React Native 全量化实践

WEB 技术栈打造移动研发新模式

陈敏亮

Dear Developer,

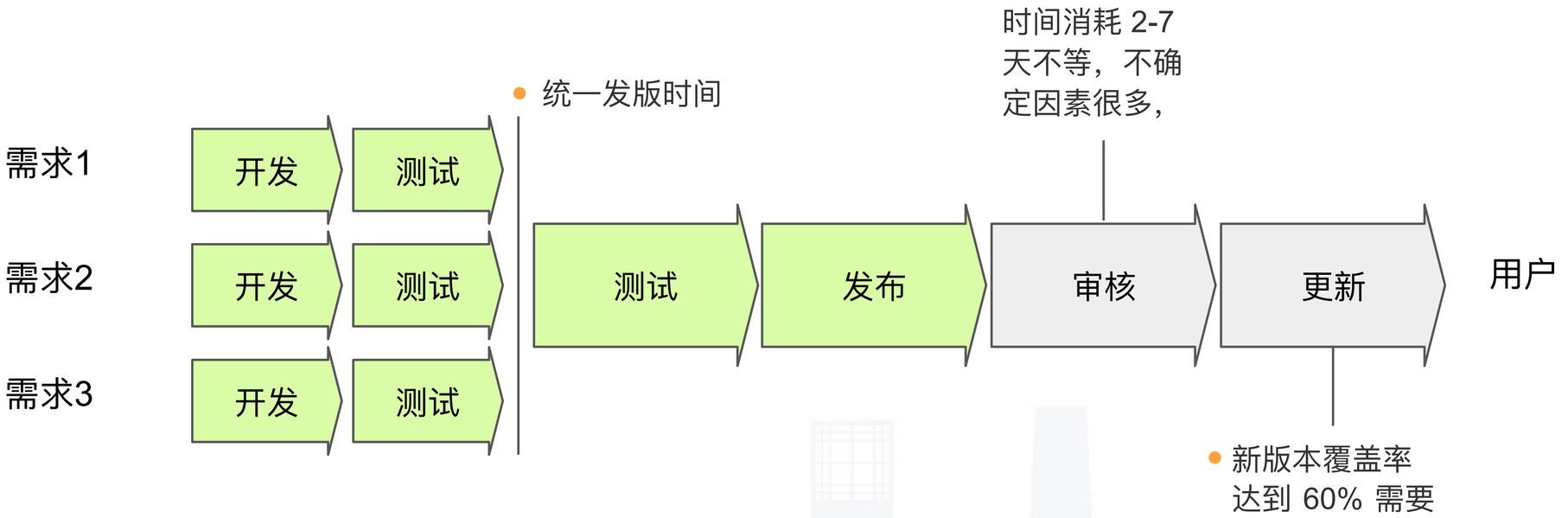
Your app, extension, and/or linked framework appears to contain code designed explicitly with the capability to change your app's behavior or functionality after App Review approval, which is not in compliance with section 3.3.2 of the Apple Developer Program License Agreement and App Store Review Guideline 2.5.2. This code, combined with a remote resource, can facilitate significant changes to your app's behavior compared to when it was initially reviewed for the App Store. While you may not be using this functionality currently, it has the potential to load private frameworks, private methods, and enable future feature changes.

This includes any code which passes arbitrary parameters to dynamic methods such as dlopen(), dlsym(), respondsToSelector:, performSelector:, method_exchangeImplementations(), and running remote scripts in order to change app behavior or call SPI, based on the contents of the downloaded script. Even if the remote resource is not intentionally malicious, it could easily be hijacked via a Man In The Middle (MiTM) attack, which can pose a serious security vulnerability to users of your app.

⚠️ Apple 警告邮件

3.8

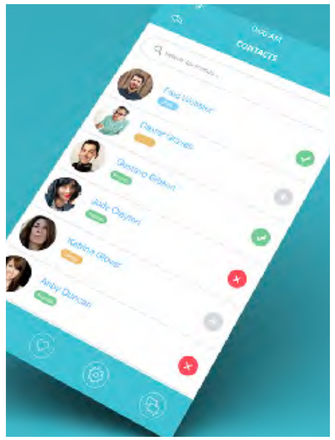
传统客户端研发 - 问题之一



- 每个迭代需要在审核与版本更新等非可控流程上消耗**超过一周时间**
- 审核与更新慢会严重放大线上问题，通过增加测试时间和测试流程可以解决这一问题，但这使得整体迭代周期拉得更长

传统客户端研发 - 问题之二

传统客户端研发 - 问题之二



需求1 开发 测试

需求2 开发 测试

需求3 开发 测试

测试 发布 审核 更新

需求1 开发 测试

需求2 开发 测试

需求3 开发 测试

测试 发布 审核 更新

传统客户端研发 - 问题

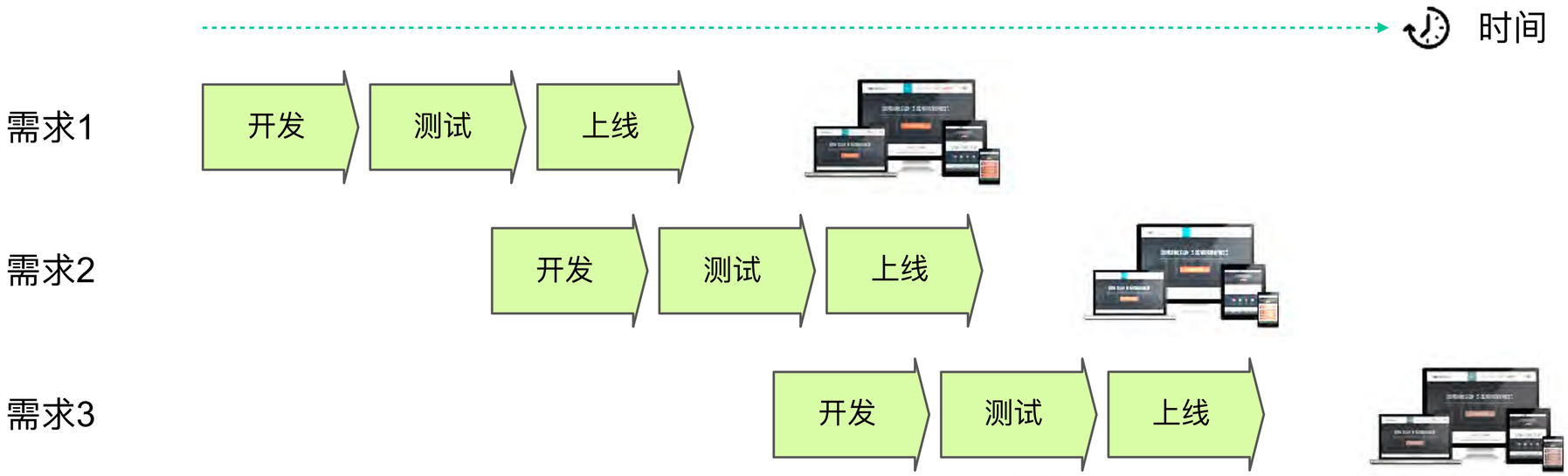
迭代低效

重复劳动

传统客户端研发问题 VS 创新产品

- 资源有限 - 更少的资源做更多的事情
- 时间窗口有限 - 更短的时间做更多的迭代
- 上述问题对于创新产品愈发严重

WEB 业务迭代流程的启示



- 无审核、更新环节，上线即生效，周期短，成本极低
- 按模块上线，无版本概念
- 迭代流程可控，一天可多次上线
- 无需区分平台

整体 WEB 化思路

技术层面

协作层面

为什么是 React Native

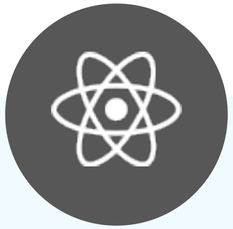
团队诉求与特点

- ☆ 资源有限 - 跨平台
- ☆ 时间窗口有限 - 追求更高的迭代效率
- ☆ 优秀的客户端团队
- ☆ 优秀的 WEB 前端团队

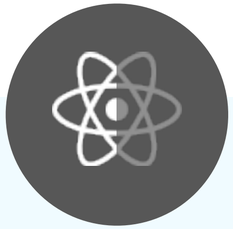
React Native 特点

- ☆ 相对靠谱的动态化方案
- ☆ 高度使用已有 WEB 技术栈

全量使用 OR 部分使用



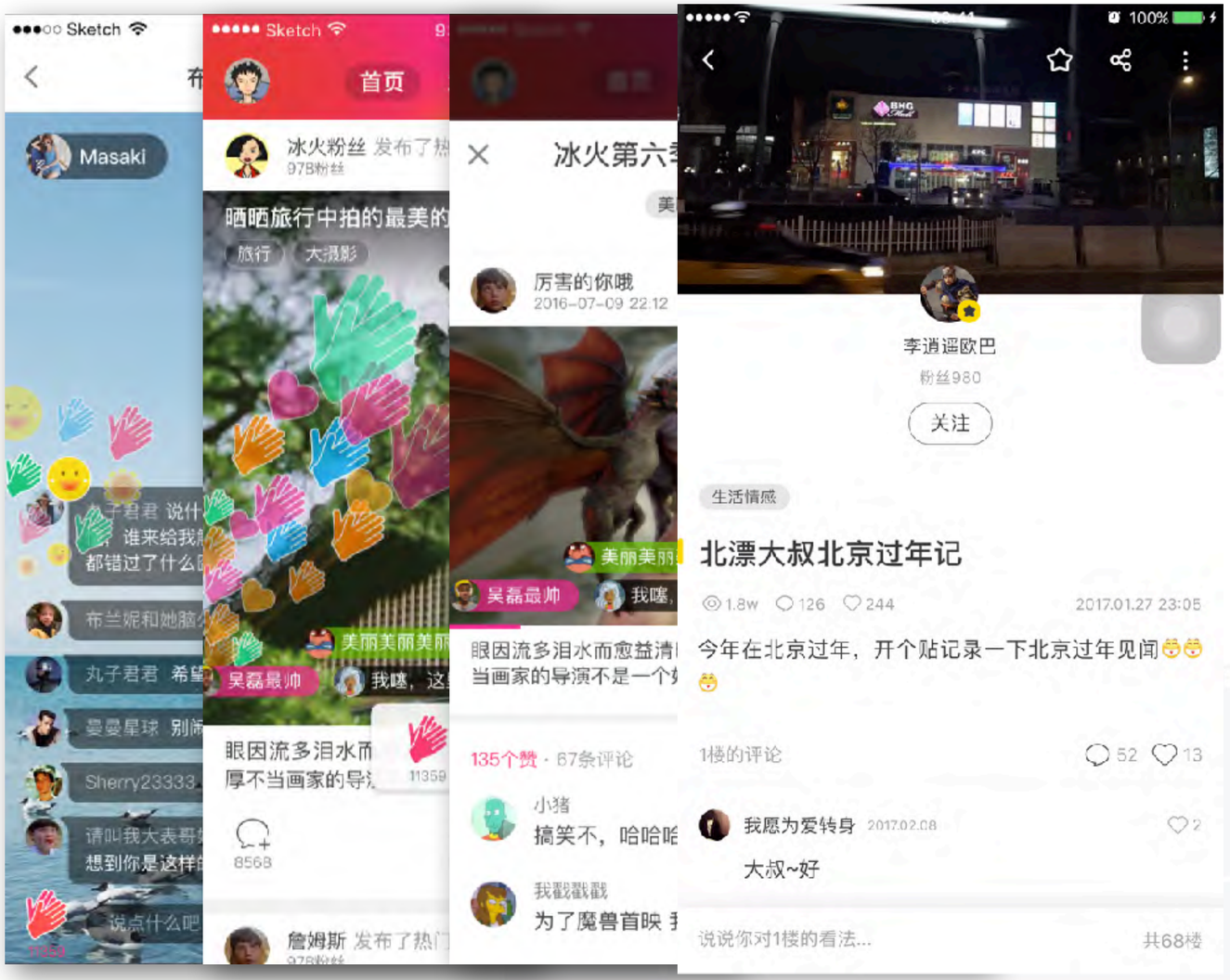
- 业务层实现采用 JS 优先原则
- 最大程度实现跨平台与动态化
- 绝大部分业务需求可实现 WEB 化迭代



