携程大规模应用React Native的工程化实践

携程旅行网
赵辛贵

React从入门到精通

掌握当下最热门的前端利器

王沛
eBay 资深技术专家

你将获得

1. 全面学习 React 常用技术栈
2. 深入理解 React 设计模式
3. 常见场景下的编程实战指南
4. 掌握用 React 开发大型项目的能力

立即扫码，免费试看

Topic

- 面临的挑战和方案的选择
- ReactNative在携程的使用现状
- CRN框架的优化
- 我们的经验与实践
- 总结与未来展望

为什么会使用RN

包
大小

占用AppSize小

性能
体验

用户体验佳

动态
更新

支持动态更新

开发
成本

相对较低

社区
资源

相对成熟，社区活跃

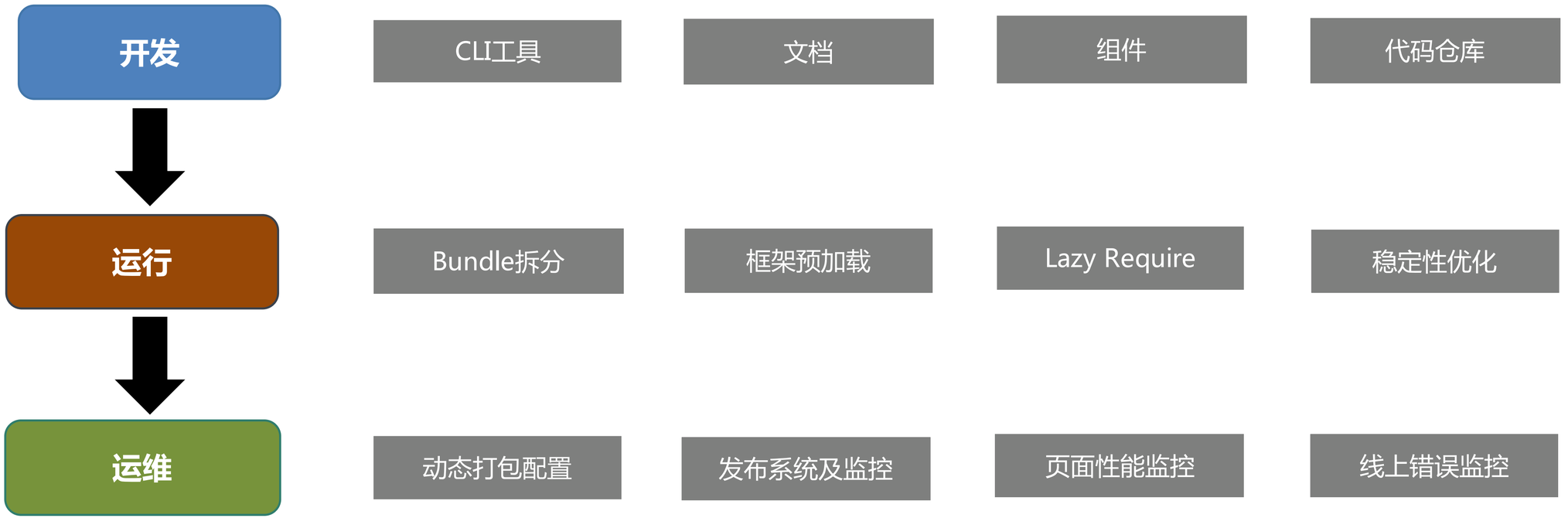
现状 – 业务广泛接入

- 携程旅行App，线上70+ RN业务模块
- 大量首页入口使用RN开发
- PV占比超过H5 Hybrid
- 集团内多App已接入使用

现状 – 业务深度使用

- 核心业务、全流程使用
  - Native转RN
  - Hybrid转RN
- 复杂业务高(火车票)
  - 710个JavaScript模块
  - 100个Page
  - 代码7z压缩后600KB

CRN框架介绍

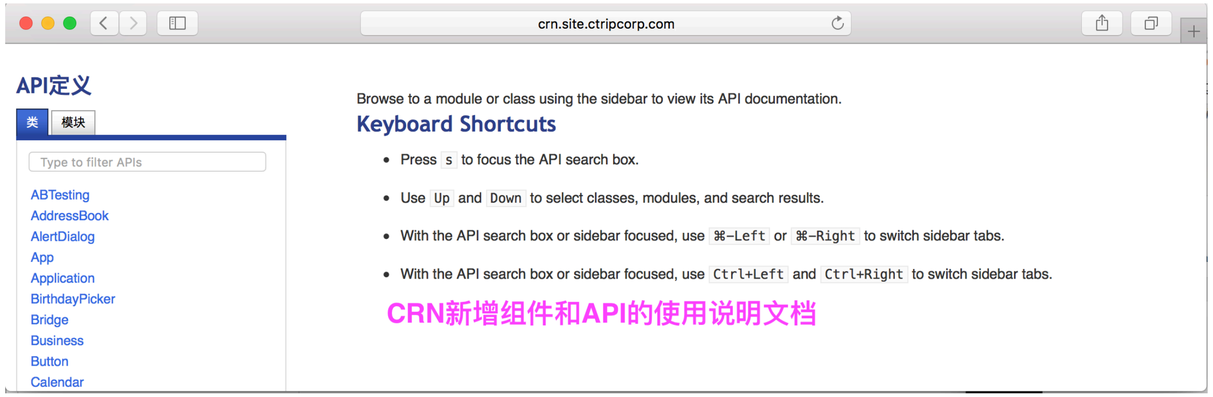


CRN是基于RN定制，以实现开发友好、运行稳定高效、运维可监控的大前端开发框架。

开发 – CLI和文档支持

➢ CLI和文档

```
msi-pc:~ jim$ crn
Usage: crn <command> [options]
Commands:
   init                  建立并初始化CRN工程，可指定AppId，默认为携程App
   start                 启动CRN服务,默认端口8081
   run-ios               运行指定AppId的IOS App
   run-android           运行指定AppId的Android App
   run-patch             执行patch,替换CRN修改过的lib文件
   openurl               在指定AppId的App中打开某个URL
   cli-update            更新cli版本
   web                   CRN Web命令入口
   example               创建CRN组件和API调用Demo工程
   aux                   增强型功能入口:log Server、本地打包、上传开发包
Options:
   -h, --help            显示命令帮助
   -v, --version         显示版本

http://crn.site.ctripcorp.com/, crn-cli@2.2.39
```



开发 – 组件扩展支持

- 70+ API和UI组件支持
- 重写fetch、listview等组件
- 扩展ScrollView组件
- 跨容器 (Native、Hybrid、RN)通讯支持
- 组件开发原则，RN优先，Native兜底

View组件

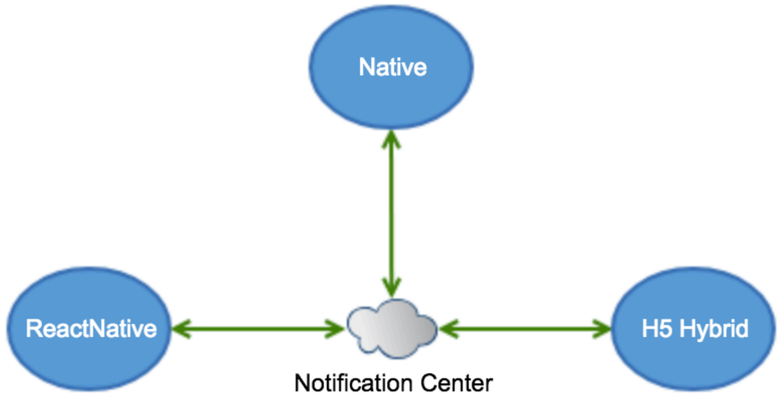
- AdView
- Button
- Carousel
- CRNListView
- CRNListViewDataSource
- DatePicker
- DatePickerWidget
- HeaderView
- HtmlText
- LinearGradient
- LoadControl
- LoadingFailedView
- LoadingNoDataView
- LoadingView
- MapView
- Page
- RefreshControl
- ScrollView
- SegmentedControl
- Slider
- SpriteImage
- SwipView
- IconfontView
- .....

