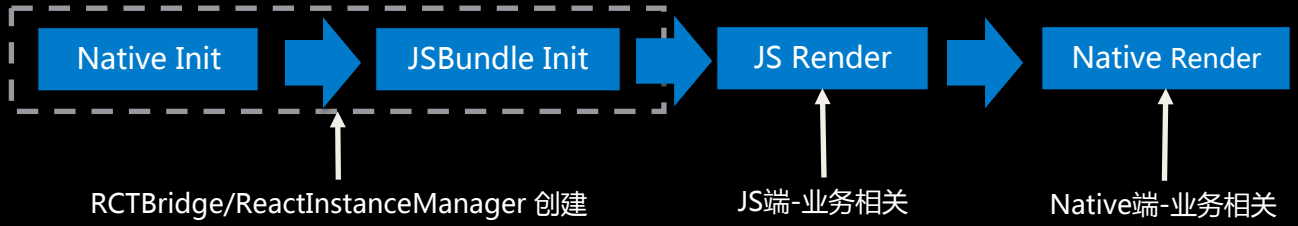


Bundle拆分 - 方案1

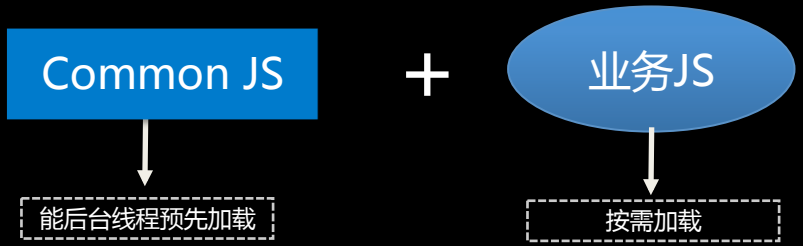
1. 实现方案
   - import react/react-native, 打框架包;
   - 标准命令，打业务包;
   - 从业务包中删除框架包中已经有的模块代码；
2. 优/劣
   - 无侵入性、分析文件格式拆分即可；
   - 实现成本低;
   - bundle作为整体加载，速度无法提升;

Bundle拆分 - 方案2

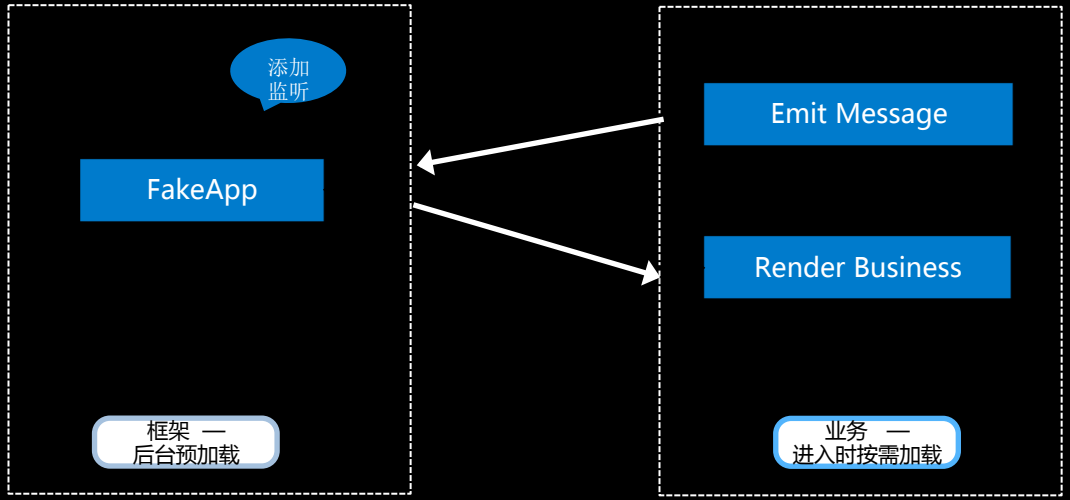
1. 加载流程回顾



2. 新的解决方案



Bundle拆分 - 方案2加载



1. 能后台加载的是一个完整的RN App，需设计一个空白页面的FakeApp；
2. 所有RN App的入口都是FakeApp；
3. 需要记录业务App的模块入口；

Bundle拆分 - 方案2的FakeApp设计

```
import React, { Component } from 'react';
import {
  AppRegistry,
  View,
  DeviceEventEmitter
} from 'react-native';

/* 1. 添加全局监听 */
var mainComponent = null;

DeviceEventEmitter.removeAllListeners(); //fixed ios multi-trigger
DeviceEventEmitter.addListener("RenderModuleEventListener", function(event) {
  console.log("Render moduleId: " + event.moduleId);
  if (event && event.moduleId) {
    mainComponent = require(event.moduleId);
  }
});

/* 2. 定义FakeApp模块 */
var FakeApp = React.createClass({
  render: function() {
    var _content = null;
    if (mainComponent) {
      _content = React.createElement(mainComponent, this.props);
    }
    return _content || <View/>;
  },
});

 /* 3. 注册FakeApp模块 */
AppRegistry.registerComponent('FakeApp', () => FakeApp);
```

Bundle拆分 - 方案2Tips

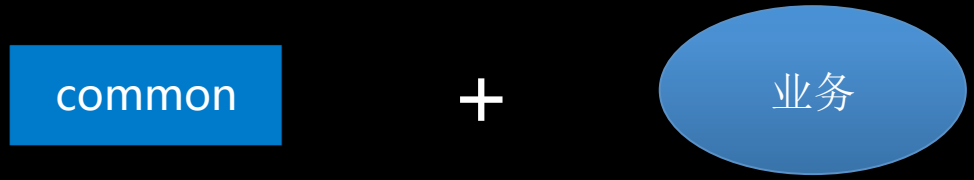
打包Tips

- Android和iOS分开打common包<FakeApp>，业务包使用iOS的；
- mapping文件记录common包的moduleID和module，供多个业务使用；
- 业务代码中原有的入口模块需要export；
- 记录业务代码入口的moduleID；