- 部分非核心需求采用 React Native 实现
- 只支持部分功能的热更新
- 主体迭代流程依然是传统客户端方式

挑战

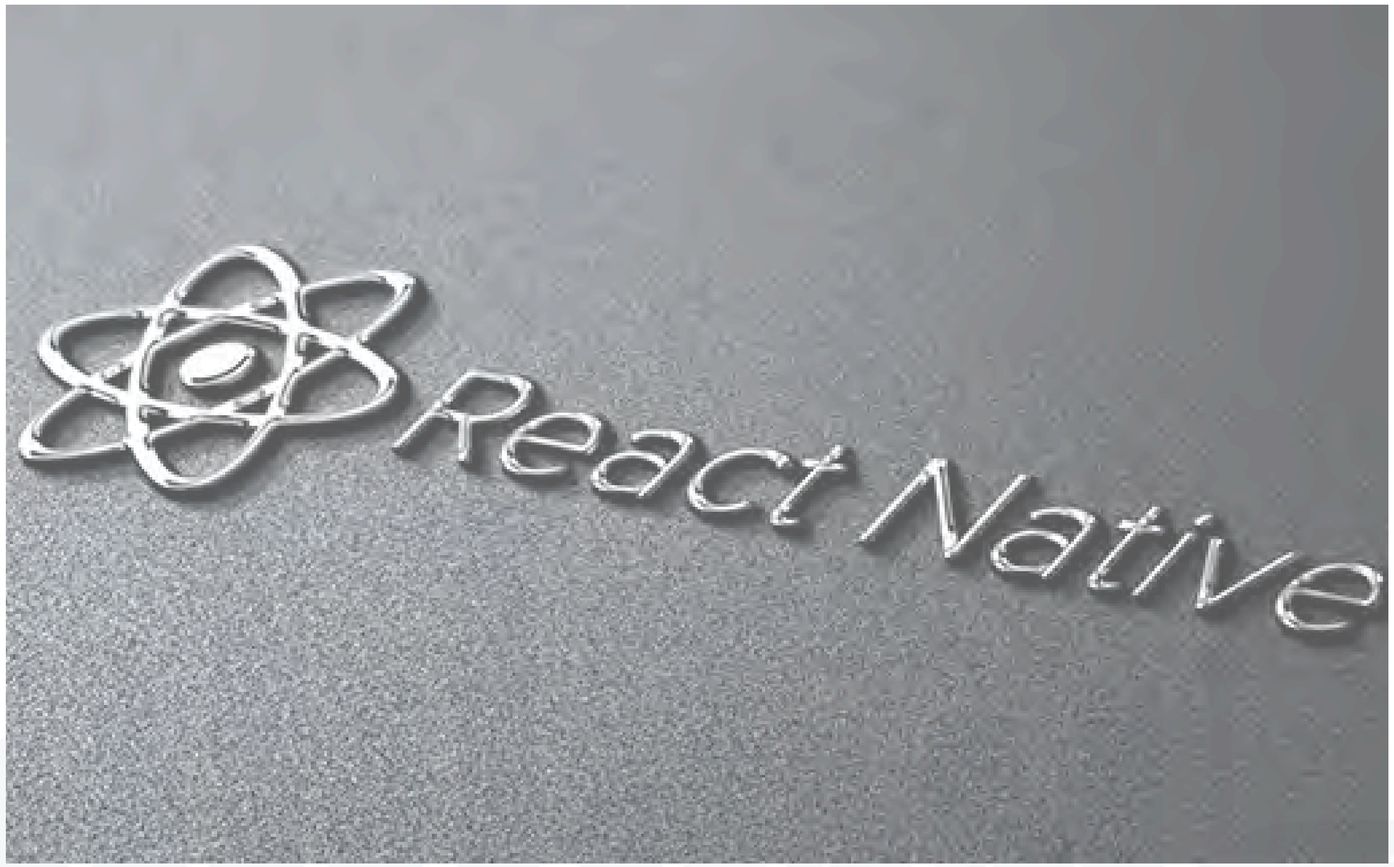
产品层面



- ☆ 媒体多样性
  - ☆ 视频
  - ☆ 图片
  - ☆ 资源
- ☆ 功能多样性
  - ☆ 聊天室
  - ☆ 富文本编辑
  - ☆ 长内容
- ☆ 交互多样性
  - ☆ 交互弹幕
  - ☆ 动画特效