API组件

- App
- ABTesting
- AddressBook
- Application
- Calendar
- Channel
- Device
- Encrypt
- Env
- Event
- Fetch
- ImagePicker
- Location
- Log
- Pay
- PhotoBrowser
- QRCode
- ScreenShot
- Share
- Storage
- Toast
- URL
- User
- Zip
- .....

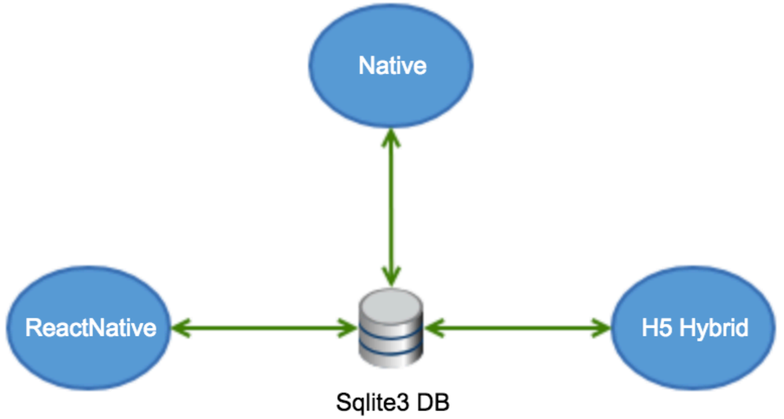
开发 – Event机制

- Event机制解决多端通讯问题
  - Event.addEventListener(name, callback)
  - Event.removeEventListener (name)
  - Event.sendEvent(name, data)
  - 基于Notification实现



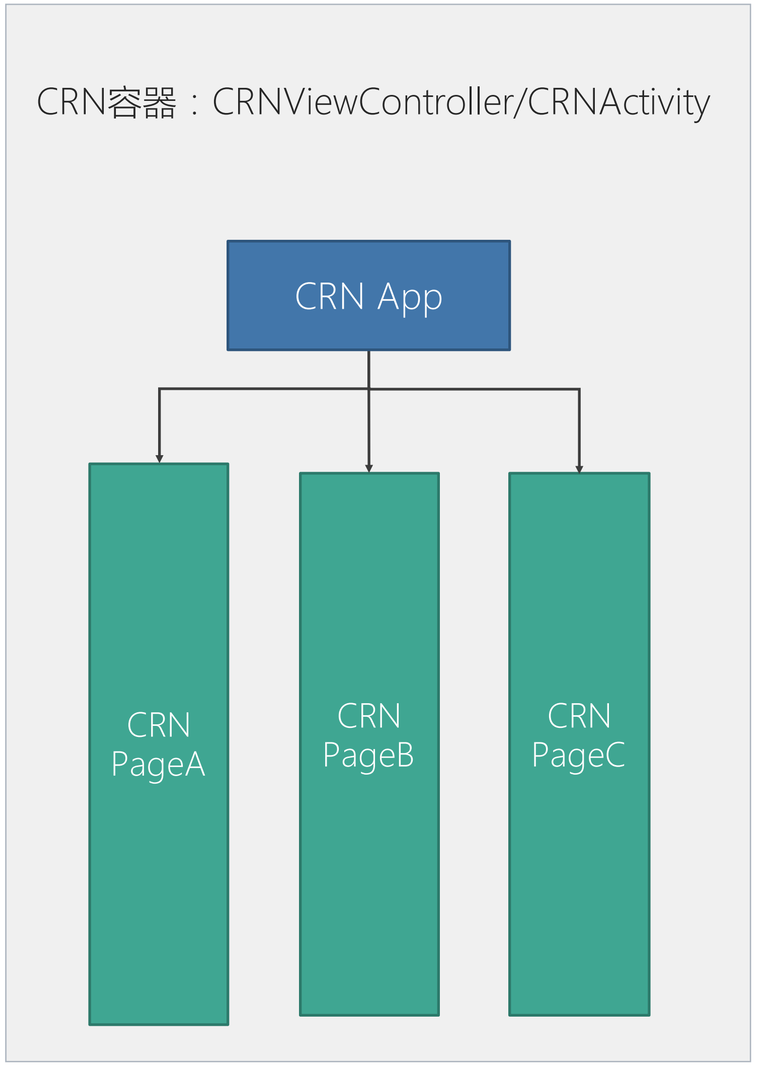
开发 – 统一Storage

- 统一Storage提供
  - Storage.load(params, callback)
  - Storage.remove(params)
  - Storage.save(params)
- 设计说明
  - 存取支持domain区分不同业务
  - 存取expireDate支持
  - 存取内置加解密支持



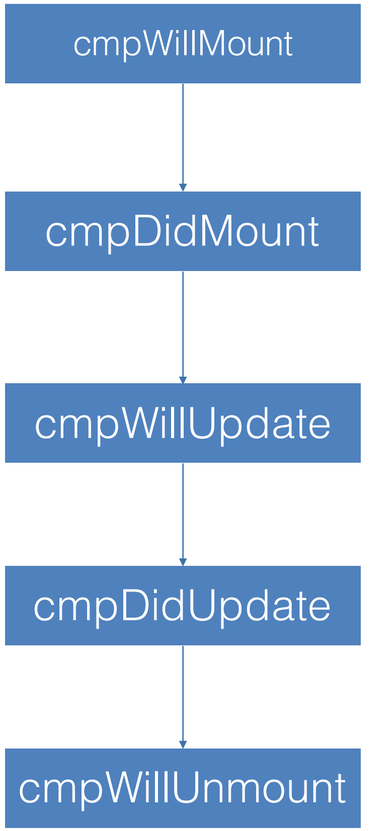
开发 – CRN Page模型

- CRN Page 模型
  - CRN 容器: 包含ReactRootView
  - 业务Page继承自CRN Page基类

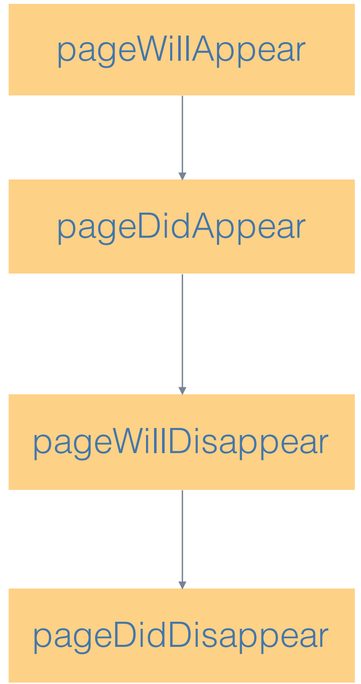


开发 – Page生命周期扩展

- CRN Page生命周期扩展
  - 所有CRN页面继承同一Page基类
  - 解决和Native/Hybrid/其它RN业务模块页面切换无回调问题
  - 解决App前后台切换无回调问题
  - 同一CRN容器内不同Page间切换