加载Tips

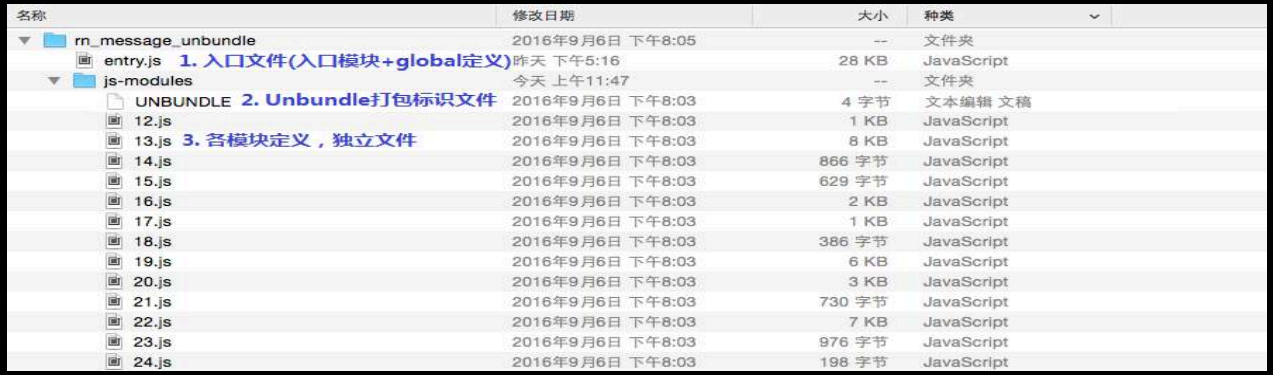
- 资源加载调整，Android 遵循iOS资源规则；

Bundle拆分 - 方案2瓶颈



- 业务代码一次性载入执行；
- 随着业务代码量的增加，JavaScript的执行瓶颈复现；

背景知识 - Unbundle&Prepack



名称	修改日期	大小	种类
rn_message_unbundle	2016年9月6日 下午8:05	--	文件夹
entry.js 1. 入口文件(入口模块+global定义)	昨天 下午5:16	28 KB	JavaScript
js-modules	今天 上午11:47	--	文件夹
UNBUNDLE 2. Unbundle打包标识文件	2016年9月6日 下午8:03	4 字节	文本编辑 文稿
12.js	2016年9月6日 下午8:03	1 KB	JavaScript
13.js 3. 各模块定义，独立文件	2016年9月6日 下午8:03	8 KB	JavaScript
14.js	2016年9月6日 下午8:03	866 字节	JavaScript
15.js	2016年9月6日 下午8:03	629 字节	JavaScript
16.js	2016年9月6日 下午8:03	2 KB	JavaScript
17.js	2016年9月6日 下午8:03	1 KB	JavaScript
18.js	2016年9月6日 下午8:03	386 字节	JavaScript
19.js	2016年9月6日 下午8:03	6 KB	JavaScript
20.js	2016年9月6日 下午8:03	3 KB	JavaScript
21.js	2016年9月6日 下午8:03	730 字节	JavaScript
22.js	2016年9月6日 下午8:03	7 KB	JavaScript
23.js	2016年9月6日 下午8:03	976 字节	JavaScript
24.js	2016年9月6日 下午8:03	198 字节	JavaScript

1. Unbundle
   - android支持，每个模块一个独立文件；
   - nativeRequire 固定路径查找需要的模块；
2. Prepack
   - iOS支持，一个二进制格式文件，以0xFB0BD1E5开头；
   - nativeRequire 从该二进制文件查找需要的模块；

CRN Unbundle方案

1. CRN打包文件格式

2. CRN Config文件格式

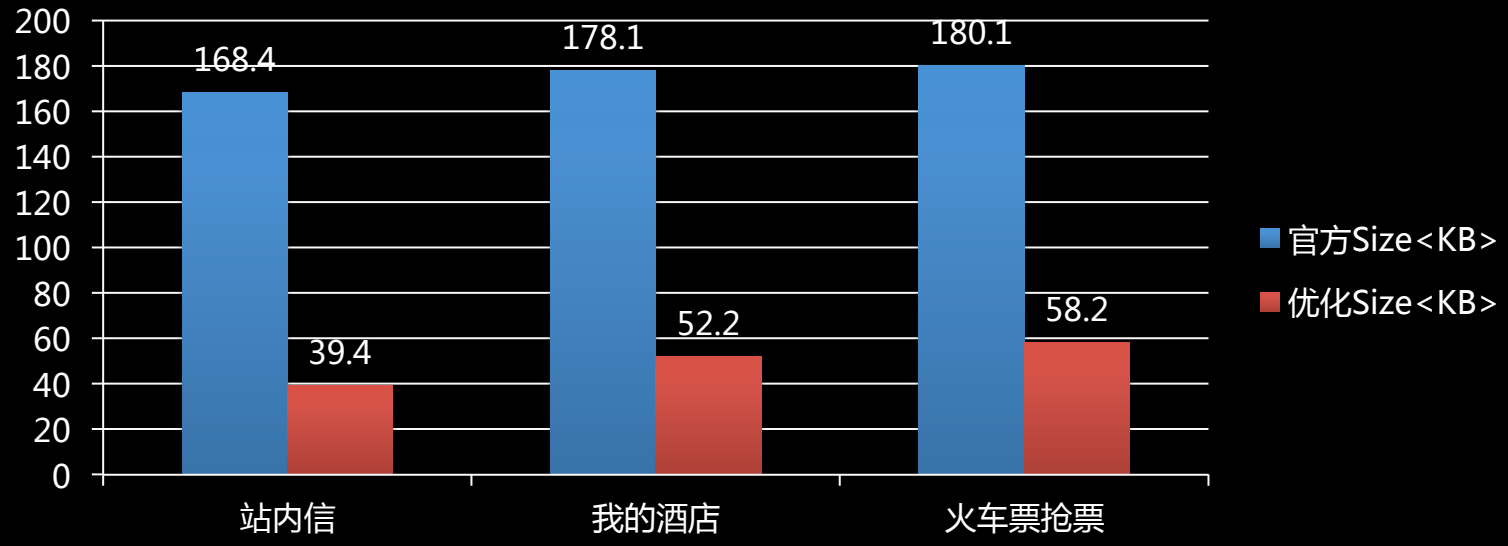
_crn_config

```
1  main_module=666666
2  module_path=js-modules
3  666666=0.js
4  666667=0.js
5  666668=1.js
6  666669=1.js
7  666670=2.js
8  666671=2.js
9  666672=3.js
10 666673=3.js
11 666674=4.js
12 666675=4.js
13 666676=5.js
14 666677=5.js
15 666678=6.js
16 666679=6.js
```