技术层面

- ☆ 工程化设施匮乏
- ☆ 复杂交互场景下的性能瓶颈
- ☆ 框架本身的稳定性与跨平台兼容性



解决思路

技术定位

更丰富的研发技术栈，找准各自定位

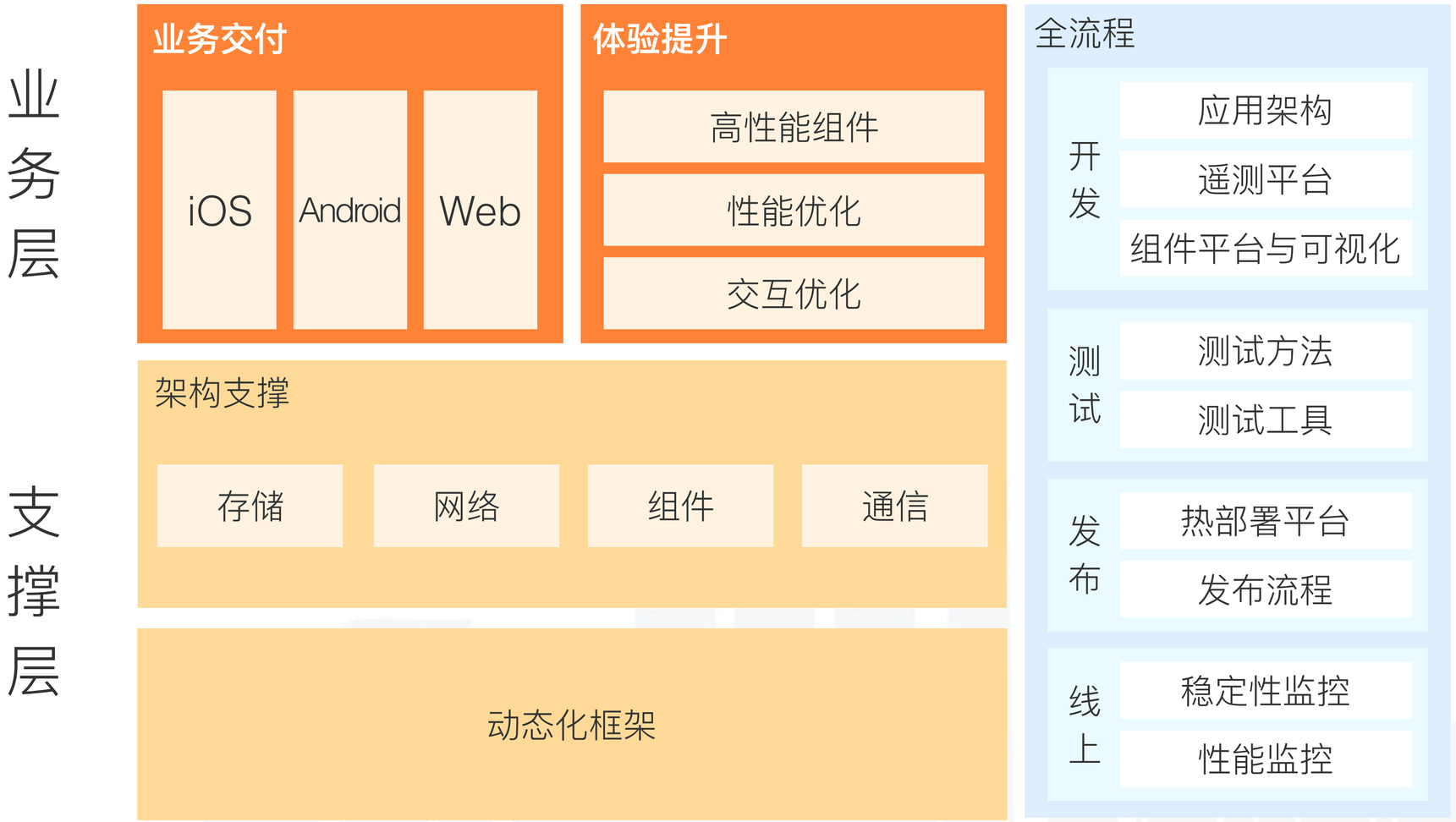
技术生态

构建客户端动态化研发全流程，完善技术生态

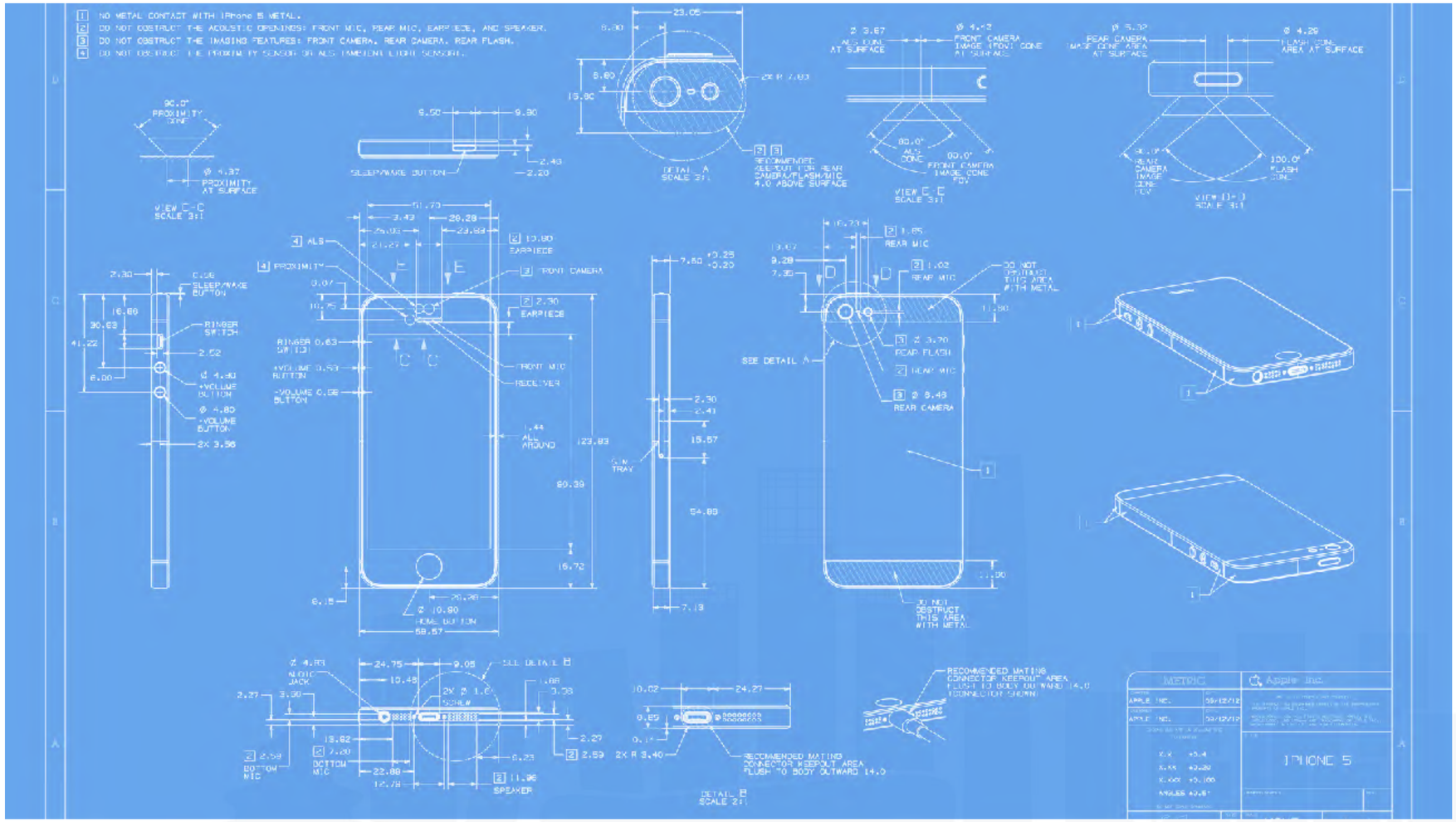
技术格局

突破 React Native 现有技术格局，按需定制

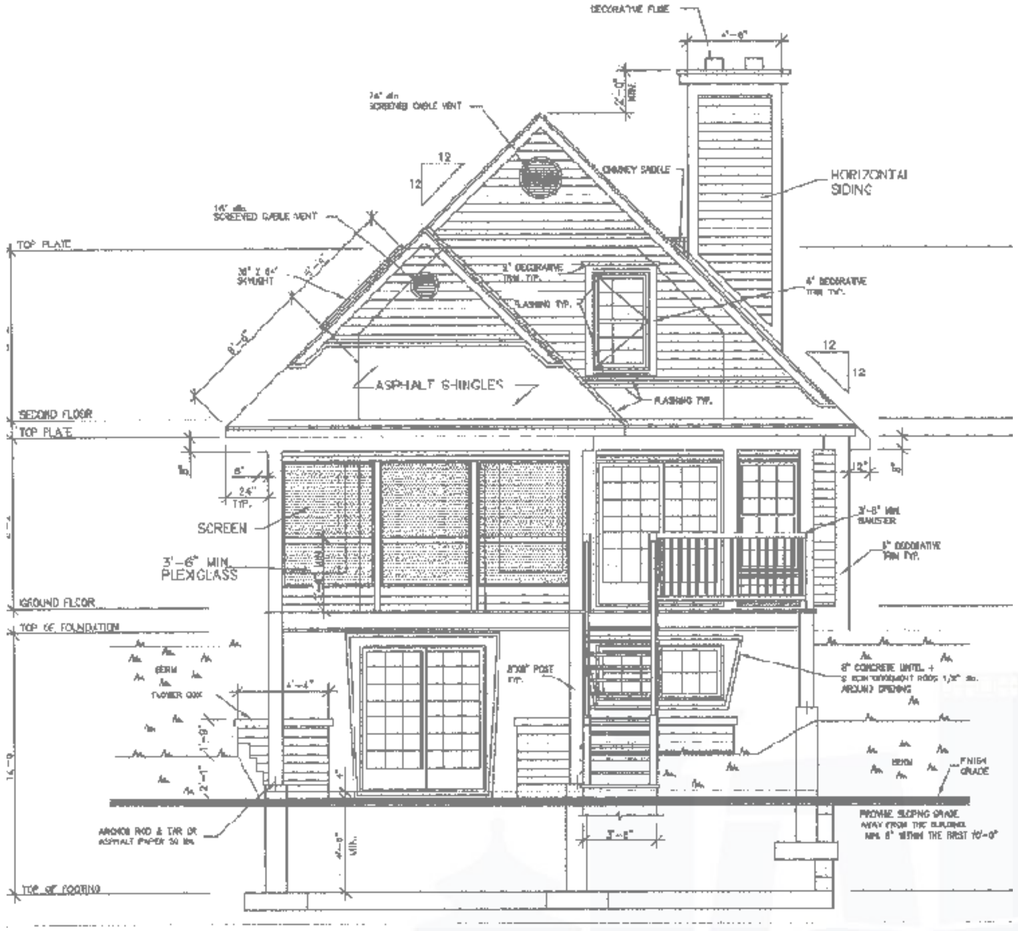
技术全景


业务层
支撑层
业务交付
iOS
Android
Web
体验提升
高性能组件
性能优化
交互优化
架构支撑
存储
网络
组件
通信
动态化框架
全流程
开发
应用架构
遥测平台
组件平台与可视化
测试
测试方法
测试工具
发布
热部署平台
发布流程
线上
稳定性监控
性能监控

应用架构


1 NO METAL CONTACT WITH IPhone 5 METAL.
2 DO NOT OBSTRUCT THE ACOUSTIC OPENINGS: FRONT MIC, REAR MIC, EARPIECE, AND SPEAKER.
3 DO NOT OBSTRUCT THE IMAGING FEATURES: FRONT CAMERA, REAR CAMERA, REAR FLASH.
4 DO NOT OBSTRUCT THE PROXIMITY SENSOR OR ALS (AMBIENT LIGHT SENSOR).
SLEEP/WAKE BUTTON
RINGER SWITCH
VOLUME BUTTON
EARPIECE
FRONT CAMERA
FRONT MIC
RECEIVER
REAR MIC
REAR FLASH
REAR CAMERA
SIM TRAY
HOME BUTTON
DO NOT OBSTRUCT THIS AREA WITH METAL
SPEAKER
IPHONE 5

