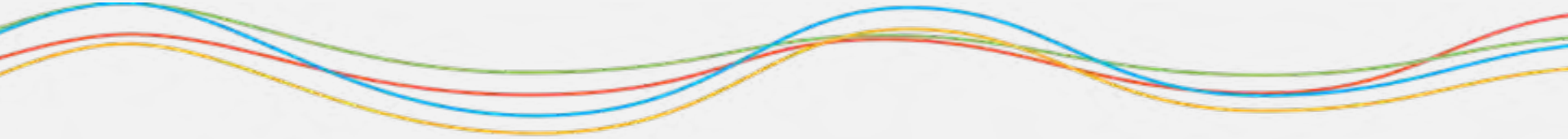
APMCon

H5与原生体验的融合
蘑菇街混合开发平台的优化实践

美丽联合集团—王兴楠（赤木）



About Me

上海交通大学硕士毕业后，曾任职于上海Intel，从事多年浏览器内核与Web引擎研发工作；
混迹于开源社区，Chromium开源项目Committer，WebKit开源项目Committer；

Crosswalk项目早期核心开发者；

现为美丽联合集团技术专家，负责混合开发体系的建设，目前专注于基于动态跨平台技术的下一代移动开发体系。

Agenda

APMCon

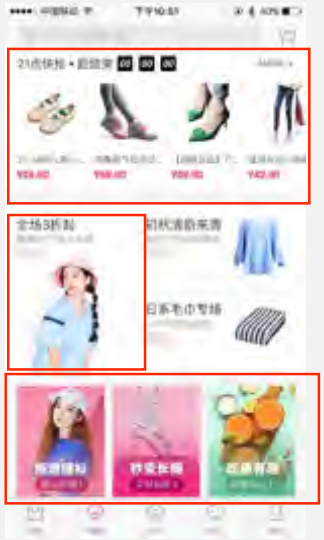
- 混合开发平台介绍 （Hybrid Development Platform）

- HDP的优化实践：
  - 基于独立内核的体验优化，秒杀WebView
  - 动态跨平台框架的研发与应用，让H5的体验与原始无异
  - 离线化体系的建设，实现H5页面的“秒开”