常规组件生命周期



CRN新增

运行 – Bundle拆分和Require优化

- 为什么要拆分
  - 简单HelloWorld项目release打包520KB，压缩后147KB
  - 大量业务模块上线，size占用是问题
- 为何要改造Require
  - 后台预加载提升首屏渲染时间
  - 支持动态按需加载

运行 – Bundle打包文件结构

- 官方命令打包出的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
```

1. Global Define/Require
.
.
2. Module define
.
.
3. Require()

Bundle结构抽象

运行 – Bundle代码执行顺序

- 官方命令打包出的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
```

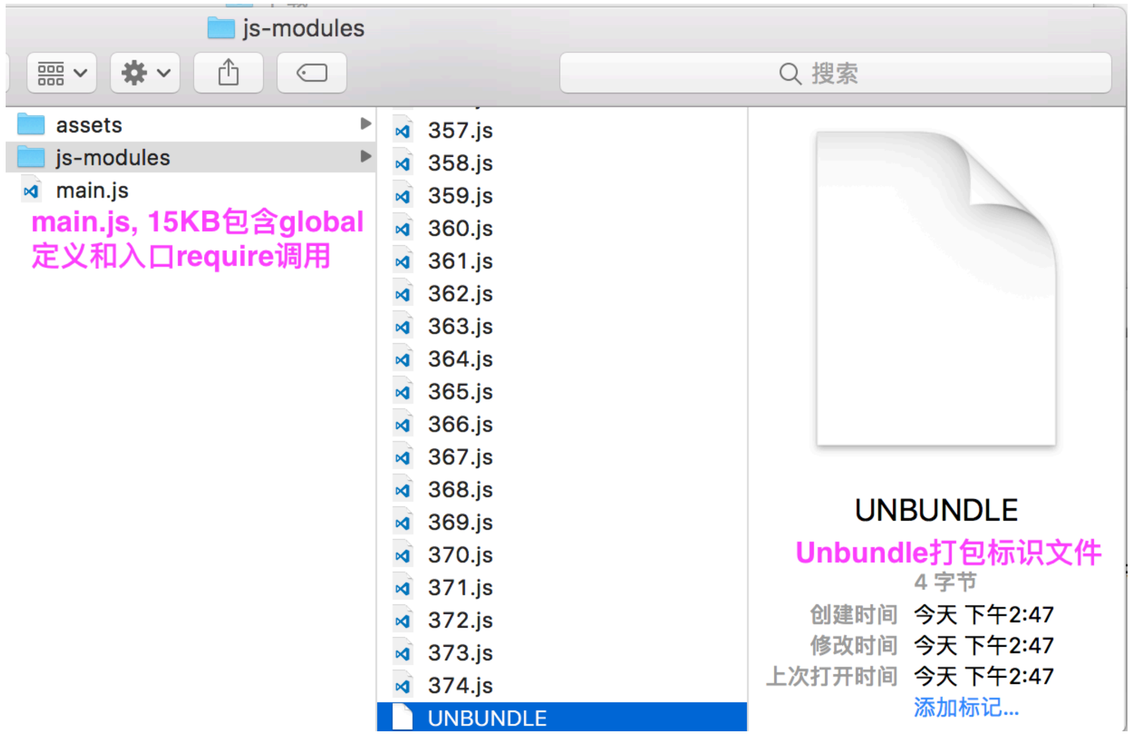
1. Global Define/Require
.
.
3. Require()
.
.
2. Module define

代码执行顺序

运行 – Android Unbundle打包

➢ RN Android Unbundle打包

```
/* 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
```



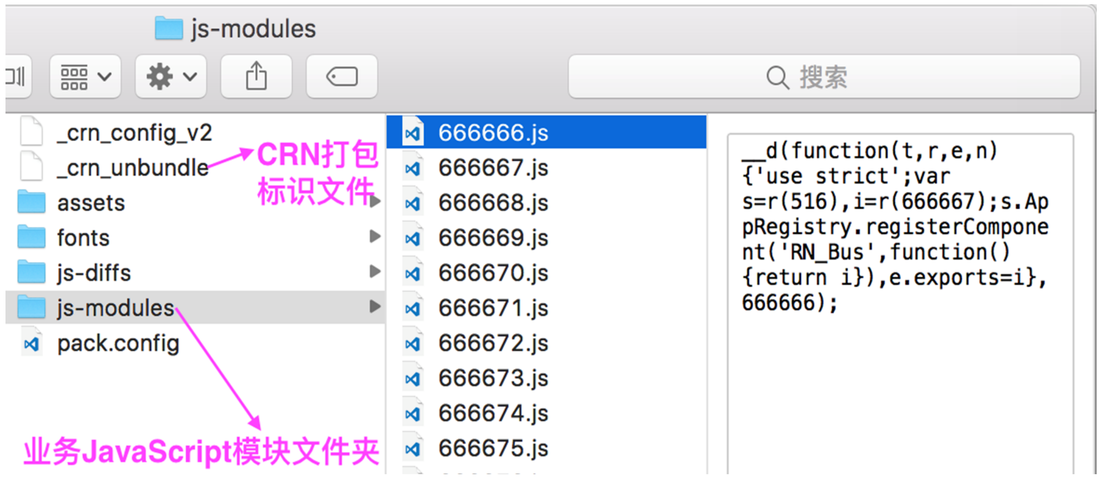
所有模块都打包成独立文件

运行 – CRN Bundle拆分

- CRN打包方案，合并common

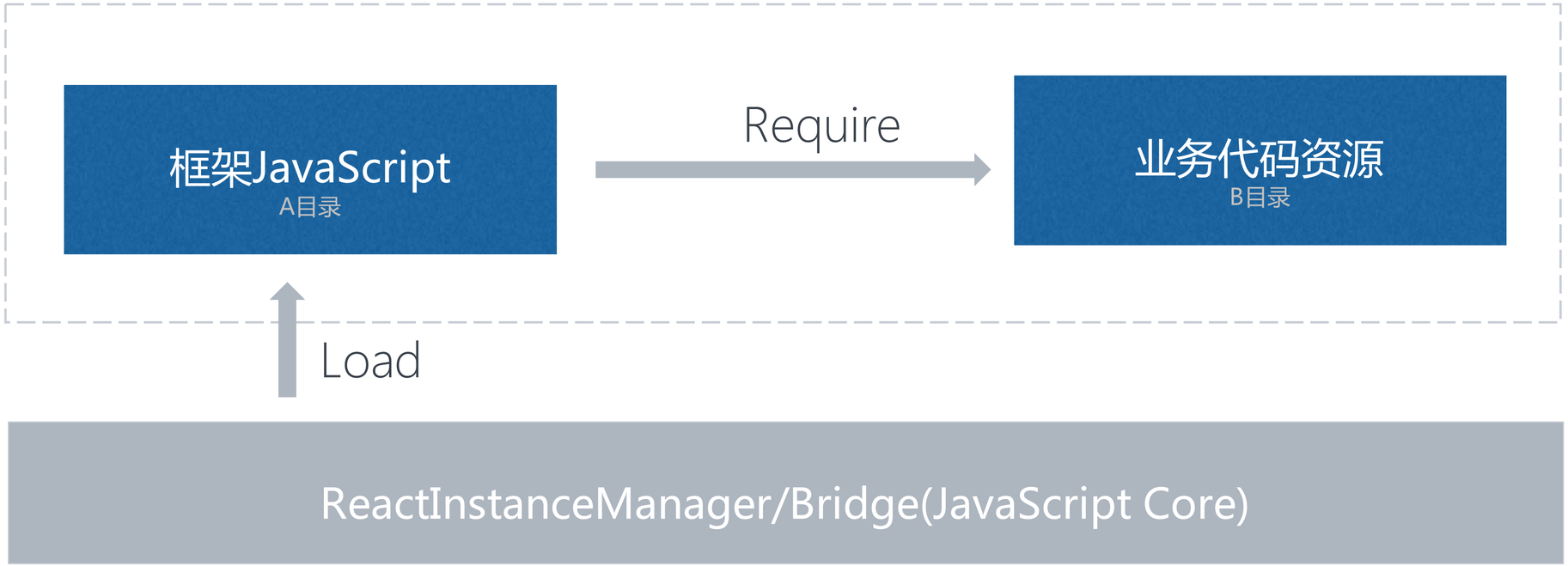
```
/* 1. 头部--全局定义部分 */
(function(global) {
global.__DEV__=true;
global.__BUNDLE_START_TIME__=Date.now();
})(typeof global !== 'undefined' ? global : typeof self !== 'undefined' ? self : this);


/* 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
```