应用架构

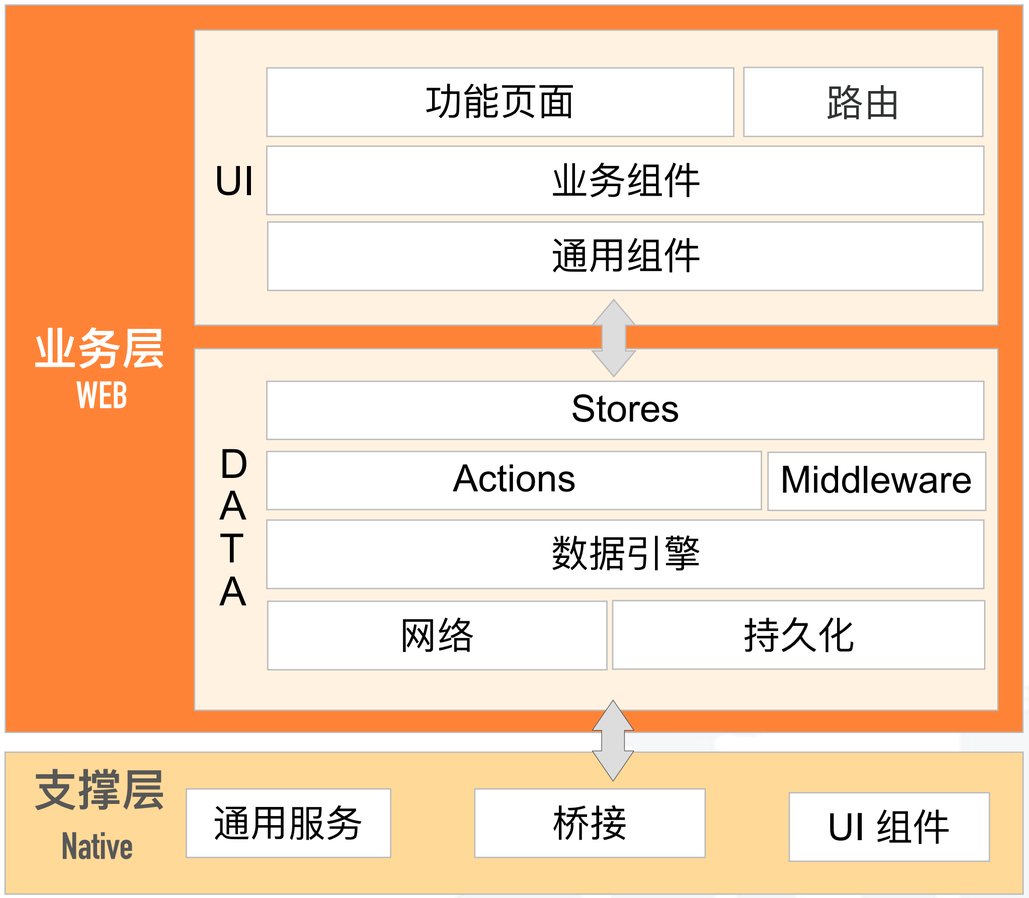


职责边界

协作效率

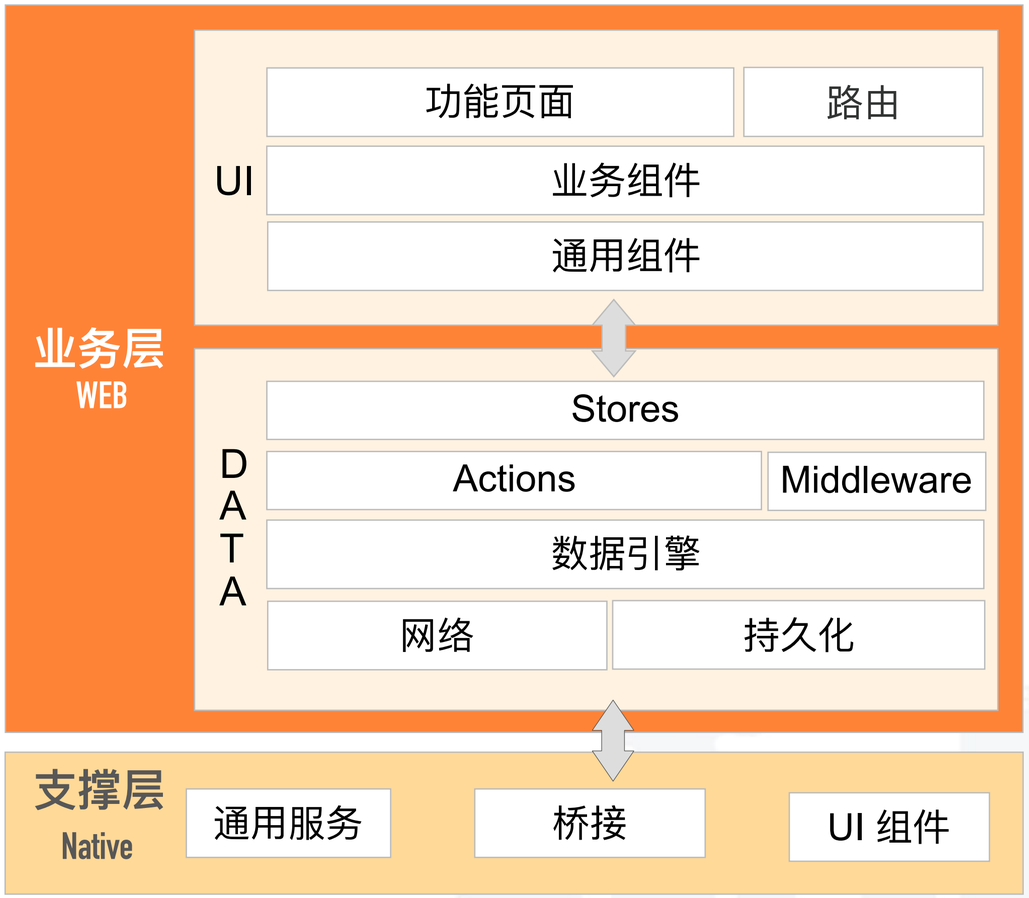
FE vs 客户端

应用架构



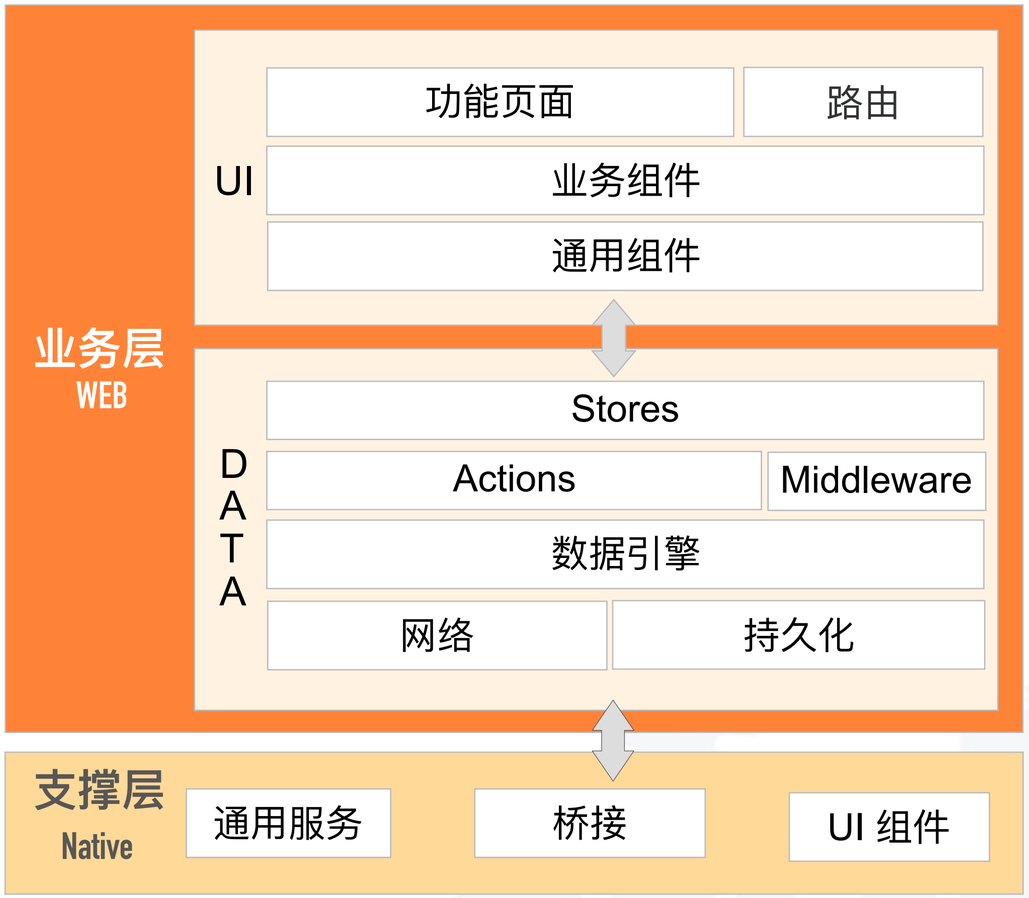
- 整体特点
  - 前端上浮
  - NA 端下沉
  - 业务层充分动态化

应用架构



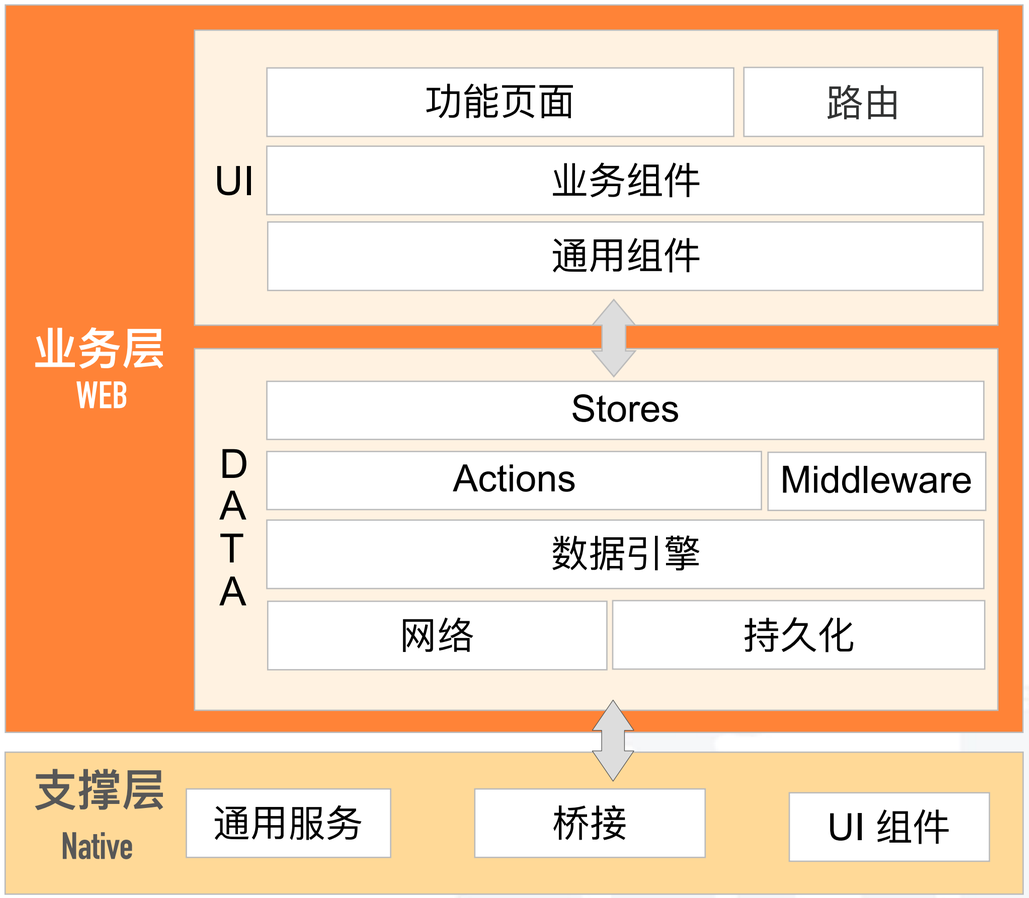
- 前端架构
  - 分层结构：UI 与 数据
  - UI 分层：通用组件 -> 业务组件 -> 页面 :: 路由
  - 数据分层：数据引擎无关，持久化对业务透明

应用架构



- 前端技术栈
  - 逻辑实现：TypeScript + React
  - 样式定义：SASS
  - 页面路由：自研 Navigation(Native)
  - 数据逻辑：数据适配层 + Redux / Mobx
  - 数据持久化：AsyncStorage