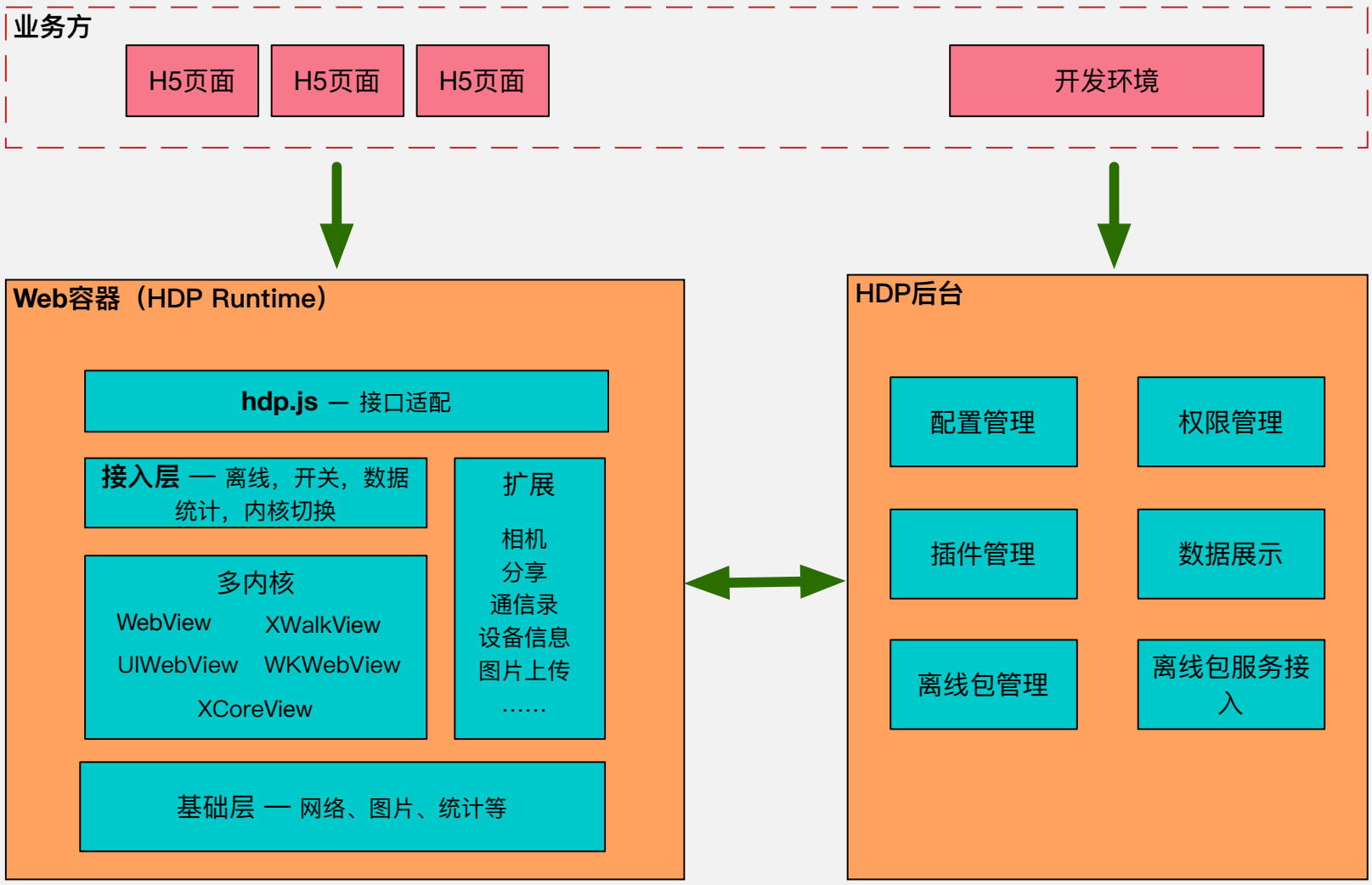
- 未来的展望与计划

meili
美丽联合集团

MOGU



HDP的结构


业务方
H5页面
H5页面
H5页面
开发环境
Web容器（HDP Runtime）
hdp.js — 接口适配
接入层 — 离线，开关，数据统计，内核切换
多内核
WebView
XWalkView
UIWebView
WKWebView
XCoreView
扩展
相机
分享
通信录
设备信息
图片上传
……
基础层 — 网络、图片、统计等
HDP后台
配置管理
权限管理
插件管理
数据展示
离线包管理
离线包服务接入