3. NativeRequire的实现

从磁盘读取JS module文件，然后执行

Bundle拆分效果

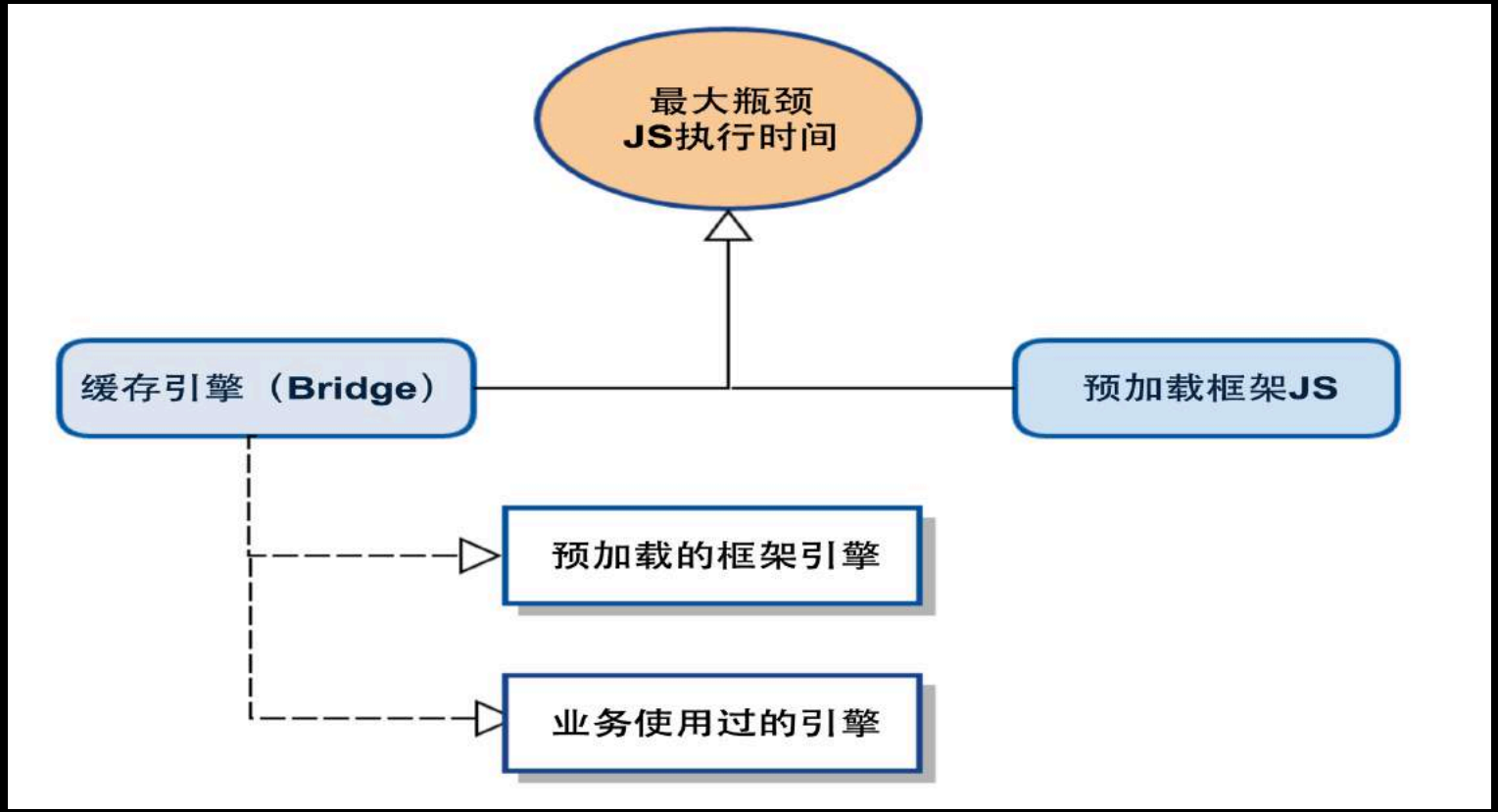


Note:使用7z压缩release包的size

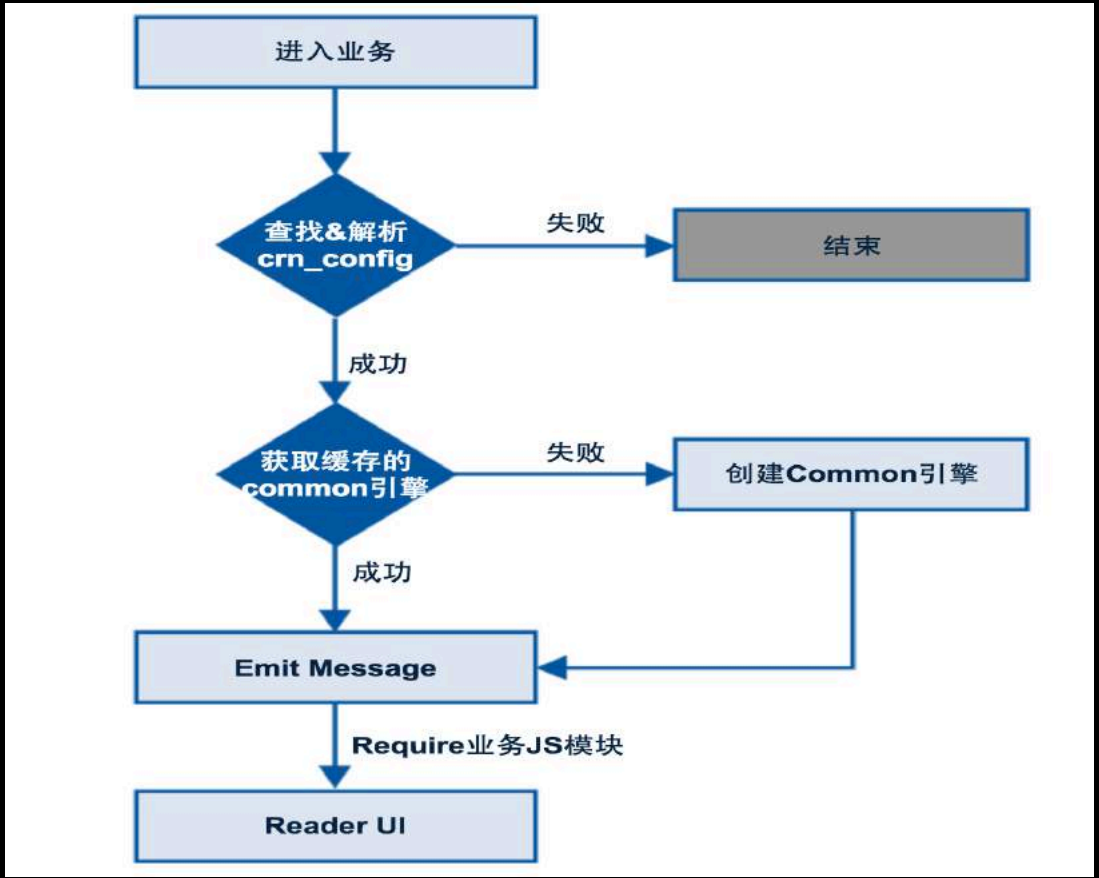
RN性能和稳定性优化

- Bundle拆分
- 页面加载提速
- 版本管理和发布
- Crash优化
- List View性能优化

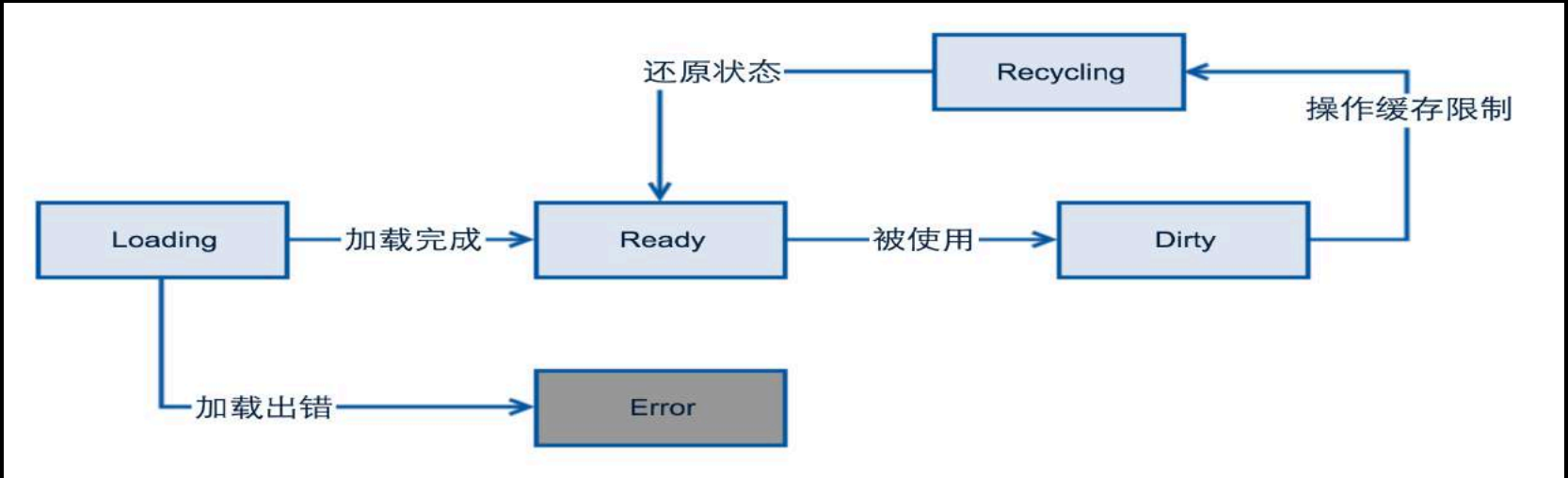
页面加载提速



页面加载流程


进入业务