官方命令打包出的bundle



运行 – Bundle拆分之后的加载

CRN Bundle拆分之后执行过程



运行– Bundle拆分之后的加载

CRN Native Require实现

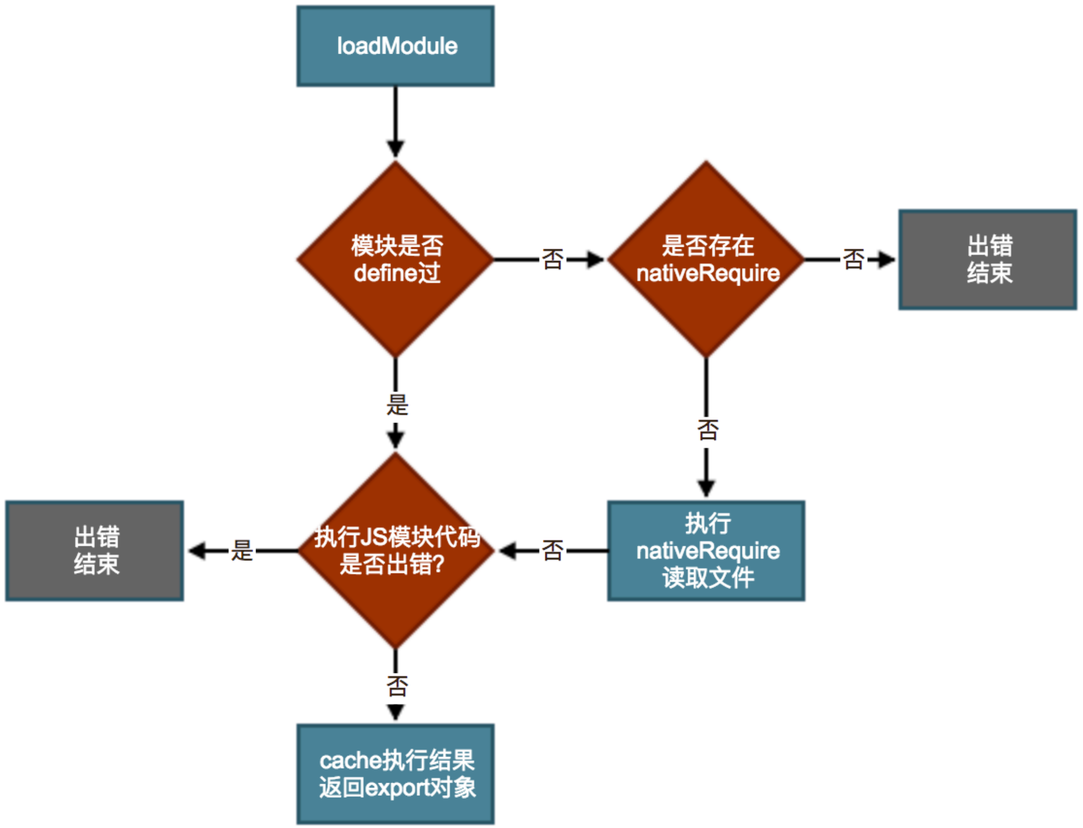
- 官方Android在unbundle打包的js-modules目录中查找执行
- 官方iOS在一个preparePack打包的二进制bundle中查找执行
- CRN改造iOS平台的nativeRequire，支持从磁盘目录查找和执行模块

运行 – JS模块加载

Define 和 Require

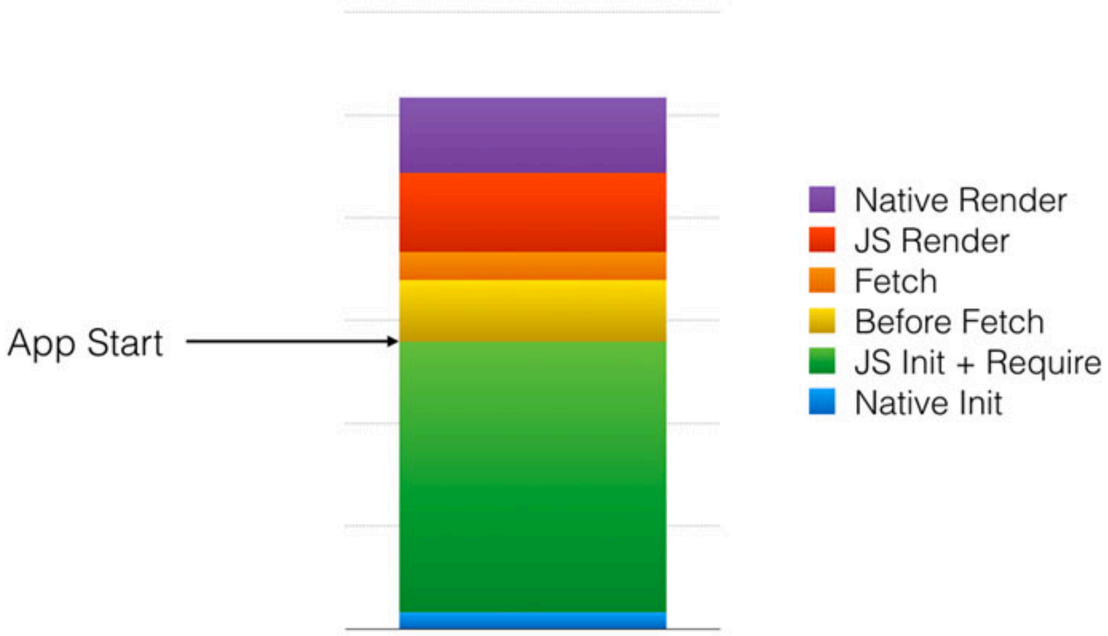
```
/*define 定义*/
function define(moduleId, factory) {
    if (moduleId in modules) {
        return;
    }
    modules[moduleId] = {
        factory: factory,
        hasError: false,
        isInitialized: false,
        exports: undefined
    };
}

/*require 定义*/
function require(moduleId) {
    var module = modules[moduleId];
    return (module && module.isInitialized) ?
        module.exports :
        guardedLoadModule(moduleId, module);
}
```