Android系统WebView的问题

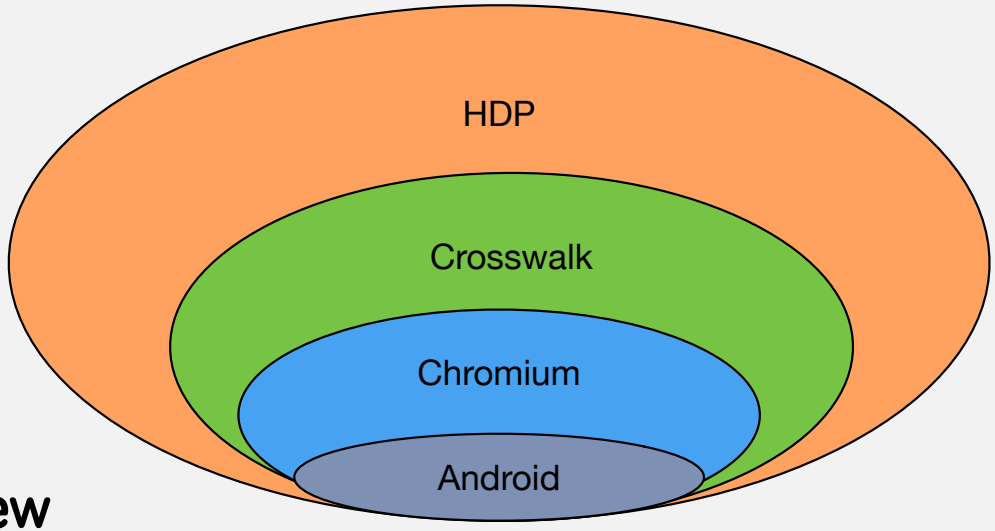
兼容性

性能

安全性

功能

无法修复的bug



集成独立的内核替代系统WebView

Crosswalk项目

二次开发

基础独立内核的问题

包大小：完整包打进去在20M以上

稳定性问题：奇葩平台的兼容性

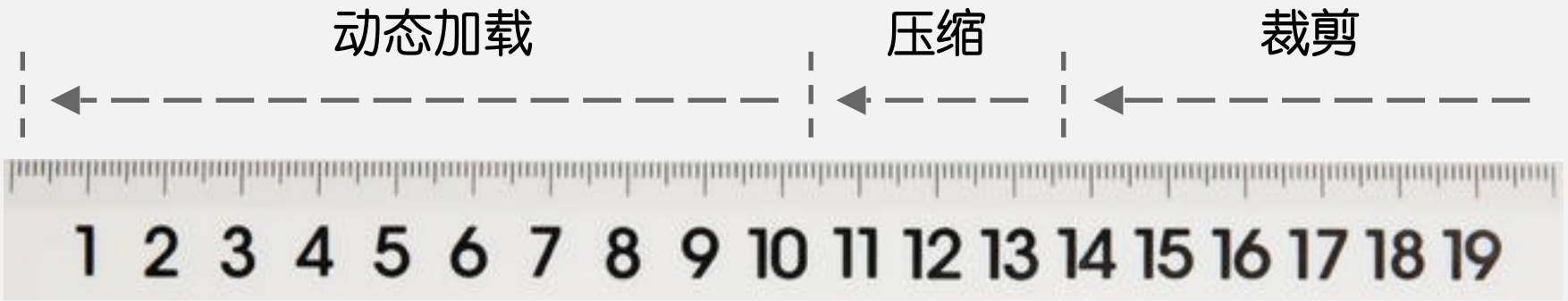
资源消耗

解决方案

减包：替换/裁剪，压缩，动态加载

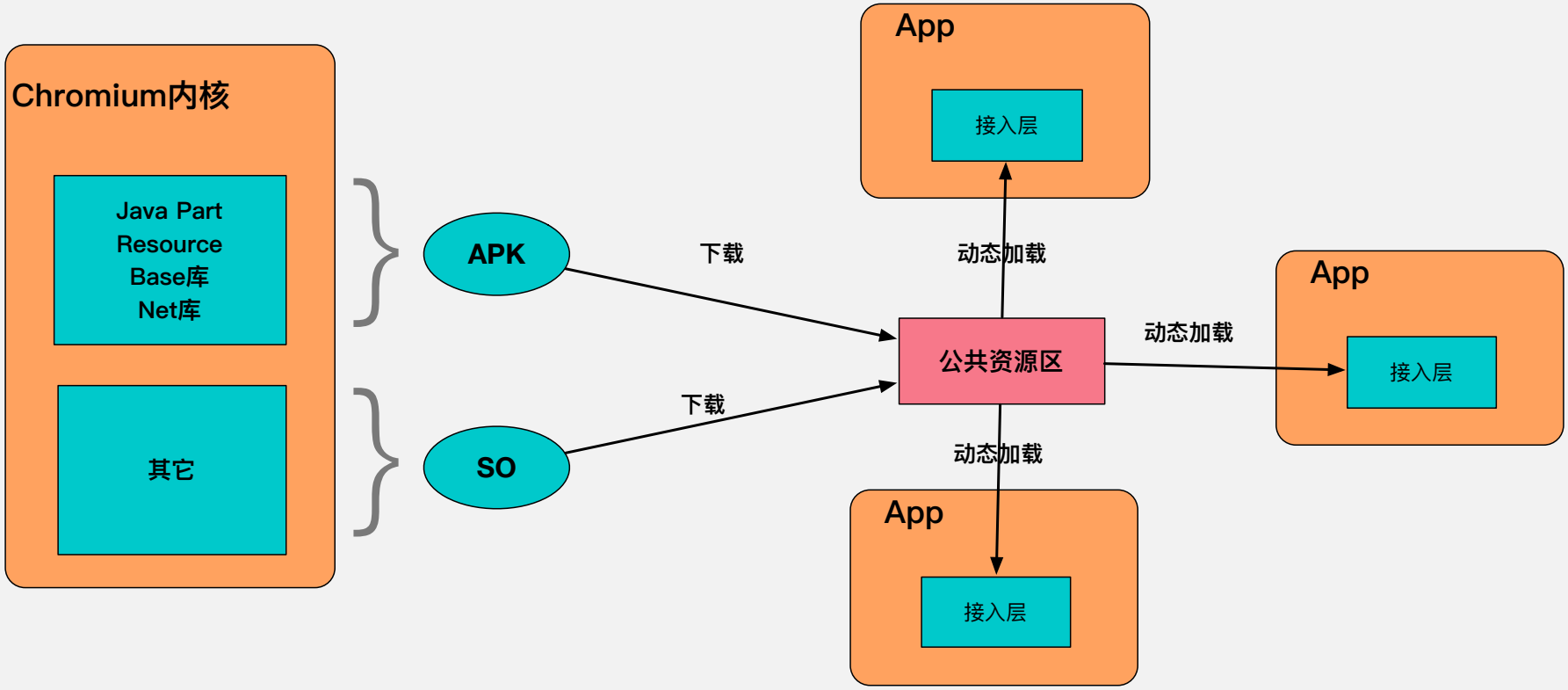
管控：细粒度的黑白名单，系统WebView回退，动态切换

限制资源：控制页面量级，设置使用阈值

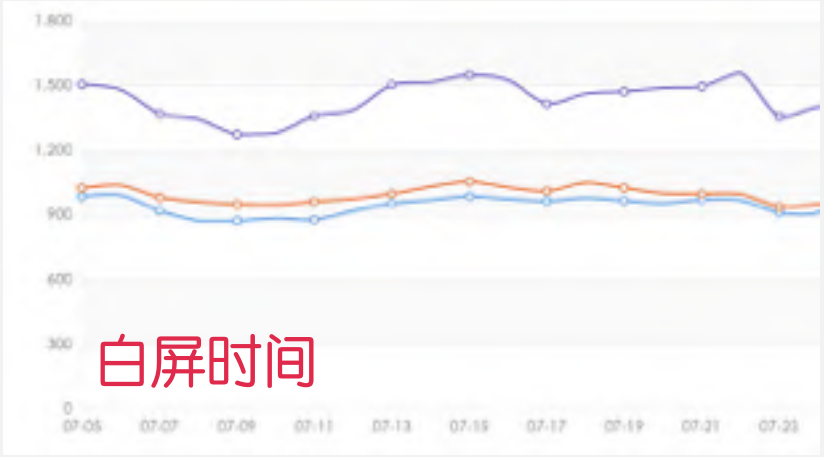


APMCon

安卓包的动态加载



性能数据



XWalkView　　WebView（4.4以上）　　WebView（4.4以下）

Crosswalk VS X5

设备	线上活动页（简单H5页面）			321大促活动页（较复杂H5页面）			会员签到页（复杂的H5页面）		
（中端设备）Nexus 5	XWalkWebView	X5WebView	对比	XWalkWebView	X5WebView	对比	XWalkWebView	X5WebView	对比
可交互时间（ms）	196	308	+36%	118	130	+9%	89	239	+62%
加载完成时间（ms）	248	325	+23%	137	152	+9%	197	240	+17%
WebView初始化时间（ms）	62	116	+46%	80	107	+25%	66	114	+42%
冷启动占用时内存M)	206.6	155	-33.2%	239.7	183.1	-30.9%	188	171.2	-9.8%