查找&解析
crn_config
失败
结束
成功
获取缓存的
common引擎
失败
创建Common引擎
成功
Emit Message
Require业务JS模块
Reader UI

引擎(bridge)生命周期

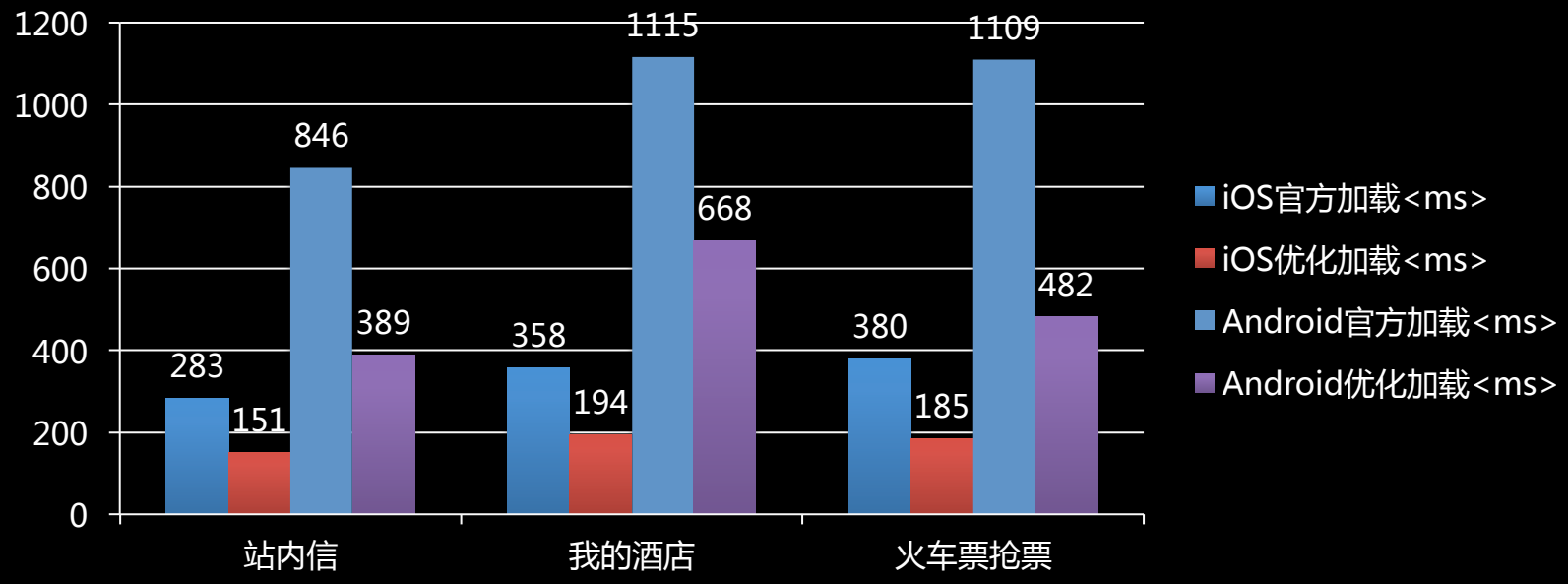


引擎(bridge)缓存策略

缓存策略

- 缓存池存储: Ready状态的引擎2个；
- Dirty数量>2时候，没有被使用的dirty引擎会被回收；
- Dirty通过unRequire业务代码，Restore到Ready状态；
- 页面生命周期新增RN容器显示、销毁状态；