应用架构



代码量
50000+

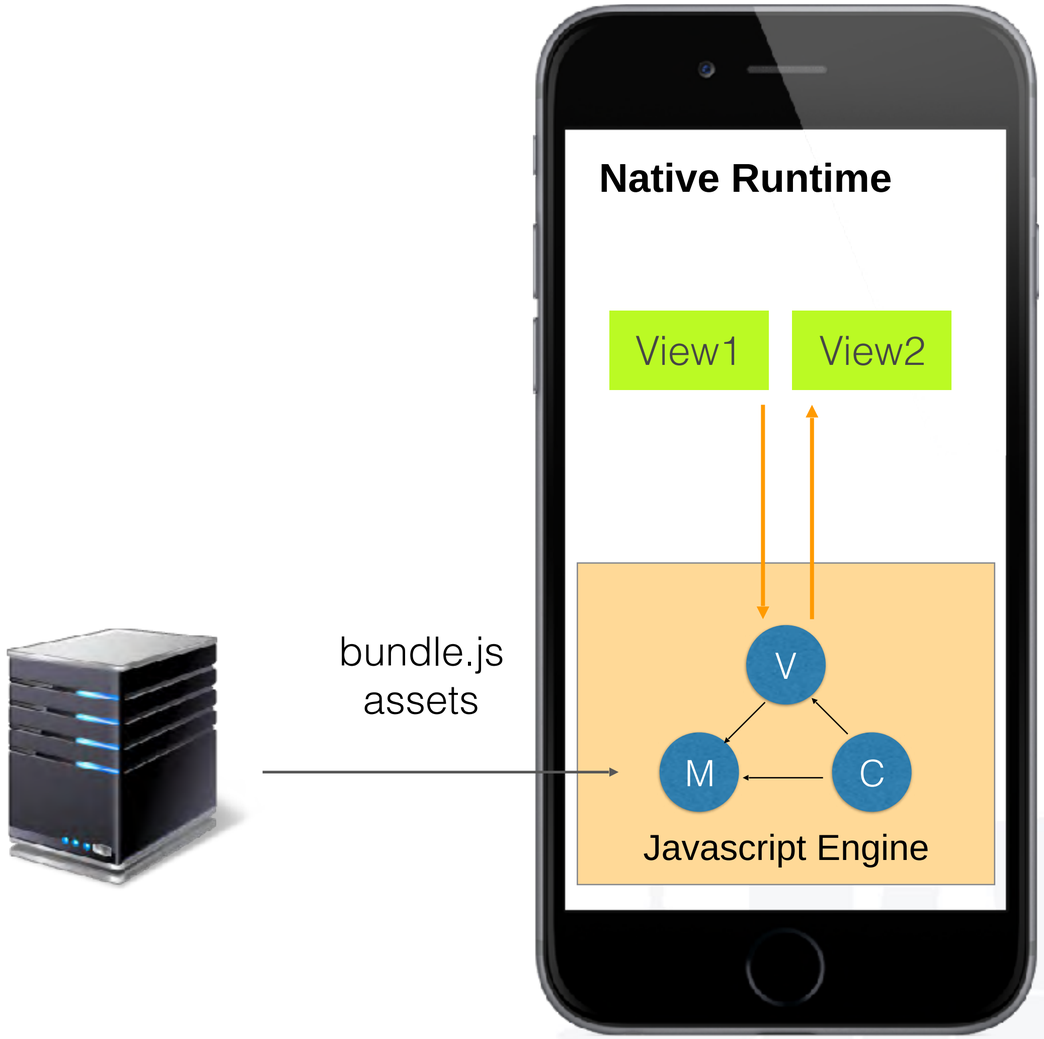
组件数
60+

页面数
30+

双端复用
95%+

性能

性能瓶颈



JavaScript 引擎单线程 - 交互卡顿

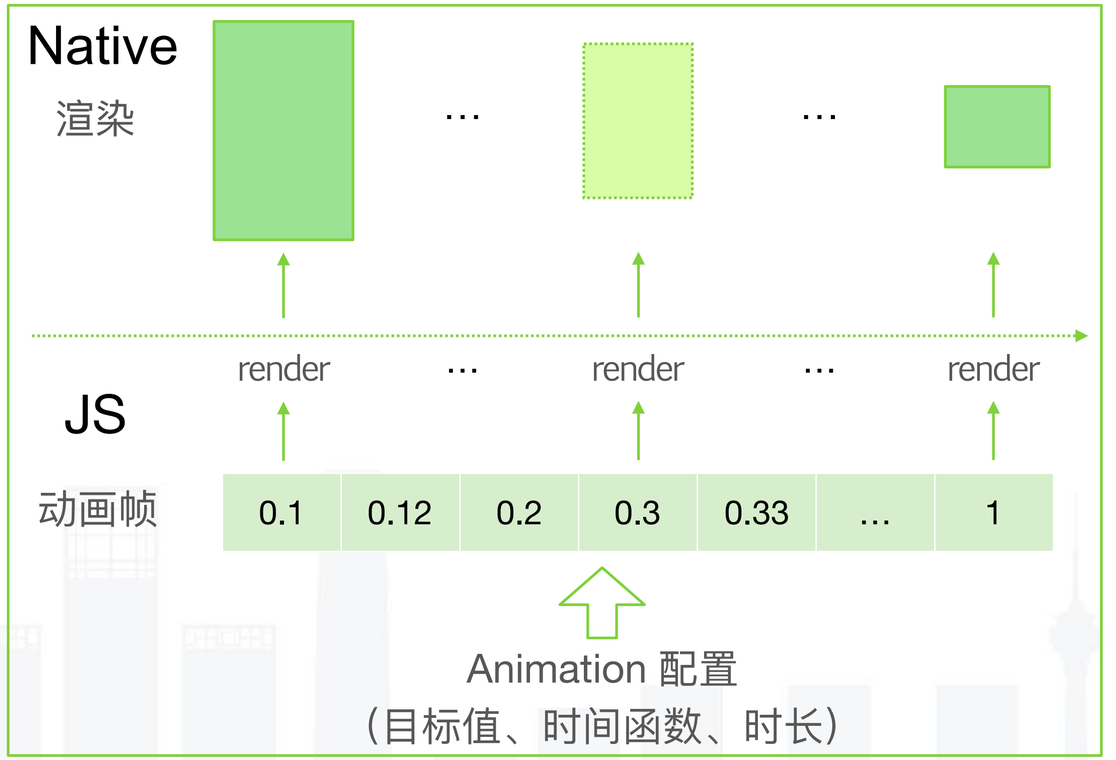
内存优化存在缺陷 - 内存消耗大

高性能组件

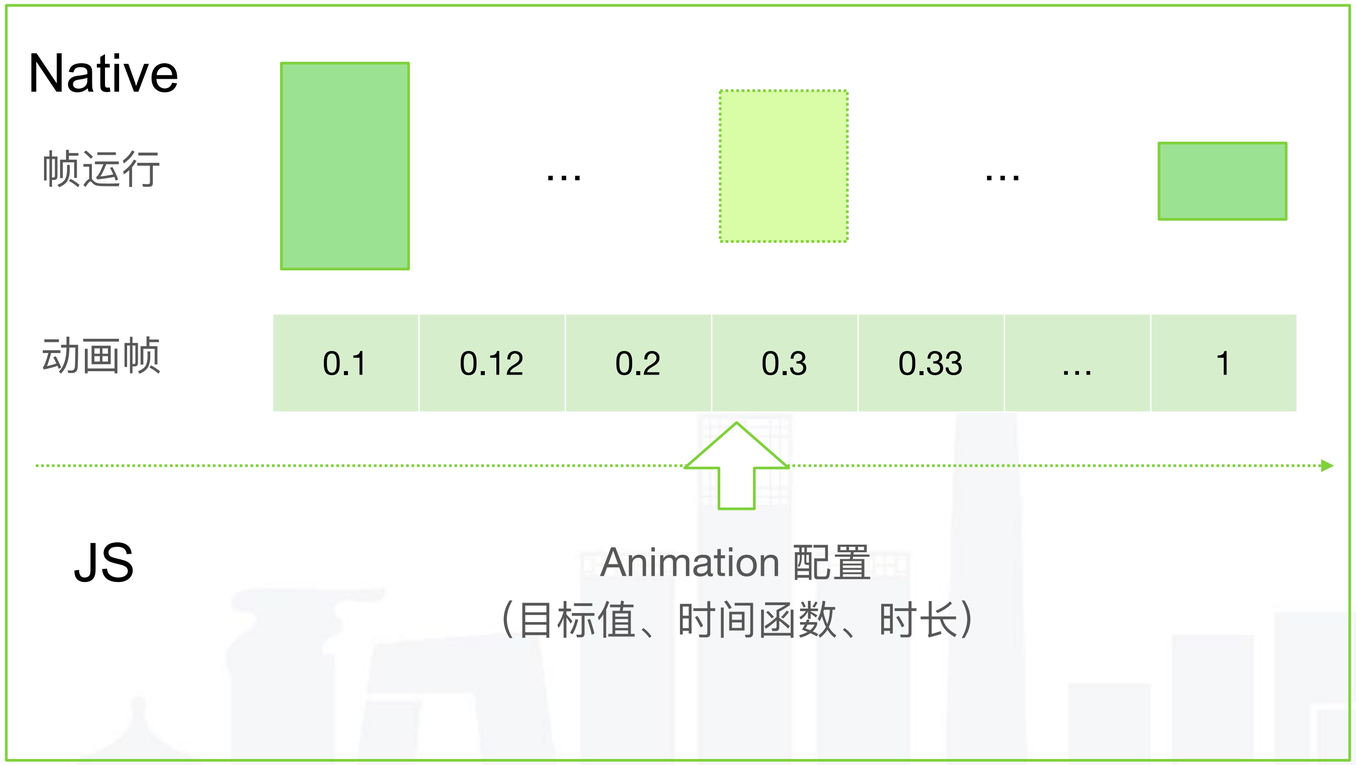
Animation	高性能动画组件，解决复杂动画、交互卡顿问题
Navigation	高性能切页组件，提升页面切换体验
Listview	列表组件性能优化，解决内存以及部分性能问题

高性能组件 - Animation

- Animation 问题
  - JS 计算动画帧
  - JS 单线程性能瓶颈
  - 5 ~ 15 FPS
  - 严重影响用户体验



高性能组件 - Animation



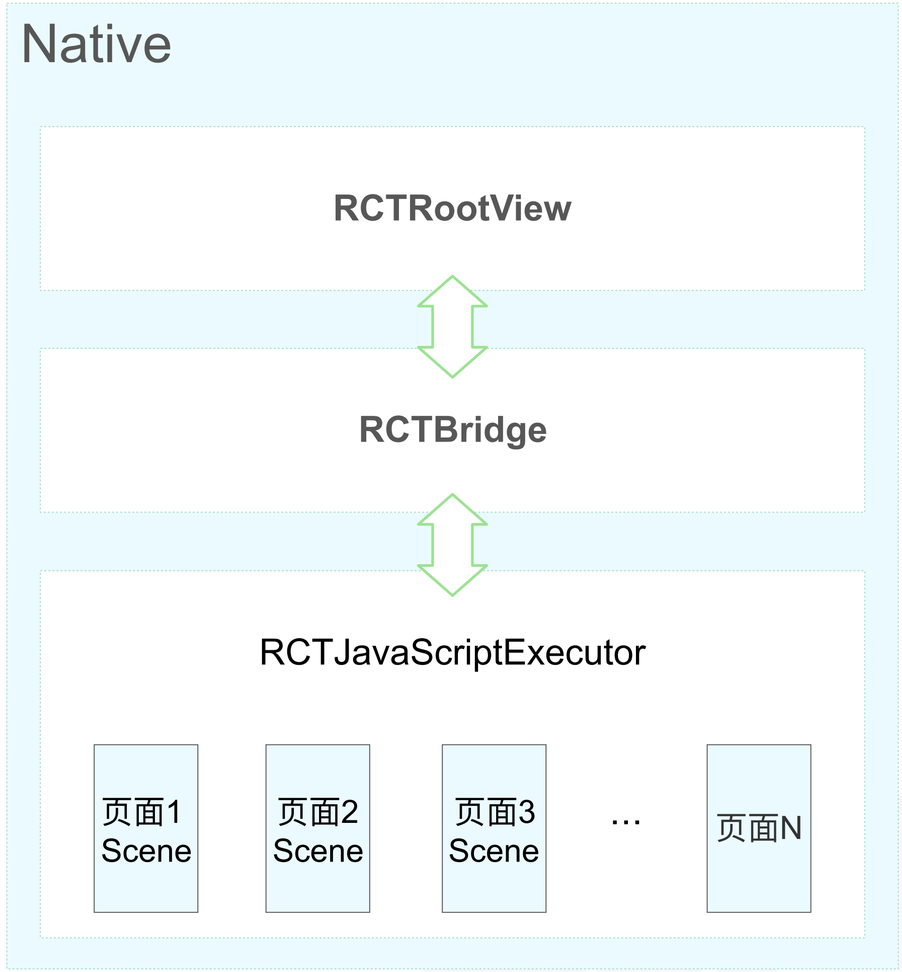
高性能组件 - Animation

```
34          return (
35              <View style={styles.container}>
36                  <Animation.AnimationView
37                      style={[styles.anim, { left, top }]}
38                      autoplay autoclear
39                      data={[{
40                          type: 'Scale',
41                          from: 1,
42                          to: 30,
43                          from2: 1,
44                          to2: 30,
45                          duration: 400
46                      }, {
47                          type: 'Alpha',
48                          from: 1,
49                          to: 0,
50                          duration: 400
51                      }]}
52                      onEnd={this.handleAnimationEnd}
53                      ><View style={styles.wave} />
54                  </Animation.AnimationView>
55              </View>
56          )
```

高性能组件 - Animation

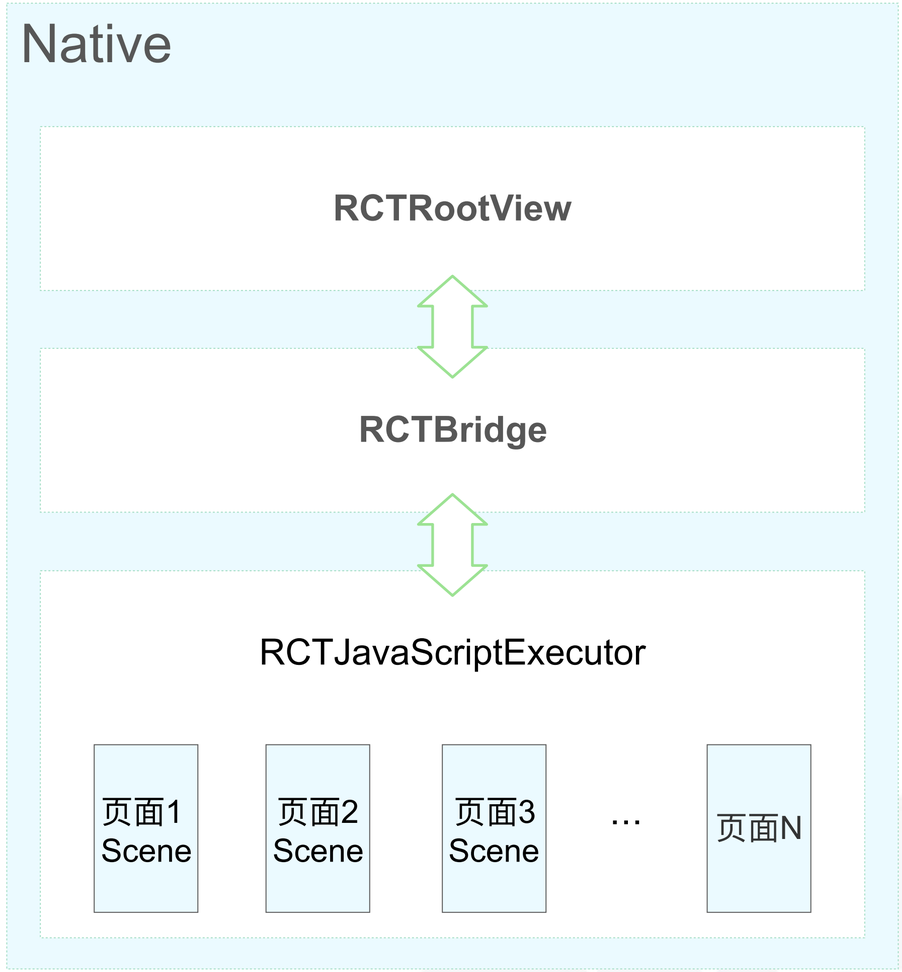
- Animation 效果
  - 复杂场景下 FPS 由 15 FPS 提升至 50+FPS
  - 90%+ 动画效果采用自研库实现
  - 大幅提升产品整体体验

高性能组件 - Navigation



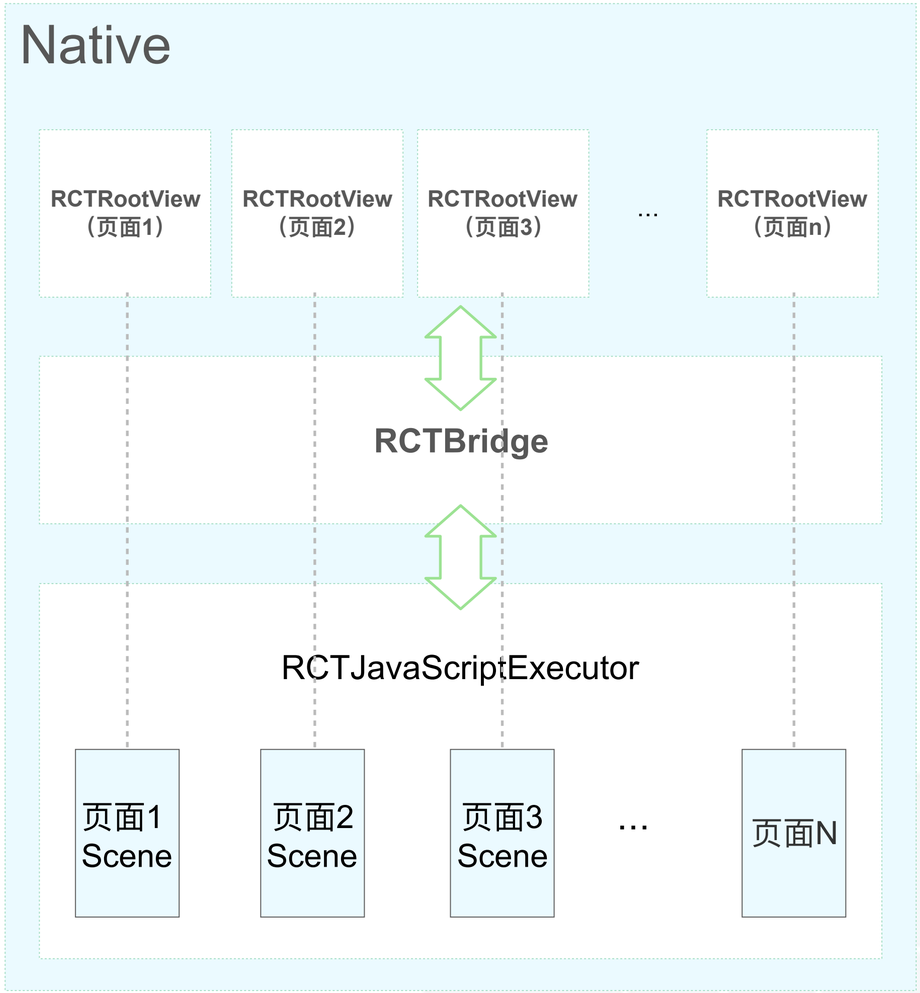

```
require('./home/home.component')
require('./comment/comment.scene')
require('./home/components/search/search.compo
require('./home/components/search/components/s
require('./common/web-view-page/index')
require('./user/user.scene')
require('./user/setting/setting.scene')
require('./user/setting/message-setting/messag
require('./user/about/about.scene')
require('./user/activity/activity.scene')
require('./user/edit-information/edit-informat
require('./comment/comment.scene')
require('./user/setting/setting.scene')
require('./message/message-collection/collecti
require('./message/message-zan/zan.scene')
require('./message/message-remindupdate/remind
require('./message/message-notify/notify.scene
require('./message/comment/comment.scene')
```