设备	线上活动页（简单H5页面）			321大促活动页（复杂H5页面）			会员签到页（复杂的H5页面）		
（低端设备）联想乐檬K1	XWalkWebView	X5WebView	对比	XWalkWebView	X5WebView	对比	XWalkWebView	X5WebView	对比
可交互时间（ms）	228	276	+17%	125	139	+10%	133	215	+38%
加载完成时间（ms）	289	288	0	141	155	+9%	166	216	+23%
WebView初始化时间（ms）	85	97	+12%	90	112	+19%	92	121	+23%
冷启动占用内存(M)	128.4	115.3	-11.3%	158.9	114.2	-39.1%	124.8	115.1	-8.4%

WKWebView优势

APP内的内存使用量降低
js锁被取消，消除了一些功能限制
WKWebview的渲染能力有很大的提升
crash量有数量级的降低

优化实践 — 集成WKWebView

APMCon

iOS下不同容器内核的内存统计

webview个数/APP内存值	WKWebView	UIWebView
1张	91M	113.5M
2张	92M	132.5M
3张	93.1M	144.2M
4张	93.8M	155.1M
5张	94.5M	168.5M
备注	页面URL:http://act.mogujie.com/tghdchz?title=%E5%9B%A2%E8%B4%AD%E5%87%BA%E6%B8%B8%E6%B4%BB%E5%8A%A8 机型：iPhone6	

优化实践 — 集成WKWebView

APMCon

crash类型/APP版本	732	733	734
webcore WebCore::SecurityOriginHash::hash(WebCore::SecurityOrigin*) 机型 iOS9 99% iOS7 1%	323	344	99
JavaScriptCore bmalloc::Heap::allocateXLarge(std::__1::lock_guard<bmalloc::StaticMutex>&, unsigned long) 机型 iOS9 100%	25	68	1
JavaScriptCore bmalloc::VMHeap::grow() 机型 iOS9 100%	17	32	3
WebCore ___ZN7WebCore16DiskCacheMonitorC2ERKNS_15ResourceRequestENS_9SessionIDEPK20_CFCachedURLResponse_block_invoke1 机型 iOS9 100%	82	104	16
WebCore WebCore::DiskCacheMonitor::~DiskCacheMonitor() 机型 iOS9 100%	34	36	5
备注	UIWebView 100%	UIWebView 100%	IOS9 WKWebView 100%

集成WKWebView面临的问题

无法同步App中的Cookie

— 方案：通过JS注入

导航时无法禁止PageCache

— 方案：回退时强制刷新页面

JS重复注入有bug

— 方案：先清空后注入

页面关闭后内容不释放

— 方案：通过加载空字符串，主动清空页面

无法做网络拦截（URLProtocol）

— 目前无法解决，只能避免依赖的场景

背景

刚需

目前的移动开发的开发的两大痛点，动态性与跨平台开发的能力的缺失。

趋势

随着业界React Native、Weex、LuaView等新的技术框架的发展，移动端动态跨平台能力的建设已经成为业界的趋势。

效率

为了提升开发效率、减低成本、保持在业界的竞争力，我们集团的移动端开发急需一套适应自身业务场景的高效的动态跨平台开发框架。

目标

业务驱动，完全为集团产品的特定业务场景定制；
跨平台，具备一次开发三端（H5，Android，iOS）运行的能力；
动态，不依赖App版本发布而更新线上业务；
轻量，高性能；
基础设施完善，工具链、调试工具、后台等满足业务开发人员的需求。