页面加载时间



Note：单机多次验证，iOS由iPhone6设备测试 / Android由Sony Xperia测试

线上数据

1. 页面加载耗时

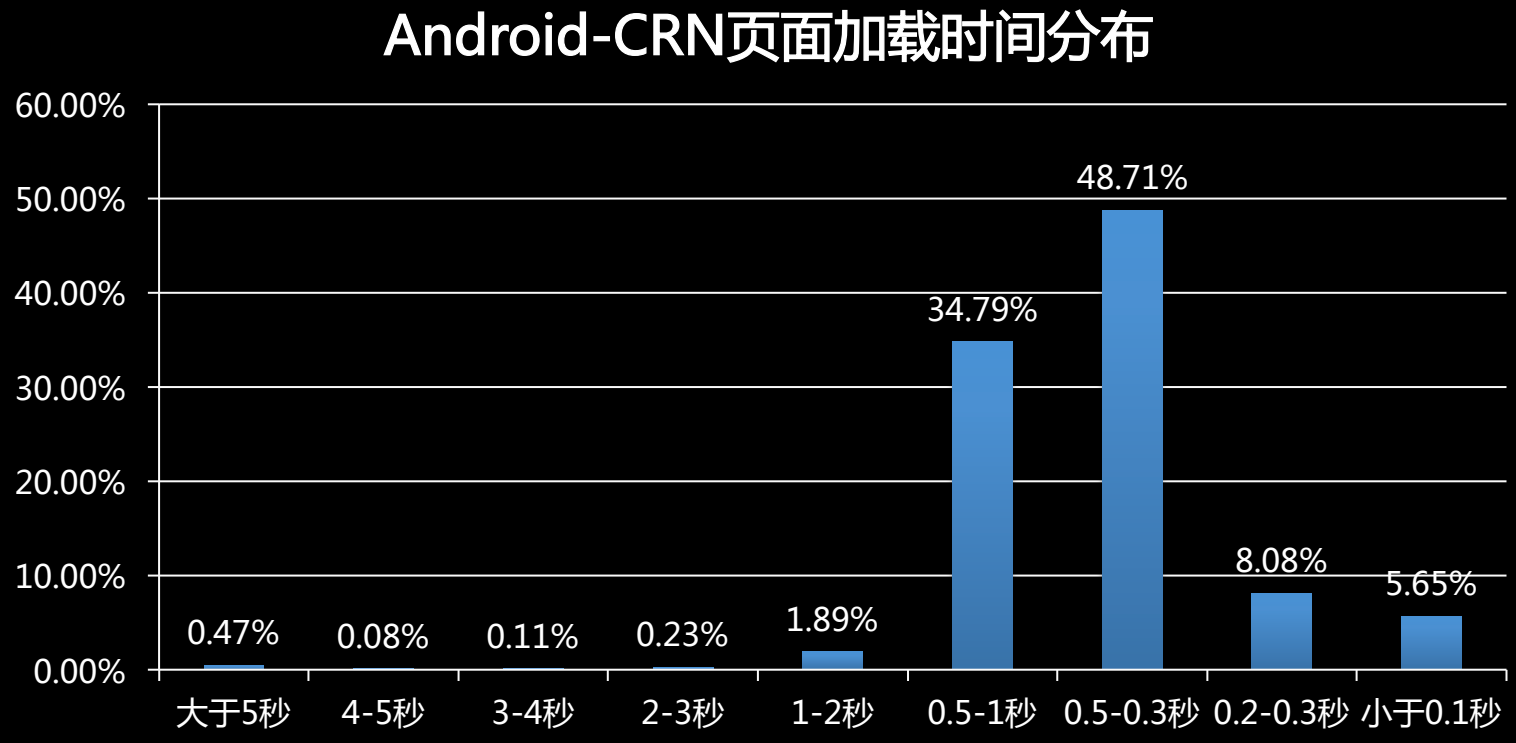
平台	RN(秒)	Hybrid(秒)
iOS	0.225	1.01
Android	0.494	1.54