加载JS模块



运行 – 预加载

- 预加载
  - Require是代码真正执行的点
  - Require耗时长
  - 后台Require框架代码，提速首屏渲染

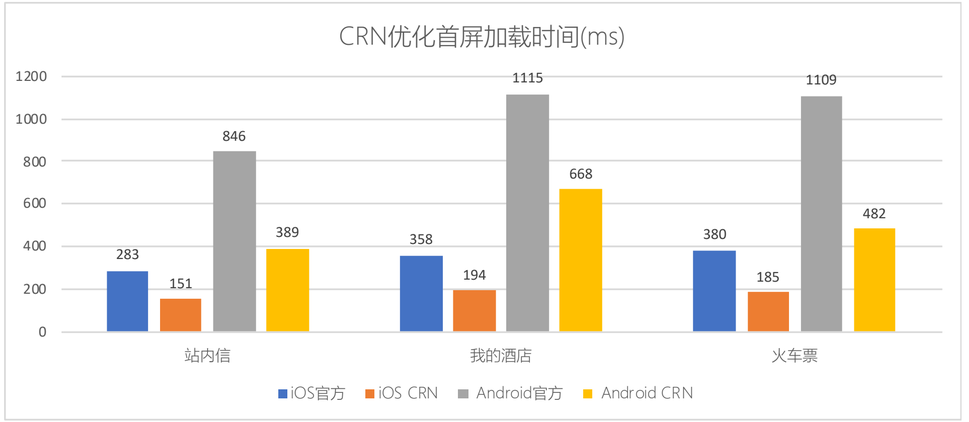


RN业务加载的耗时分布

运行 – 预加载提速首屏加载

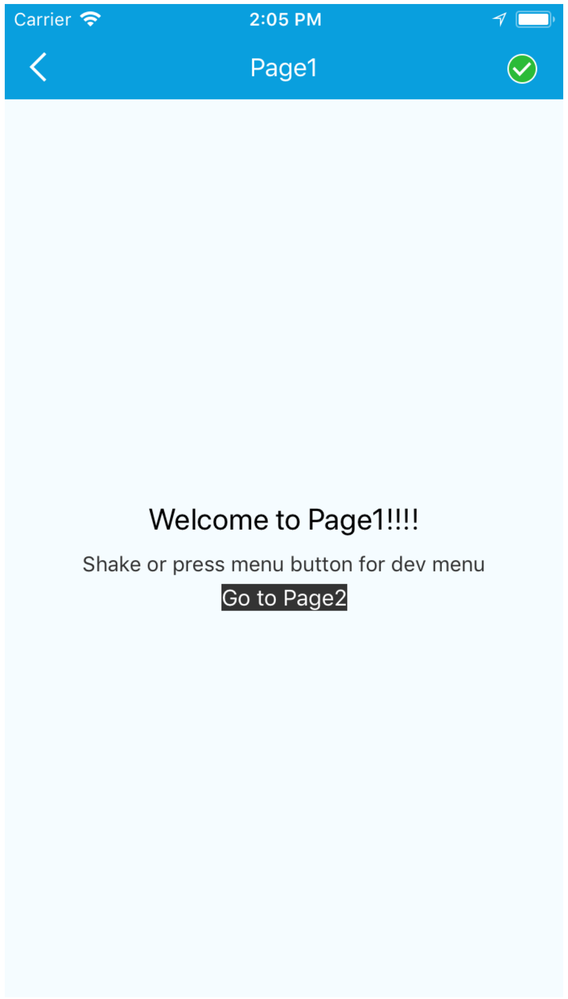
- 首屏加载提速
  - 后台创建instance（JavaScriptCore实例）加载框架代码
  - Instance被使用之后，创建新的
  - 首屏加载性能相较官方提升约45%

Instance(JSC)预创建流程

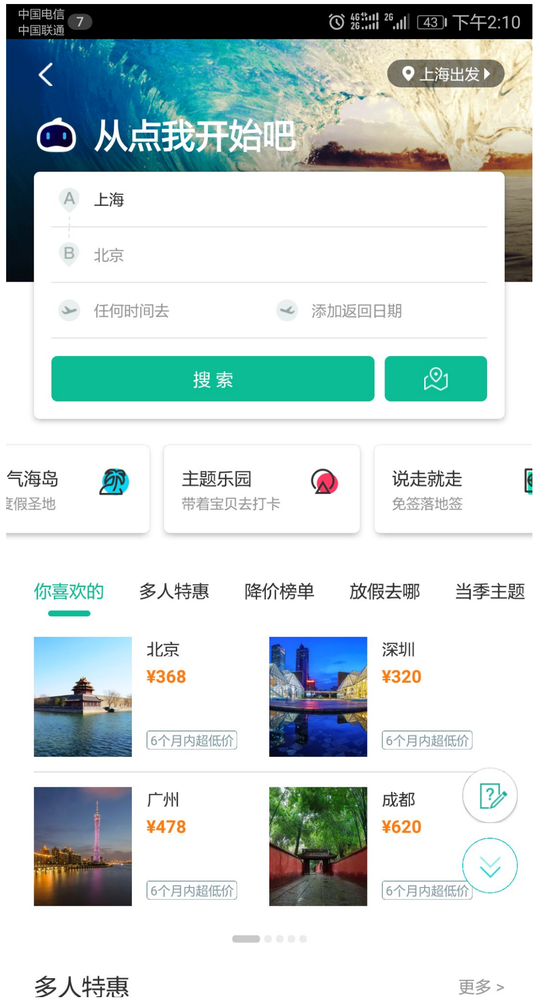


数据基于RN0.30，iPhone6\Sony Xperia多次测试

运行 – 业务复杂度增加导致的问题



简单页面，JS模块少



全业务使用，JS模块多，首屏加载依旧缓慢

运行 – 业务复杂度增加导致的问题

- 首屏加载慢问题重现
  - 业务复杂度增加，代码量剧增
  - import代码在打包过程会变成require

- 解决方案
  - 进入业务时，尽可能少执行require

```
import PageA from ("pages/PageA");
import PageB from ("pages/PageB");
import PageC from ("pages/PageC");
import PageD from ("pages/PageD");
//设置页面路由表
let pageList = [PageA, PageB, PageC, PageD];
App.startApp(pageList);
```

import模块的内部处理流程

运行 – LazyRequire方案

- CRN LazyRequire实现
  - 开发阶段，LazyRequire等价于require
  - 接入阶段，CRN Page 只需替换import
  - 打包阶段，Babel插件将lazyRequire参数从path替换成模块ID
  - 运行阶段，Page在切换时候，框架load