高性能组件 - Navigation



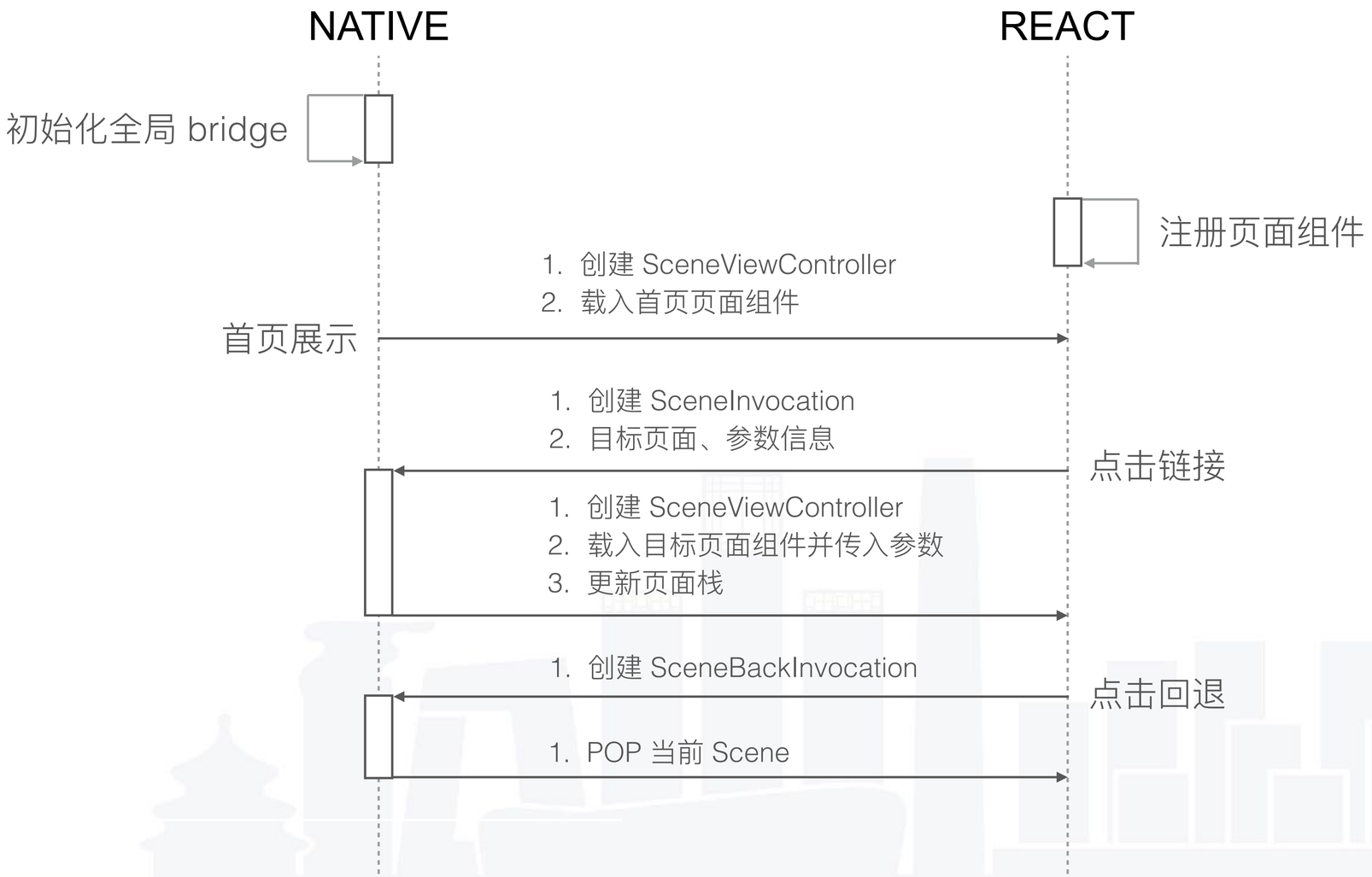
- Navigation 问题
  - JS 端模拟页面切换
  - 自带动画性能瓶颈
  - 自研动画不适用
  - 与原生客户端体验差距巨大

高性能组件 - Navigation




```
52  Object.keys(scenes).forEach(key => {
53      const scene = scenes[key]
54      AppRegistry.registerComponent(key, getSimpleScene(
55          scene.type, onlyStore
56      ))
57  })
```

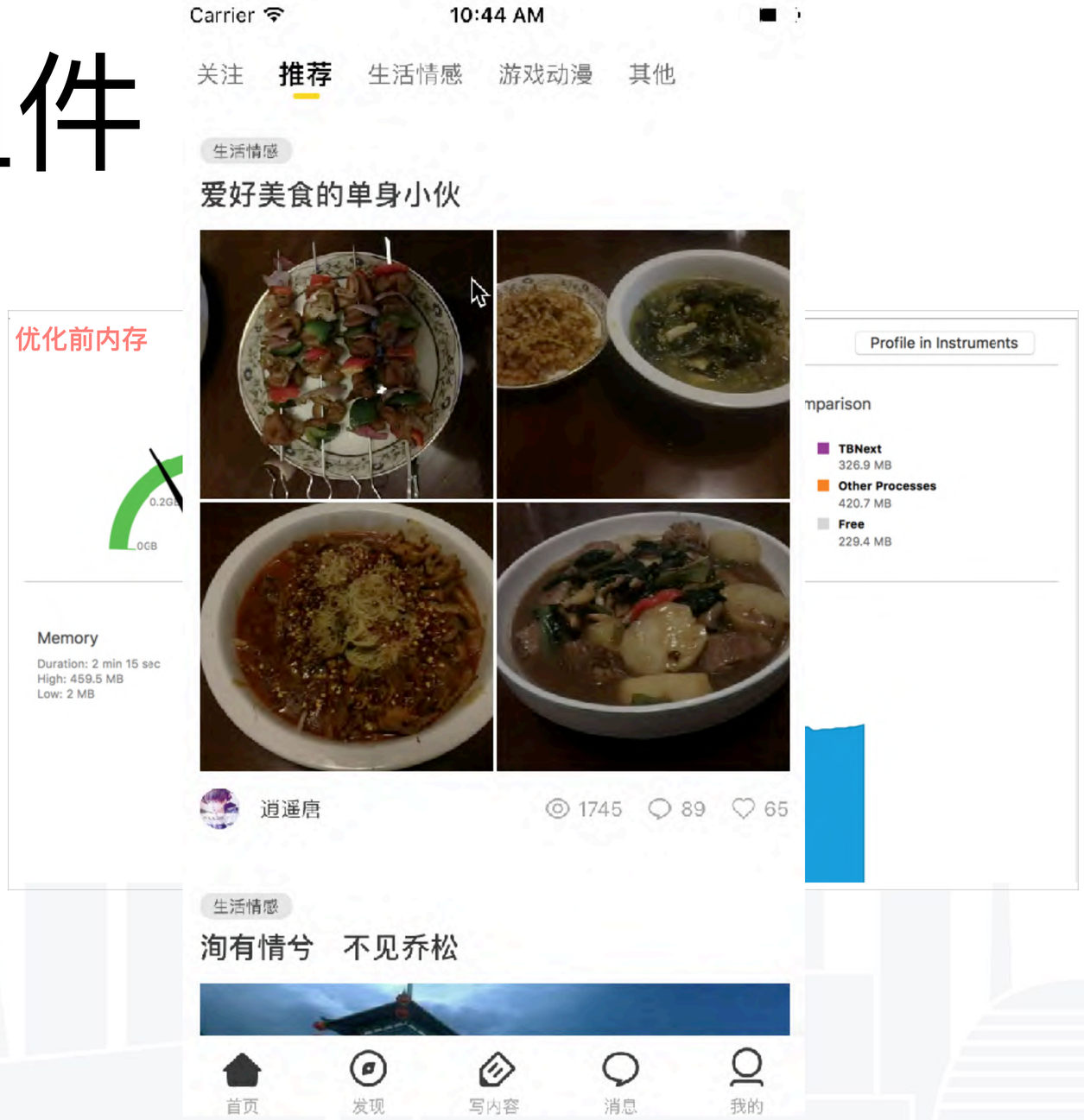
高性能组件 - Navigation



高性能组件 - Navigation

高性能组件

- Listview 问题
  - 列表无视图复用机制
  - 内存占用与 Listview 滑动距离成正比
  - 图片无内存控制机制
  - 下拉刷新卡顿明显