2. 页面加载成功率

平台	RN	Hybrid
iOS	98.43%	96.41%
Android	98.96%	96.40%

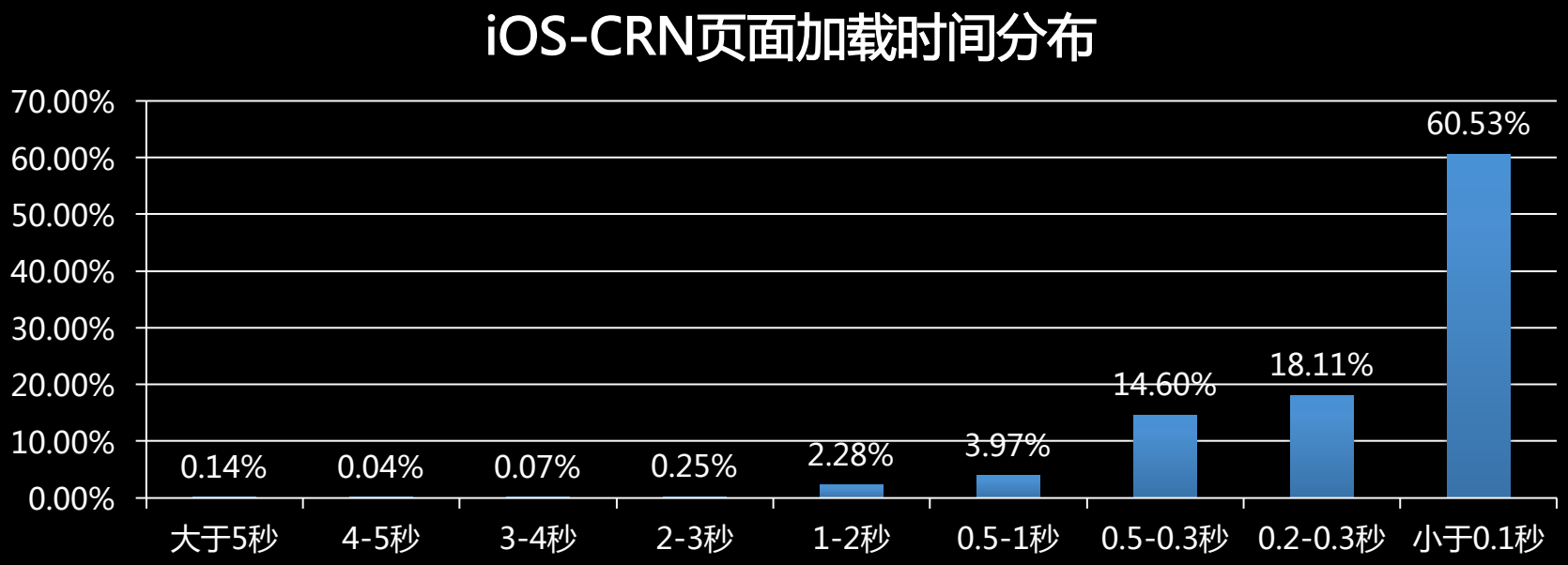
Note：基于2016-10-12线上生产数据统计

线上页面加载时间分布 - Android



Note：基于2016-10-12线上生产数据统计

线上页面加载时间分布 - iOS



Note：基于2016-10-12线上生产数据统计

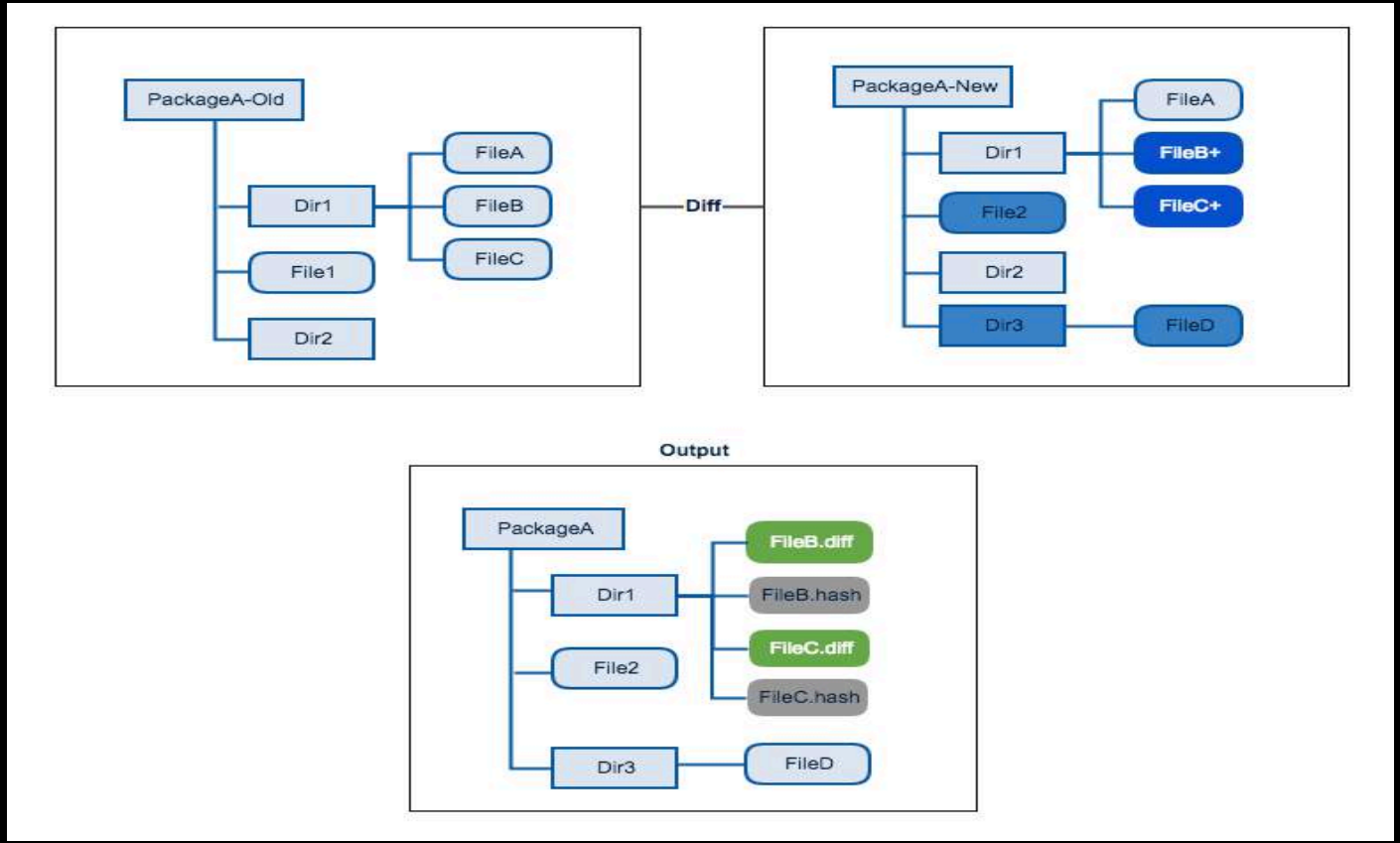
RN性能和稳定性优化

- Bundle拆分
- 页面加载提速
- 版本管理和发布
- Crash优化
- List View性能优化

发布

1. 离线包方式
   - 7z压缩 ；
   - 使用时解压安装；
   - 避免下载，减小对网络依赖；
2. 差分增量更新
   - 改进的bsdiff算法，效率极大提升；
3. 分版本发布
   - App版本，RN版本，CRN版本，业务代码版本；
   - 发布平台统一build上传发布；
   - 发布平台控制RN和App版本对应关系；

发布 - 差分算法



Note: 查分算法基于bsdiff 4.3版本

发布 - 线上数据

如果直接对2个zip包直接bsdiff，产生的patch文件size约为240KB

RN性能和稳定性优化

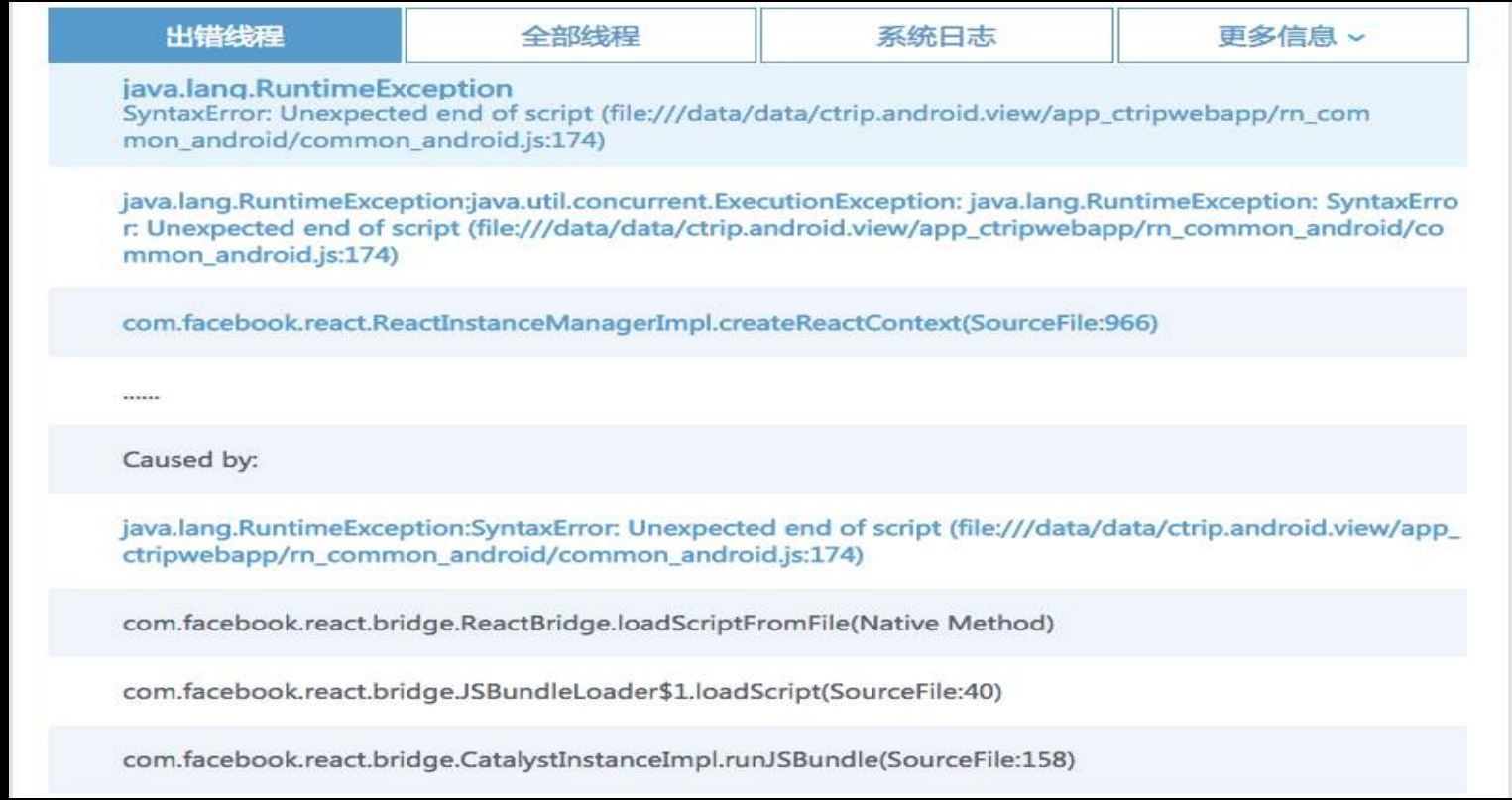
- Bundle拆分
- 页面加载提速
- 版本管理和发布
- Crash优化
- List View性能优化