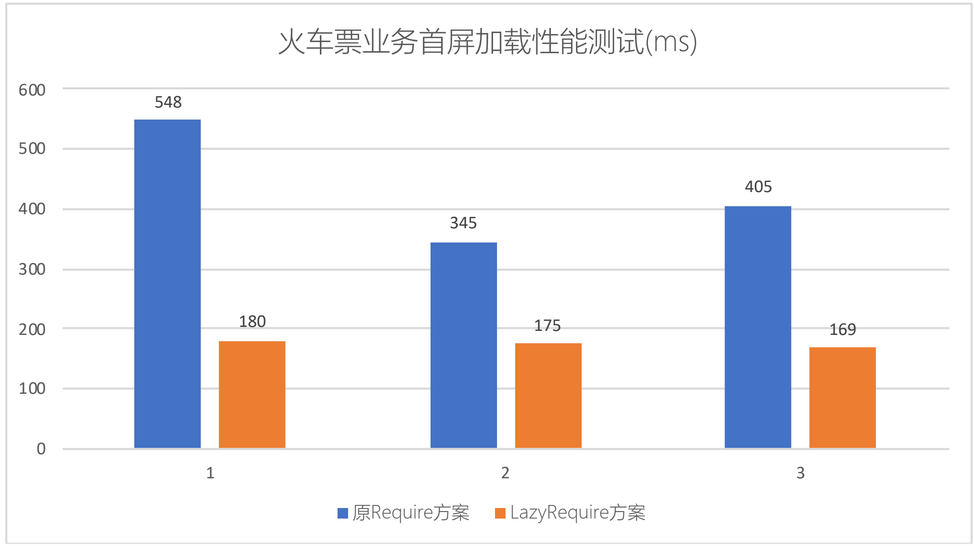
```
const PageA = lazyRequire("pages/PageA")
const PageB = lazyRequire("pages/PageB")
const PageC = lazyRequire("pages/PageC")
const PageD = lazyRequire("pages/PageD")
//设置页面路由表
let pageList = [PageA, PageB, PageC, PageD];
App.startApp(pageList);
```

```
//lazyRequire初始化
LazyModule lazyRequire(modulePath);

//lazyRequire定义
LazyModule = {
    load(); //执行真正的模块代码，返回执行结果
};
```

LazyRequire的内部实现

运行 – LazyRequire方案

- 线上数据
  - 复杂业务首屏加载提升明显(右图)
  - 彻底解决了复杂业务使用CRN的问题



数据基于RN0.41，iPhone Simulator多次测试

运行 – 其它Require优化方案

- inlineRequire方案
  - 定义全局变量，使用时，require并赋值

- ES5构造函数
  - ReactNative内部组件以此方式导出
  - 推荐使用这种方式

```
//import VeryExpensiveModule from './VeryExpensive';
                                                    import静态加载模块
let VeryExpensiveModule = null;

export default class Optimized extends Component {
  someEvent = () => {
    if (VeryExpensiveModule == null) {
      VeryExpensiveModule = require('./VeryExpensive').default;
    }                                    require('path').default, 动态执行模块代码
  };
}
```

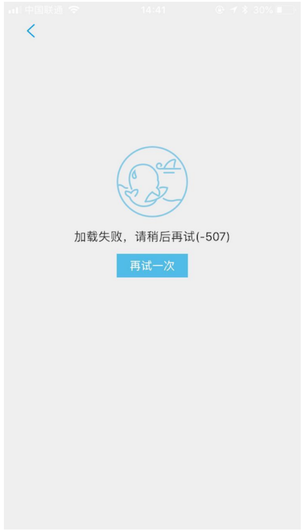
```
const ReactNative = {
  get AccessibilityInfo() { return require('AccessibilityInfo'); },
  get ActivityIndicator() { return require('ActivityIndicator'); },
  get ART() { return require('ReactNativeART'); },
  get Button() { return require('Button'); },
  get CheckBox() { return require('CheckBox'); },
  get DatePickerIOS() { return require('DatePickerIOS'); },
  get DrawerLayoutAndroid() { return require('DrawerLayoutAndroid'); },
  get FlatList() { return require('FlatList'); },  构造函数方式返回模块
  get Image() { return require('Image'); },
```

稳定性优化 – iOS

- 框架层
  - 设置错误处理handler
  - 不设置在生产环境会crash
  - 记录错误来源

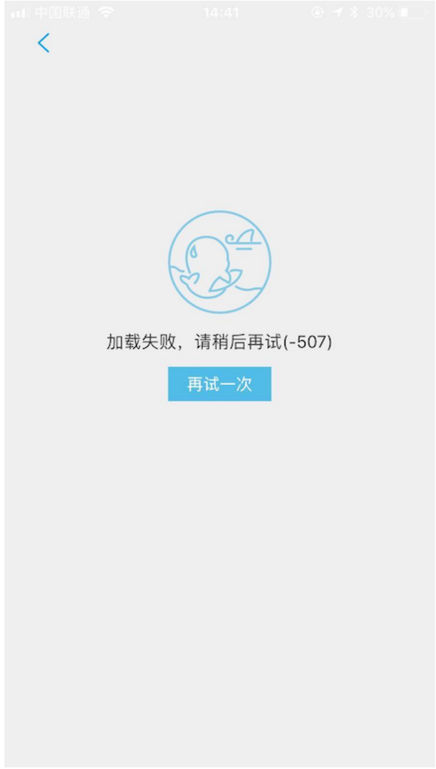
- 业务层
  - 出错显示出错浮层，支持重试

```
//错误Handler定义
typedef void (^RCTFatalHandler)(NSError *error);

//使用者需要设置错误handler，捕获错误日志
RCT_EXTERN void RCTSetFatalHandler(RCTFatalHandler fatalHandler);

/**
 * Report a fatal condition when executing. These calls will _NOT_ be compiled out
 * in production, and crash the app by default. You can customize the fatal behaviour
 * by setting a custom fatal handler through `RCTSetFatalHandler`.
 */
RCT_EXTERN void RCTFatal(NSError *error);
```



稳定性优化 – Android

- JavaScript层
  - DevSupportManager
- Native层
  - NativeModuleCallExceptionHandler
  - SO加载优化
  - 页面长时间加载不成功的优化