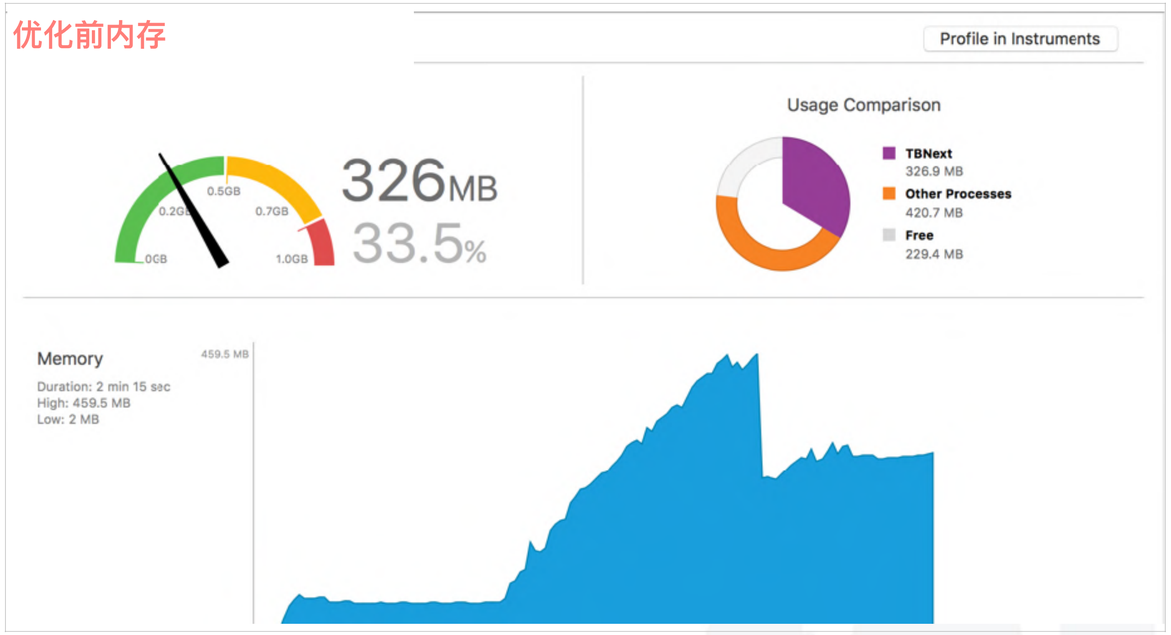


高性能组件 - Listview

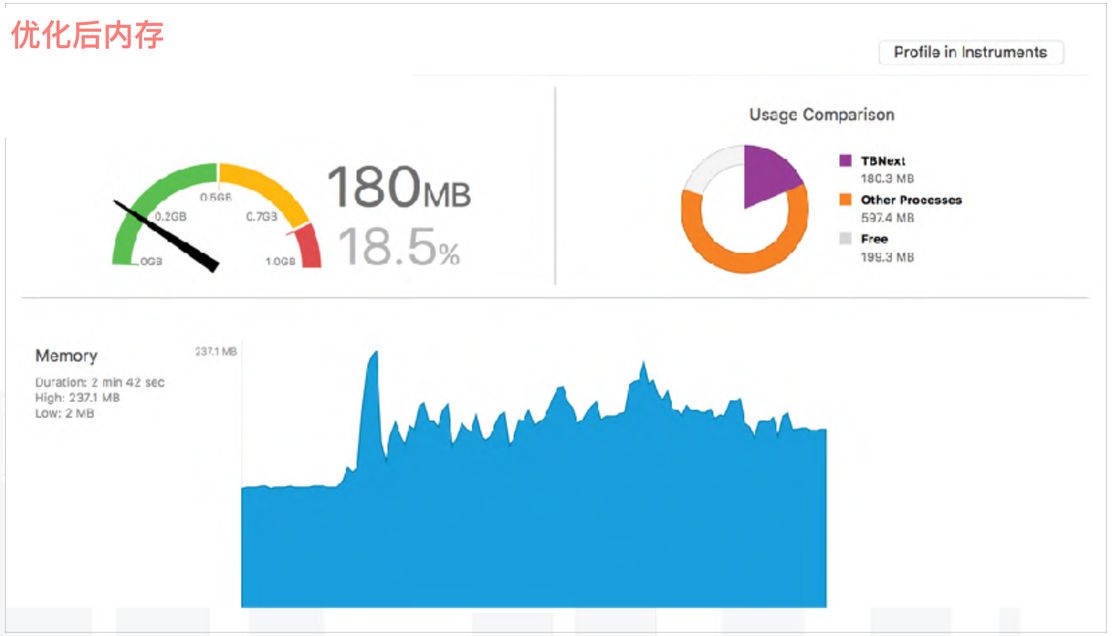
- 解决方案
  - 清空可视区外的单元容器，减少无谓的内存占用 （上滑）
  - 清除底部不可见单元，减少整体渲染数量（下拉刷新）

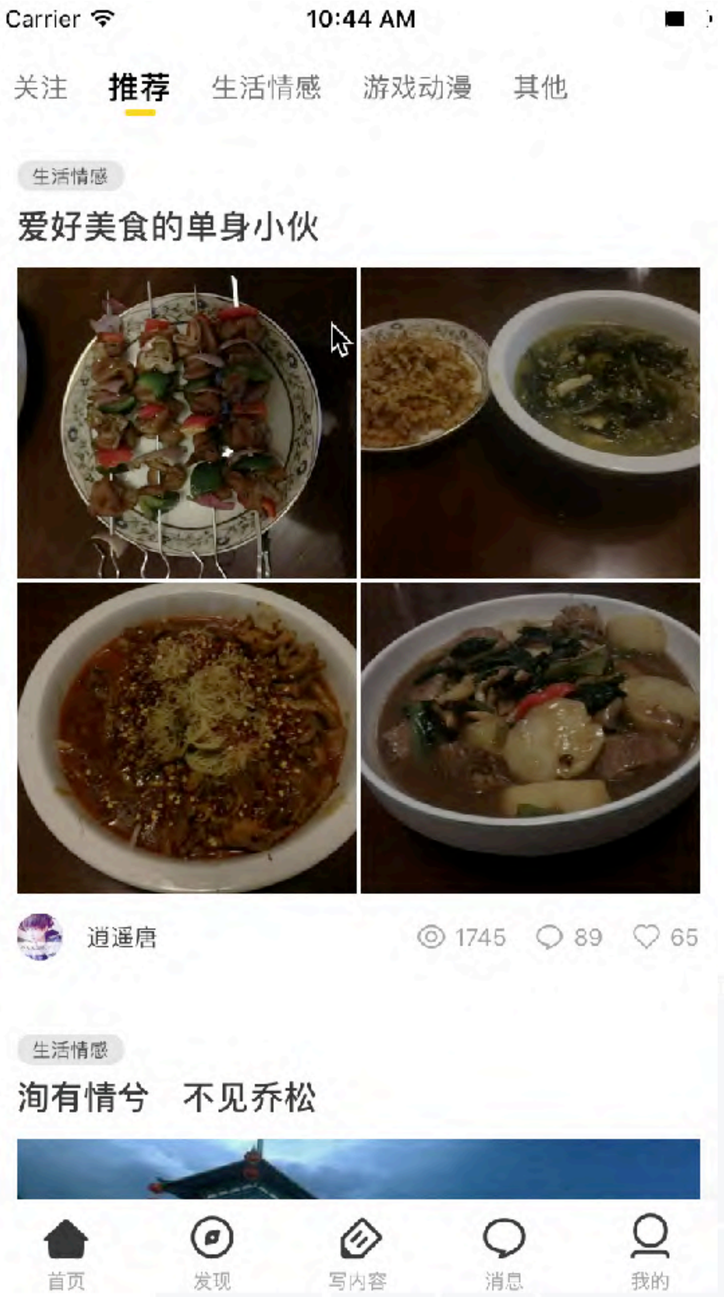
高性能组件 - Listview

优化前内存



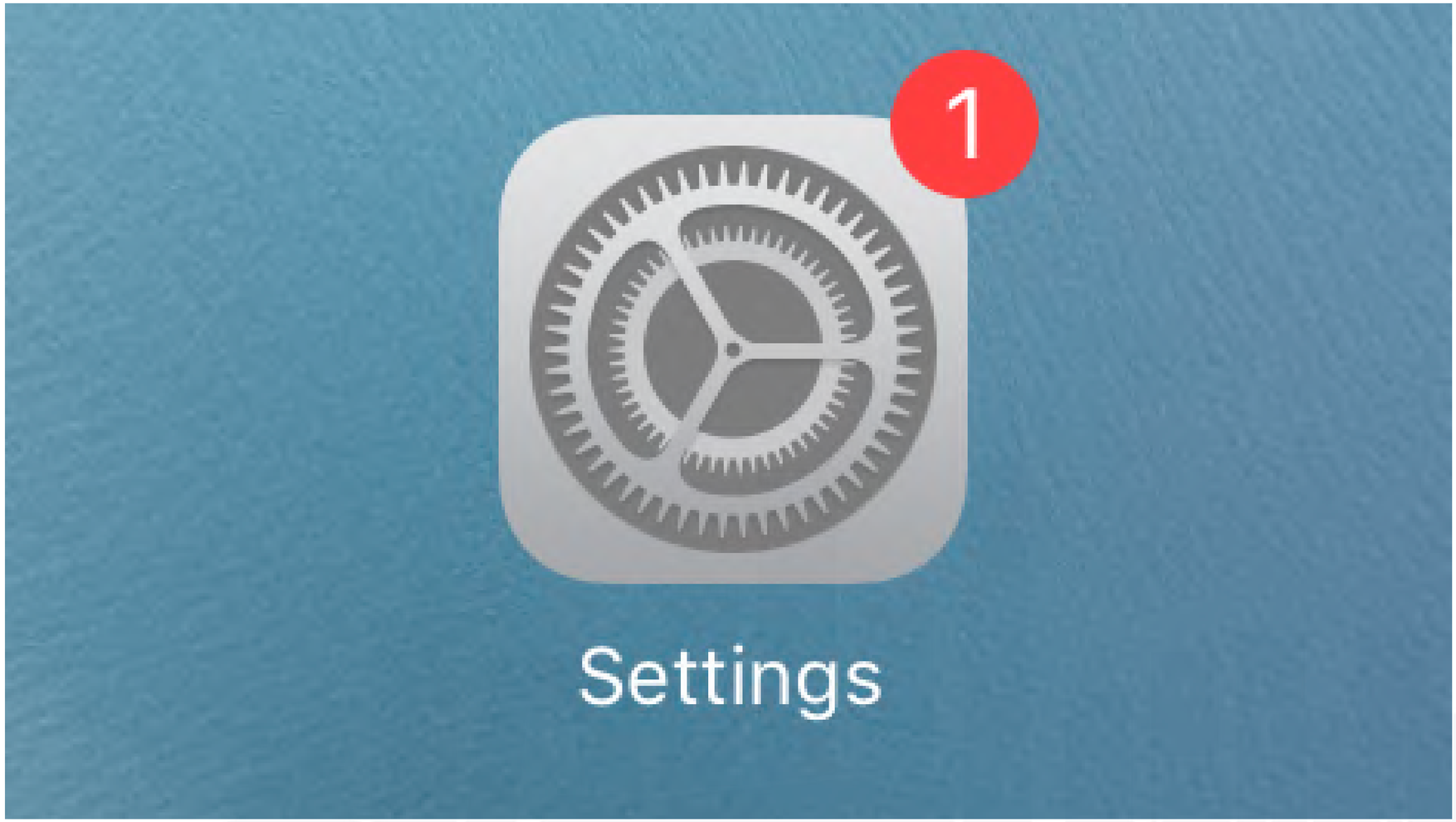
优化后内存



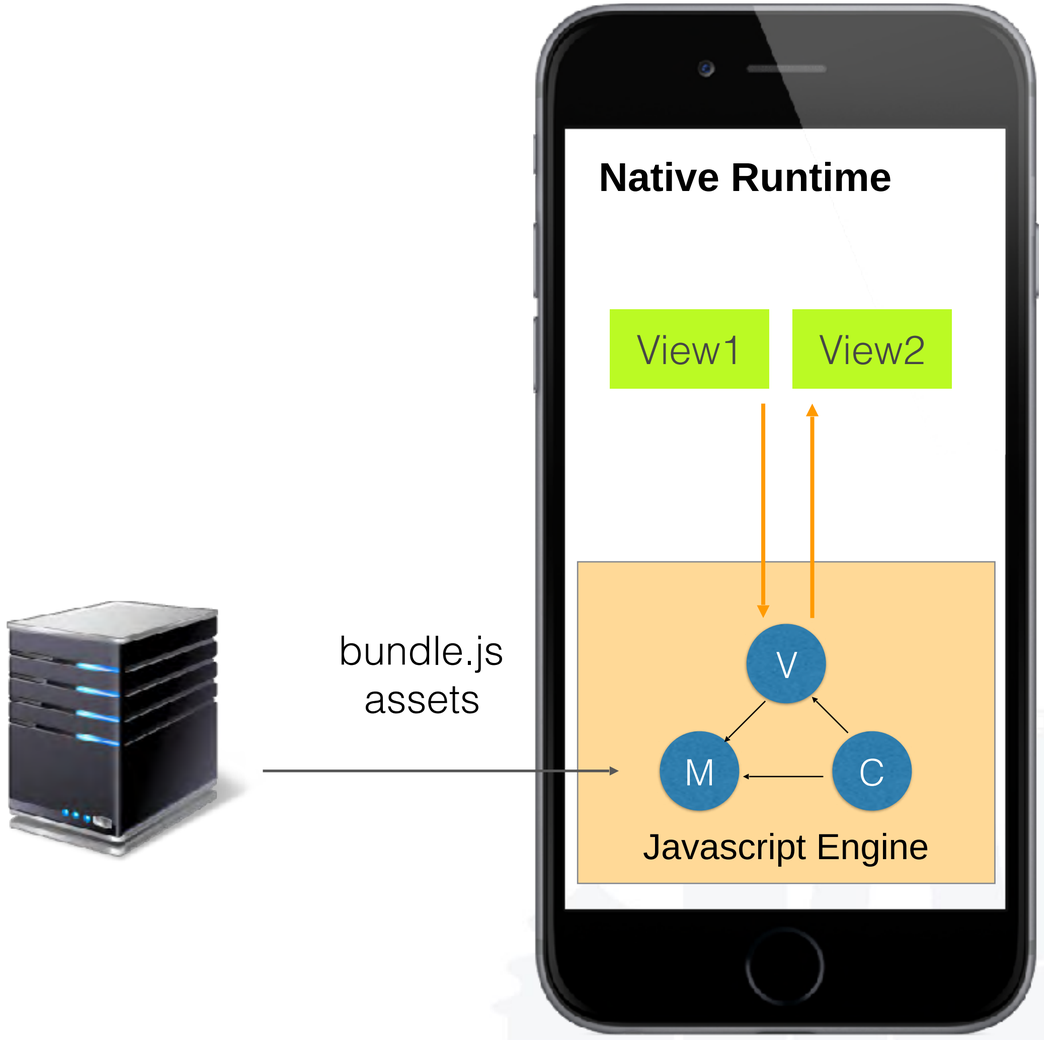

Carrier
10:44 AM
关注
推荐
生活情感
游戏动漫
其他
生活情感
爱好美食的单身小伙
逍遥唐
1745
89
65
生活情感
洵有情兮　不见乔松
首页
发现
写内容
消息
我的