Crash优化

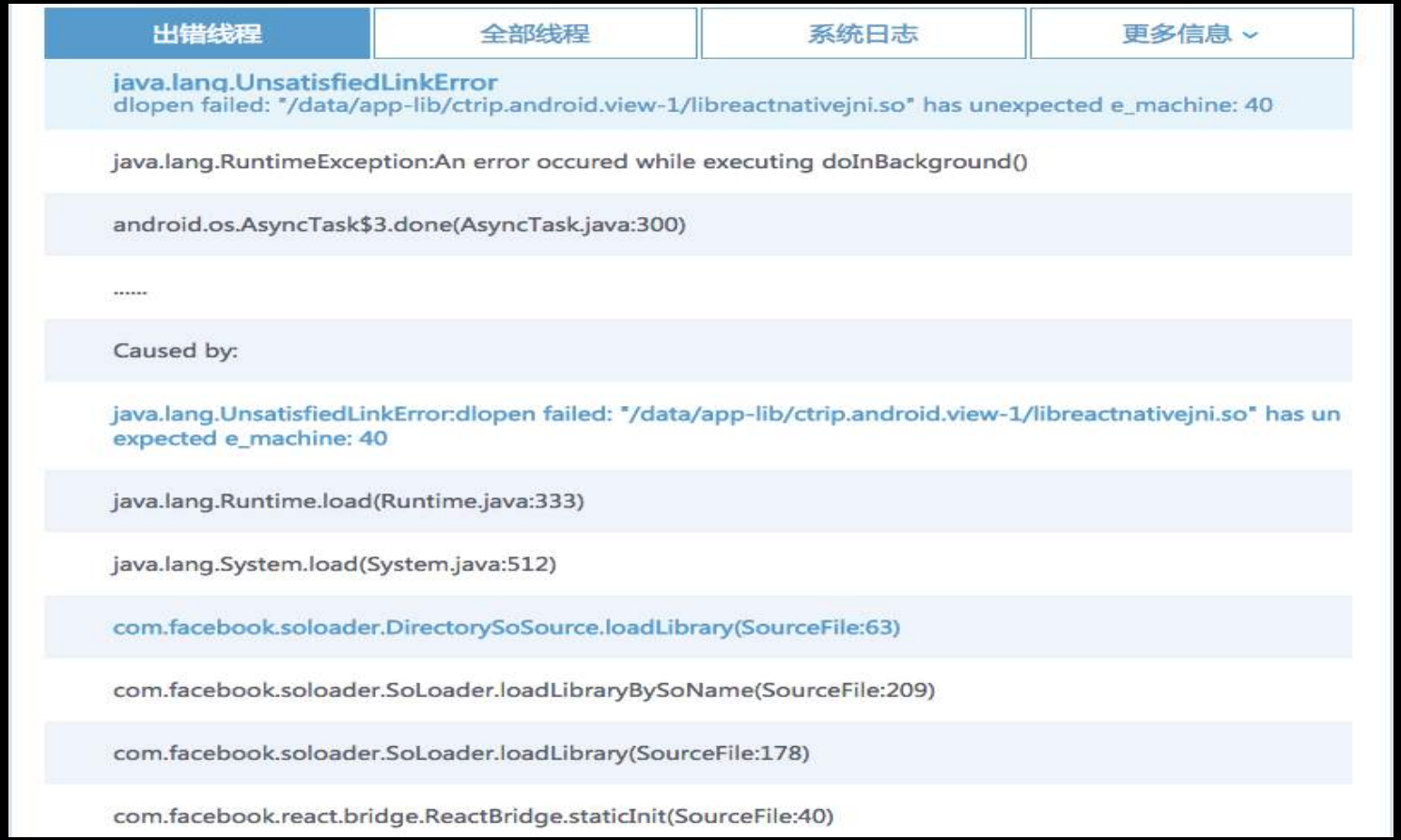
1. 整体状况
   - iOS稳定性、兼容性好于Android；
2. 出现的原因
   - RN框架层直接Throw Exception；
   - 代码稳健性不够；
3. 出现的位置
   - JSBundle加载过程中；
   - 业务运行过程中JS报错；
   - Native模块执行报错；
   - JNI层so加载报错；

Crash Sample - Android

Crash Sample - Android



Crash Sample - Android



Crash Sample - Android

出错线程 | 全部线程 | 系统日志 | 更多信息

SIGABRT

#00 pc 00022124 /system/lib/libc.so (tgkill +12) [armeabi-v7a]

#01 pc 000131c5 /system/lib/libc.so (pthread_kill +48) [armeabi-v7a]

#02 pc 000133d9 /system/lib/libc.so (raise +10) [armeabi-v7a]

#03 pc 0001209b /system/lib/libc.so [armeabi-v7a]

#04 pc 000219d8 /system/lib/libc.so (abort +04) [armeabi-v7a]

#05 pc 000086ff /data/app-lib/ctrip.android.view-1/libglog.so [armeabi-v5te]

#06 pc 00009af7 /data/app-lib/ctrip.android.view-1/libglog.so (google::LogMessage::Fail() +10) [armeabi-v5te]

#07 pc 0000bc8d /data/app-lib/ctrip.android.view-1/libglog.so (google::LogMessage::SendToLog() +204) [armeabi-v5te]

#08 pc 00009909 /data/app-lib/ctrip.android.view-1/libglog.so (google::LogMessage::Flush() +108) [armeabi-v5te]

Crash Sample - iOS

出错线程			
RCTFatalException: Unhandled JS Exception: ReferenceError: Can't find variable: global Unhandled JS Exception: ReferenceError: Can't find variable: global, stack: @1:0			
0	CoreFoundation	0x0000000182c09900	___exceptionPreprocess + 124
1	libobjc.A.dylib	0x0000000182277f80	objc_exception_throw + 56
2	CoreFoundation	0x0000000182c09848	-[NSException initWithCoder:]
3	CTRIP_WIRELESS	0x00000001004cd478	RCTFatal (RCTAssert.m:137)
4	CTRIP_WIRELESS	0x00000001004d91f8	-[RCTBatchedBridge stopLoadingWithError:] (RCTBatchedBridge.m:530)
5	libdispatch.dylib	0x000000018265d630	__dispatch_call_block_and_release + 24
6	libdispatch.dylib	0x000000018265d5f0	__dispatch_client_callout + 16
7	libdispatch.dylib	0x0000000182662cf8	_dispatch_main_queue_callback_4CF + 1844
8	CoreFoundation	0x0000000182bc0bb0	___CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__ + 12