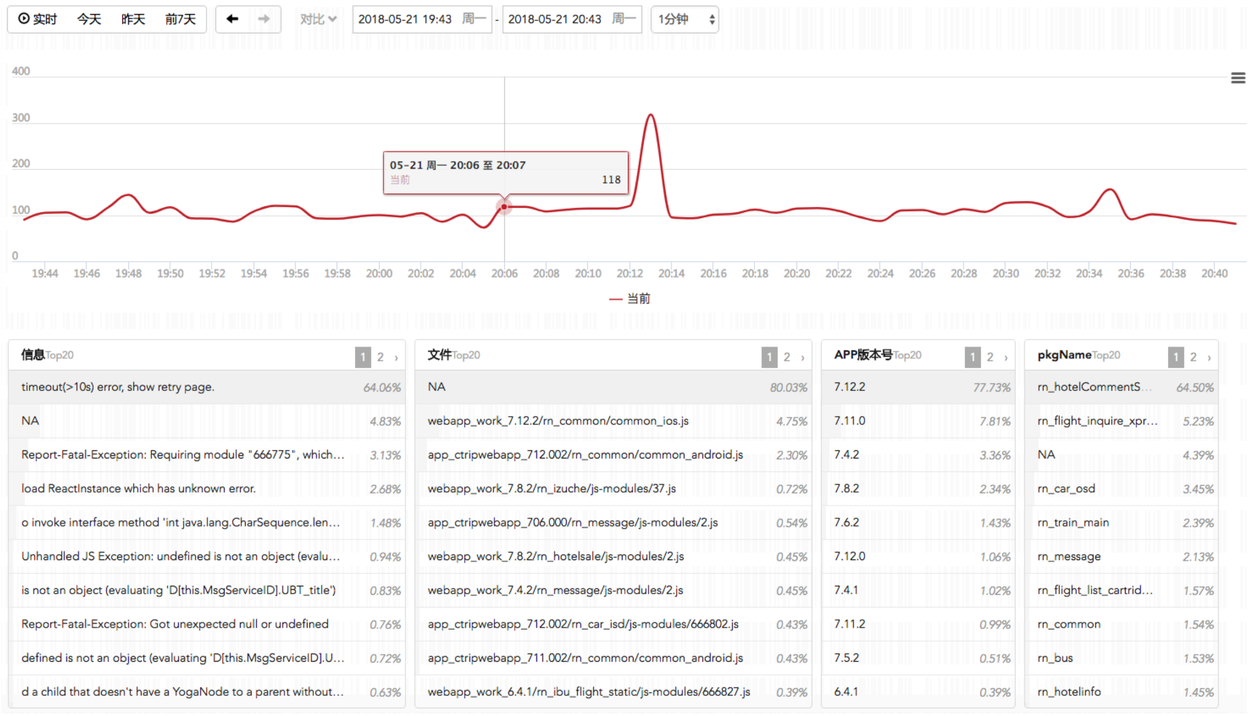


稳定性优化 – 降级处理

- 优雅降级
  - 自动降级参数传递的风险
  - 降级策略定制
- 具体方案
  - CRN原始URL：/rn_abc/index?pa=1&pb=2&px=x
  - 降级H5站点URL：http://m.ctrip.com/webapp/abc?pa=1&pb=2&px=x
  - 降级方案URL:/rn_abc/index?pa=1&pb=2&px=x&bakURL=base64Encode(http://m.ctrip.com/webapp/abc?pa=1&pb=2&px=x)

运维 – 错误日志收集

- 错误日志收集系统
  - 实时上报错误JS/Native异常日志
  - 完整callstack展示
  - 可按照模块名、版本过滤
  - 支持报警配置
  - 发布之后需关注数据走势

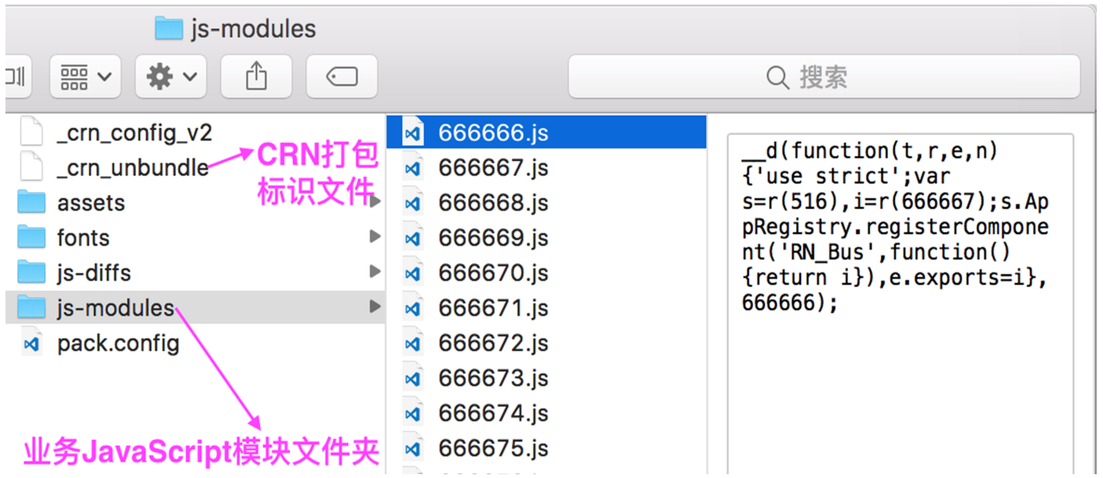


运维 – 性能统计

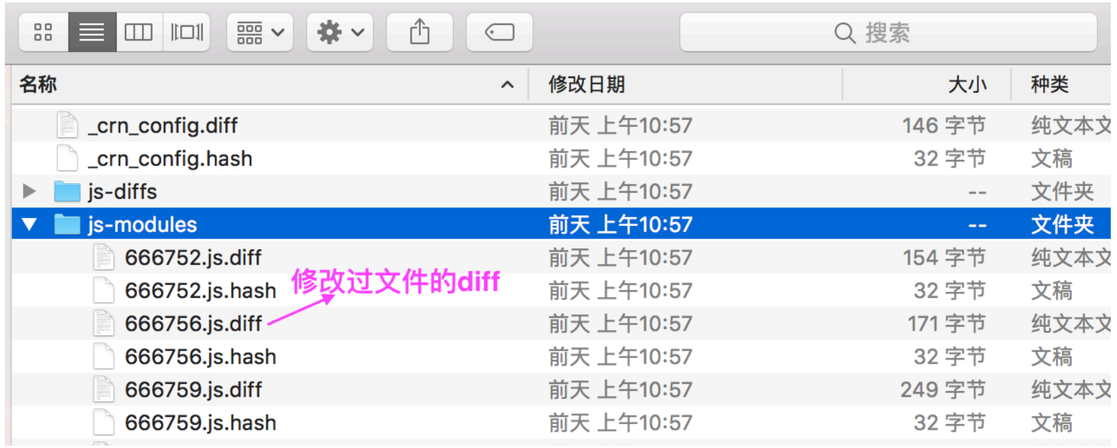
- CRN性能统计系统
  - 按业务模块，版本统计首屏时间
  - 加载耗时比例分布展示