Carrier
11:11 AM
关注
推荐
生活情感
游戏动漫
其他
晴窦儿
2
1
1
生活情感
敢进来我就敢跟你表白
帅
3
4
4
请下拉刷新查看更新
首页
发现
写内容
消息
我的

热部署



热部署



- 需求&特性
  - 多版本管理
  - 多级发布（灰度、回滚）
  - 安全性
  - 稳定性
  - 时效性

热部署

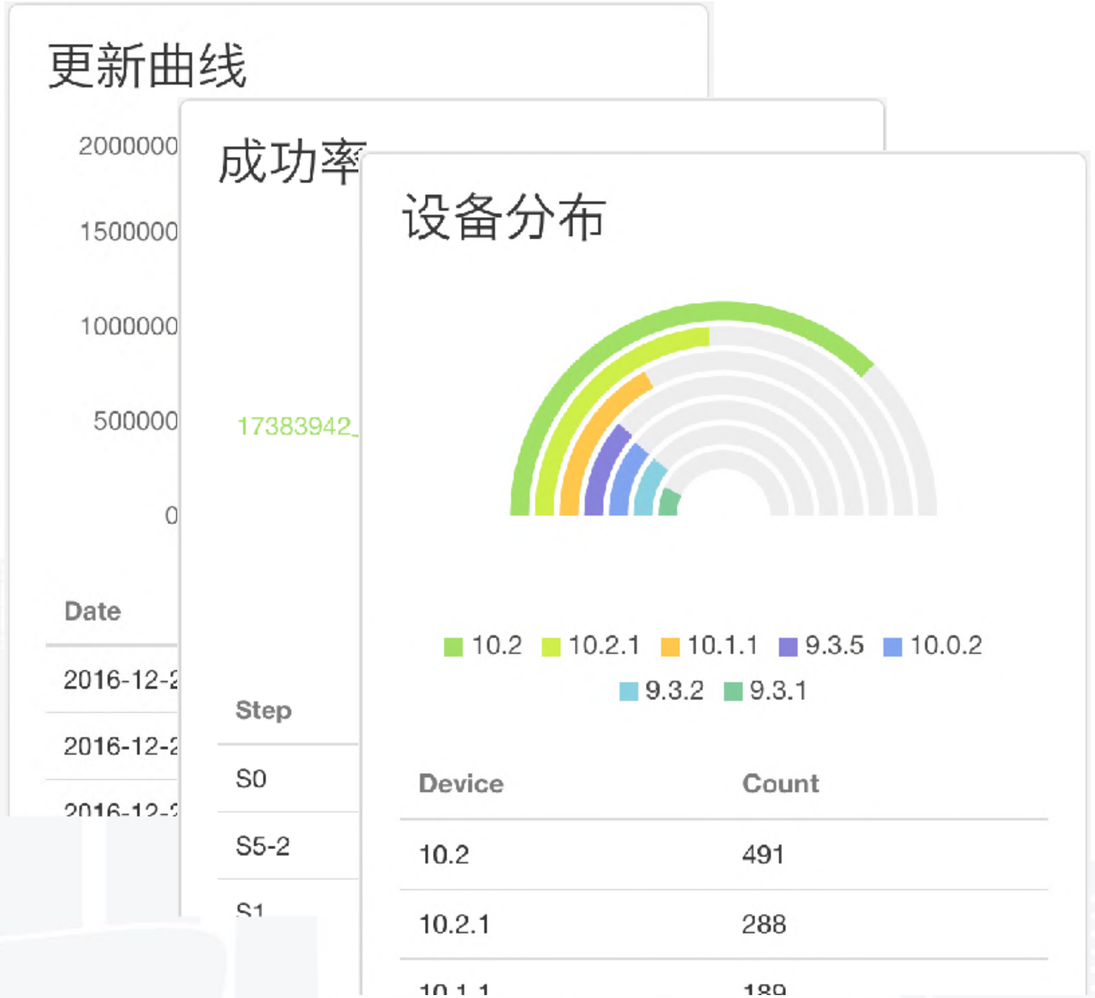
- 时效性
  - 新用户新版本到达率
  - 新版本首日覆盖率

- 问题与挑战
  - 应用启动时间
  - 网络环境
  - 终端性能

- 解决方案
  - 全量更新 -> **增量更新** + 网络环境识别 + 启动时自动升级
  - 更新策略精细化控制
    * 新页面新版本
    * 后台唤起自动升级

热部署


next updater
wefan
安卓正式
版本管理
数据报表
包设置
基本信息
待发布包
提交时间
提交人
提交时间
提交人
已发布版本
版本号
状态
操作
2.0.8.0
released
回滚
2.0.4.0
released
2.0.1.0
released
2.0.0.0
released
1.1.3.0
pre-release
灰度
版本号
状态
操作


更新曲线
2000000
1500000
1000000
500000
0
Date
2016-12-2
2016-12-2
成功率
17383942_
Step
S0
S5-2
S1
设备分布
10.2
10.2.1
10.1.1
9.3.5
10.0.2
9.3.2
9.3.1
Device
Count
10.2
491
10.2.1
288
10.1.1
189

可视化构建

可视化构建

- 运营需求的高频且低性价比
- WEB 研发中的可视化构建
- 兼容 Android、iOS、Web

可视化构建

- 基于 React Native 的可视化构建
- 复用业务组件库
- 灵活的数据源
- 跨平台支持

可视化构建


NEXT
在线设计平台
浏览
工作台
全局设置
预览
保存
发布
组件
版本
资源
爸爸去哪儿
PK
亲生爸爸
谁更吐一筹?
实习爸爸
爸爸去哪儿
芒果TV《爸爸去哪儿第四季》每周五晚20点暖心上线！本周将前往军营，上演萌版真正男子汉。明星爸爸们带着宝宝为部队里的驻边战士们献上了精彩表演，萌心蕾力父女首出裂缝。
实时讨论 24.5w 个内容
查看全部热议榜单
精选贴
请叫我金偶吧
沙溢安吉绝对搞笑担当～
刚刚看完第一期～ 满口大渣子味齁死我了
布局
布局-绝对定位
滚动布局
文字
输入框
开关
通用
定制
模板
实例数:130

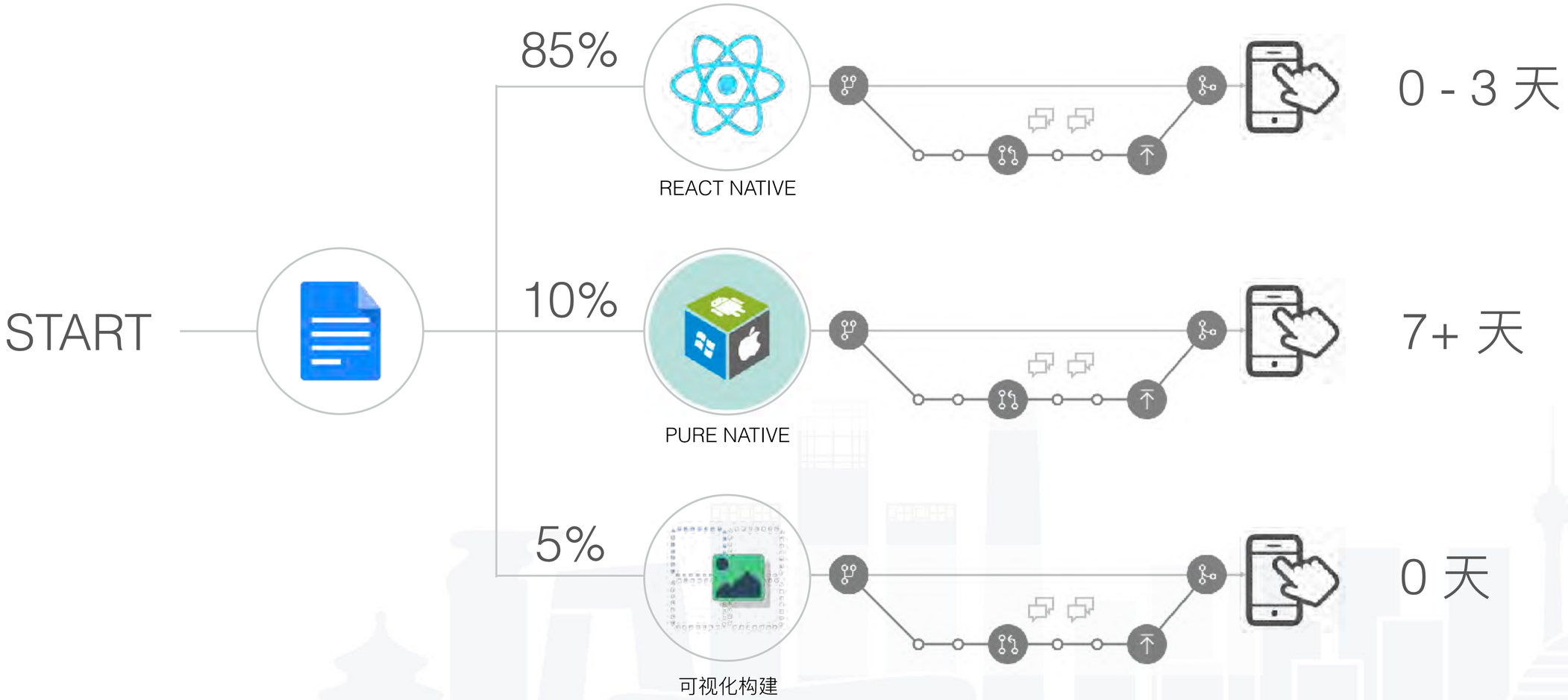

关注
推荐
生活情感
游戏动漫
其他
记录春节每一天
让全世界看见你春节模样
生活情感
一个喜欢画画的二次元宅腐女的春节日常
九南sama
生活情感
~o(〃'▽'〃)o我们家小松松

新模式下的协作

新模式下的协作

	传统模式	新模式
需求	以版本为迭代单位，众多功能点打包发布	以功能点为迭代单位，多个功能可以并行发布
开发	多功能点并行开发，合并到目标版本统一提测	多功能点并行开发，并行测试
测试	以版本为测试单位，需要对应用完整功能进行回归	以功能点为测试单位，回归压力增加
发布	多功能点统一发布	多功能点并行发布

新模式下的协作



现状

90%

动态化比例

Hour
Day

迭代周期

95%

双端复用率

40%

成本降低

关注QCon微信公众号，
获得更多干货！

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方 Geekbang. 极客邦科技 InfoQ