Crash解决方案 - iOS

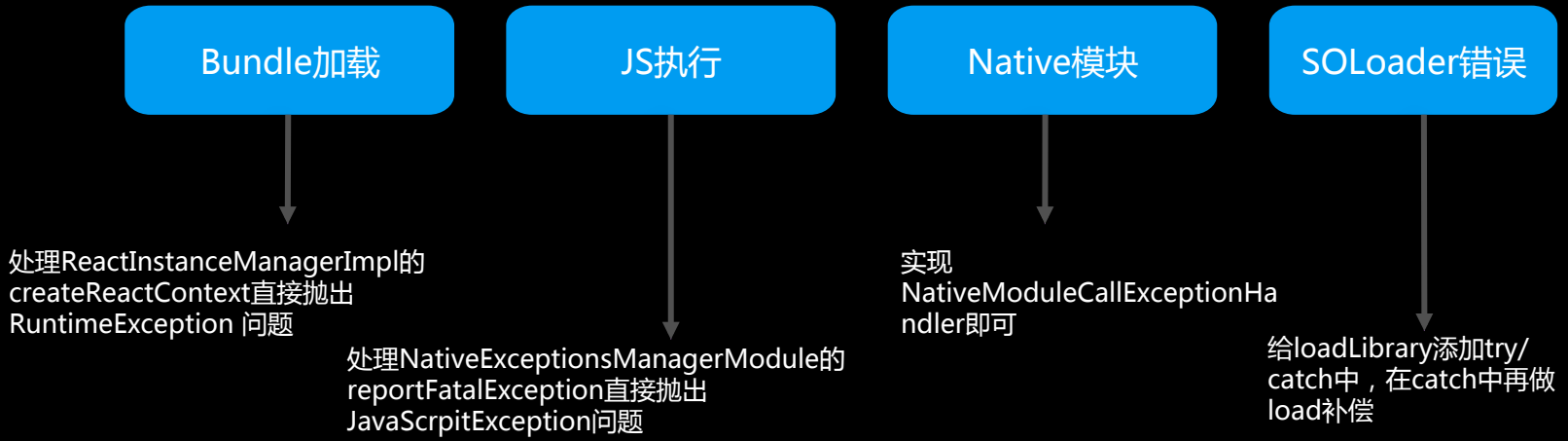
1. crash来源：void RCTFatal(NSError *error)

- 测试环境，表现为红屏；
- 生产环境，直接Raise Exception，导致crash；

2.解决: void RCTSetFatalHandler(RCTFatalHandler fatalhandler)

- 设置错误处理回调，自行处理错误；
- Bundle/JS加载执行过程中的错误信息，都会处理掉；

Crash解决方案 - Android



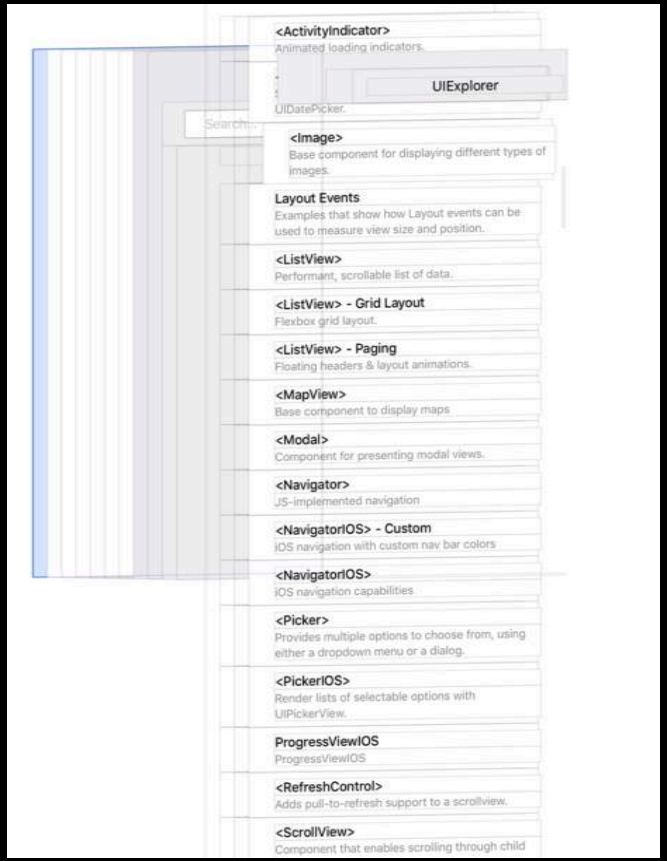
其它方面

Android的这些异常都是在RN的runtime层直接抛出异常，应当做修改，一层一层抛出给外层使用者，供UI层做处理/上报。

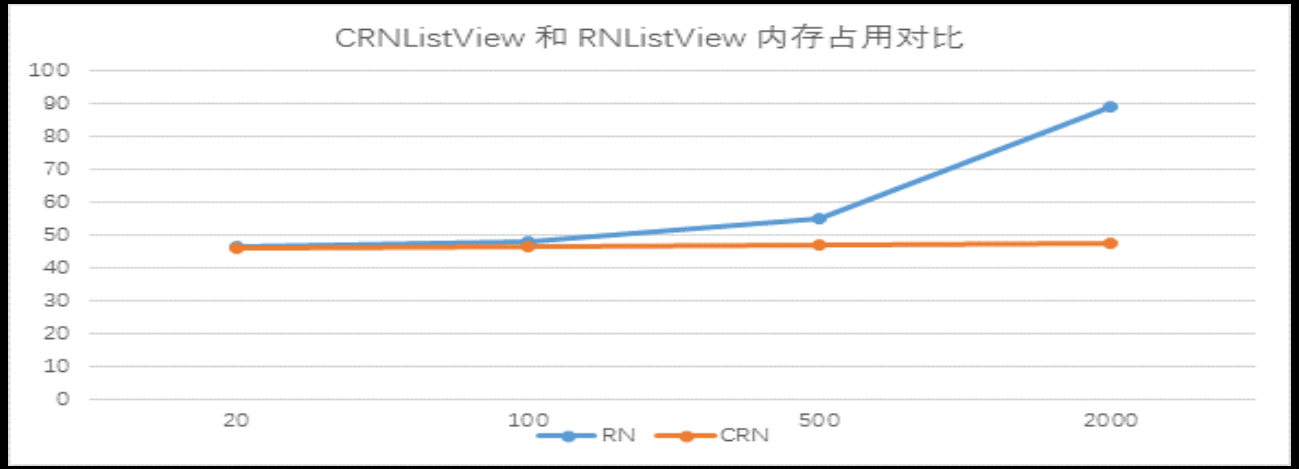
RN性能和稳定性优化

- 页面加载提速
- 版本管理和发布
- Crash优化
- List View性能优化

ListView性能问题



ListView优化



1. 基于Native UITableView/RecyclerView实现cell重用；

2. 功能扩展，集成了下拉刷新，载入更多，右侧索引等功能；

目录

总结规划

总结

- RN适用于大规模线上业务， 且RN代表未来一段时间内移动前端开发的趋势；
- 通过设计FakeApp拆分bundle+预加载+缓存能实现页面秒开；
- 错误捕获处理可以大大减少因为RN导致的crash；

未来规划

- 单引擎<bridge/ReactInstanceManager>执行所有业务；
- CRN-Web支持，让一套代码可以在三端运行，降低开发维护成本；

Thanks