运维 – 发布增量包优化

- Diff方案
  - 对zip包内部的每一项进行bsdiff，保留变化过的文件，然后7z压缩

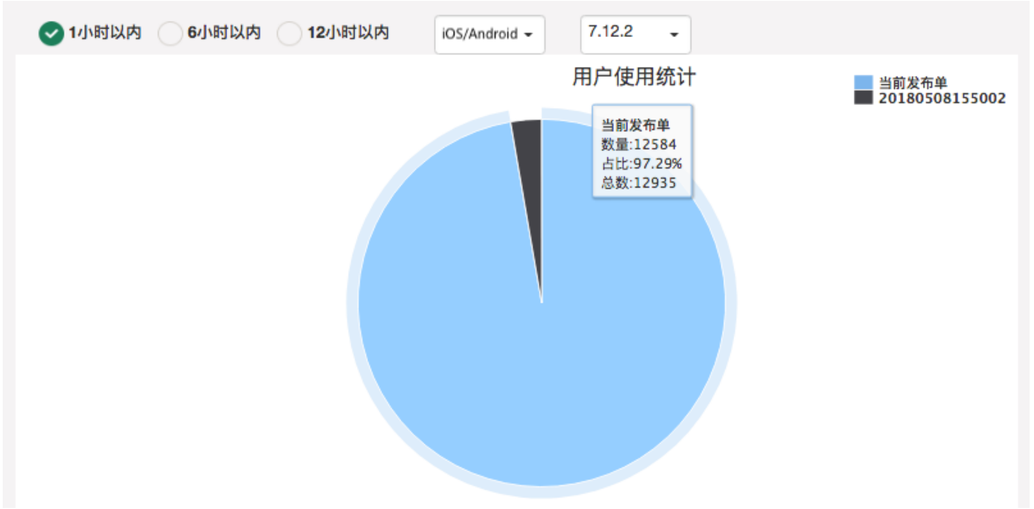


- 优化点
  - 充分利用CRN包格式优势
  - 保留业务模块id的mapping，避免新增模块而导致模块id全乱的情况



运维 – 发布增量包优化

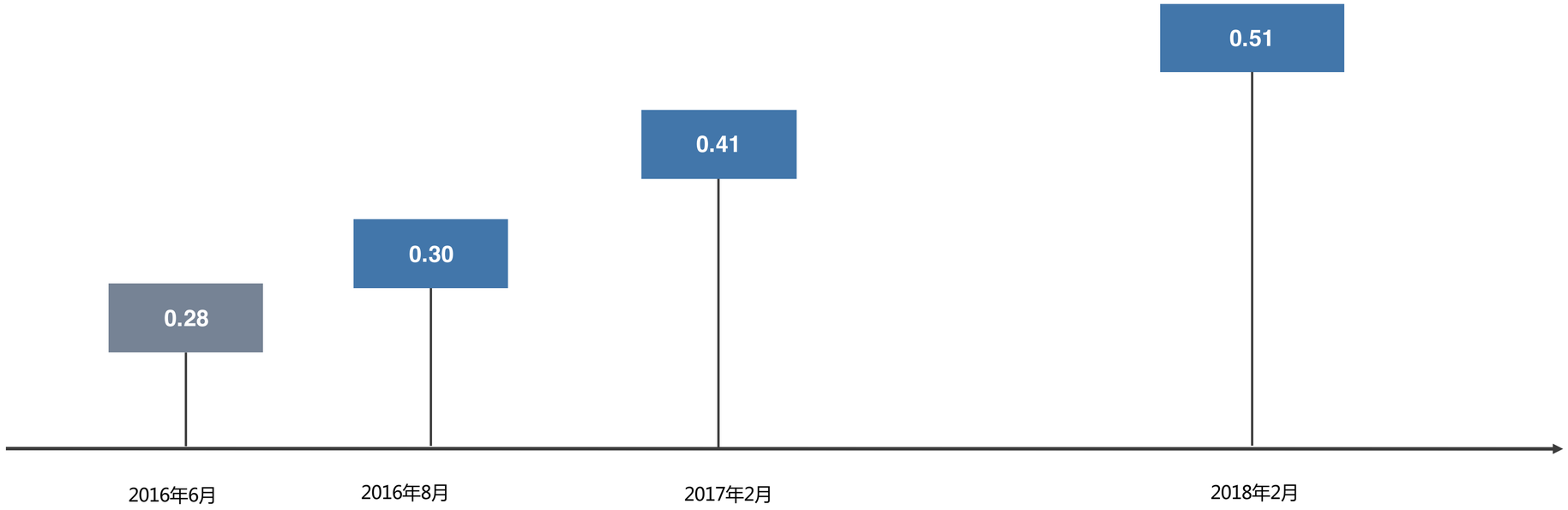
- 线上数据
  - 线上22个CRN增量包，总计325KB，平均每个包15KB
  - 发布后一小时，新包使用率达97-98%



实践经验

- 版本升级
- 组件依赖管理
- CRN包的依赖管理
- 分平台打包
- 导航栏的选择

CRN版本升级



CRN版本升级历程

CRN版本升级

➢ 升级流程

➢ 升级成本

- 框架团队成本2-3周
- 业务升级一周完成

➢ 其它

- 升级方案尽可能对业务简单
- 避免业务包做跨RN版本的发布
- 升级频率8-12个月/次

组件依赖管理

- 组件仓库
  - CRN 仓库: 常用组件，必需依赖
  - CRN_Ext仓库: 独立组件，按需依赖
  - 内网npm仓库：第三方模块，按需
  - CRN相关仓库，使用git分支依赖
- 版本管理
  - 固定版本，避免使用^、*
  - 构建过程检查依赖的版本

```
"dependencies": {
  "@ctrip/crn": "git+http://[redacted]/crn#rel/7.7",
  "react": "15.4.2",
  "react-native": "0.41.0",
  "react-native-swiper": "^1.5.4"
},
```

react-native-swiper

Readme | 1 Dependencies | 54 Dependents

Version History

1.5.9	10 months ago
1.5.8	10 months ago
1.5.7	10 months ago
1.5.5	10 months ago
1.5.4	2 years ago
1.5.3	2 years ago
1.5.2	2 years ago

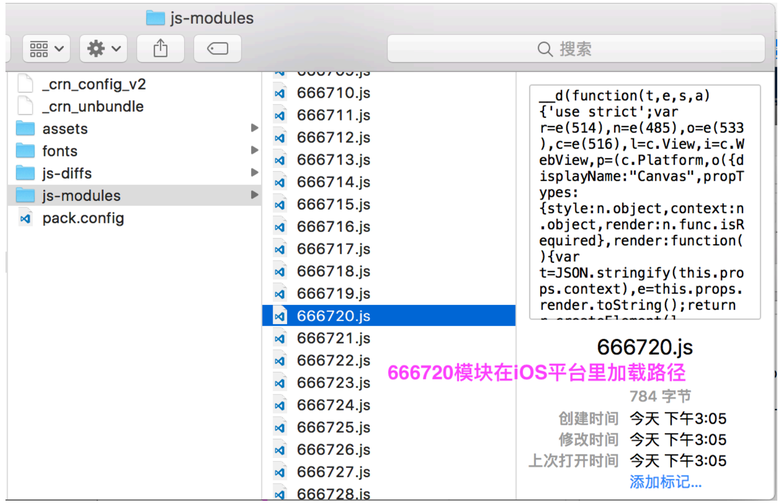
CRN包的依赖管理

- 依赖规则
  - 业务包只依赖框架包
  - 业务包之间平行，不允许依赖

- 其它
  - 发布高到达率情况下， 不提供强制更新功能
  - 不同业务包之间跳转，会触发最新增量包下载

分平台打包

- 原方案
  - CRN框架包分开打包
  - 业务包共用，以iOS平台打包
  - 正常运行一年多
- 问题暴露
  - 平台独立组件，打包产物不一致
  - 分2次打包，打包之后merge打包产物
  - Android优先在js-diffs中查找模块



导航栏的选择

- 官方Navigator
  - 0.44开始废弃
  - 复杂页面切换时，卡顿明显
  - CRN使用该方案，优化了页面切换动化
- 第三方ReactNavigator（推荐）
  - 纯JavaScript实现，性能优秀，功能强大
  - 页面切换使用animated动画
- 第三方ReactNativeNavigator
  - Native实现，跨平台兼容性有些问题

Navigator

CRN优化

总结

- CRN框架对RN进行改造优化，解决了性能和稳定性两大核心问题，且在业务团队广泛深度使用

- CRNWEB基本成熟，已有10个左右业务模块接入，未来会大力推广，CRN(WEB)一体化的解决方案，进一步降低开发维护成本

- 持续优化RN性能和稳定性，分享CRN的优化思路和方案，适时开源部分组件和方案

极客邦企业培训与咨询

帮助企业与技术人成长

精品课程

Excellent Course

- ☑《互联网大规模分布式架构设计与实践》
- ☑《基于大数据的企业运营与精准营销》
- ☑《大数据和人工智能在金融领域的应用》
- ☑《区块链应用与开发技术高级培训》
- ☑《通往卓越管理的阶梯》

扫码关注官方微信服务号
了解更多课程详细信息

Geekbang
极客邦科技

QA