业务场景的思考

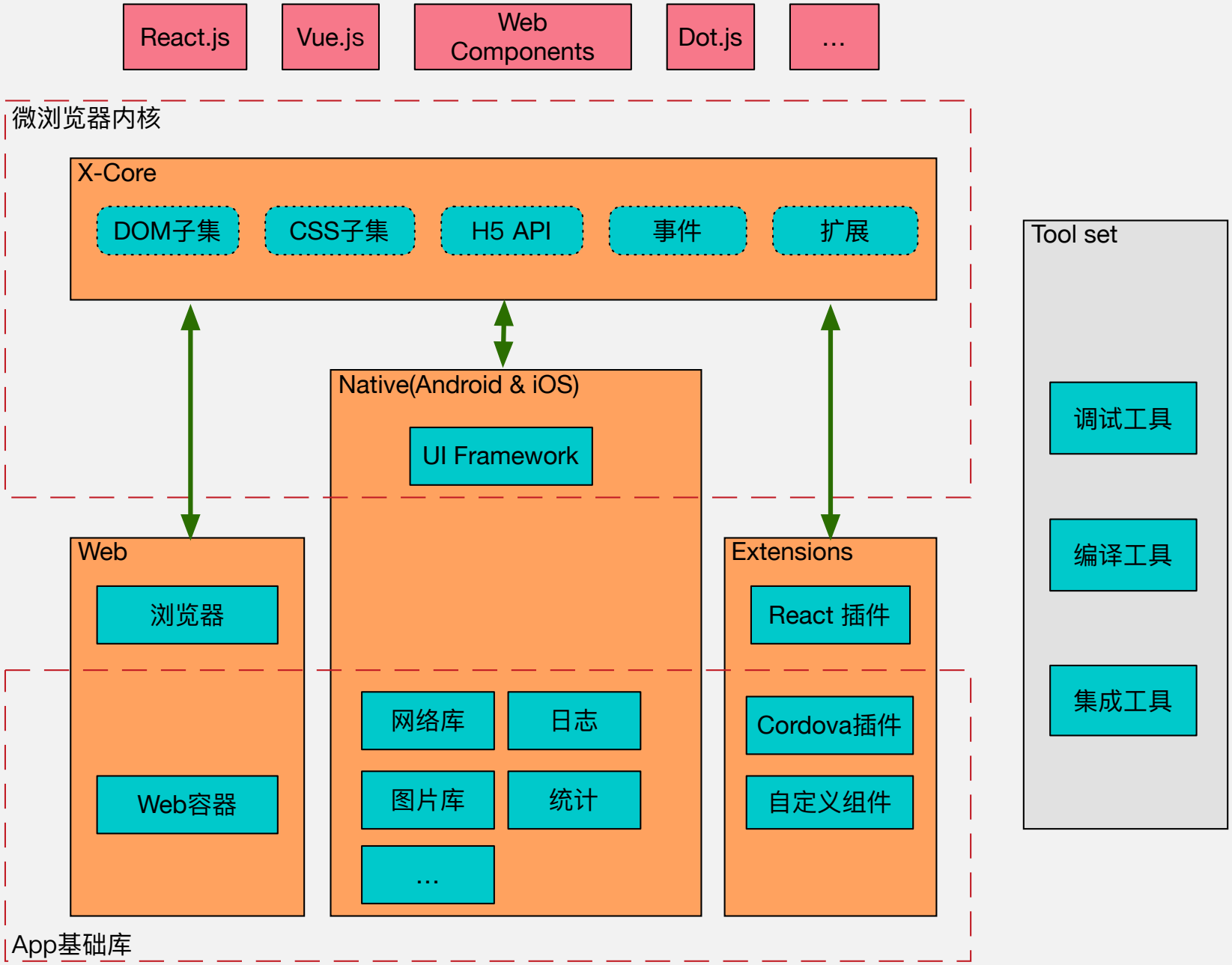
美丽集团App众多，从页面展现的场景出发分为：

- 展示型（活动页，个人信息页，商品详情，类目，信息浏览等等）
- 交互型（Feed流，图片编辑，聊天，直播等）

业务场景驱动的开发模式

- 由简单的交互场景向复杂的交互场景推进
- 由单一场景向多场景推进
- 由对动态、体验、成本要求高的业务向全平台业务推进

优化实践 — 新动态跨平台框架XCore



灵活的前端框架接入

```
import XCore, { Component, ListView } from '@mogu/xcore';
import { App } from '@mogu/xcore-render';

import Banner from '@component/xcore-banner';
import bannerData from './data/banner.js';
class Demo extends Component {
  initialize() {
    return (
      <ListView>
        <Banner models={bannerData} />
      </ListView>
    );
  }
}
App.regist(Demo);
```

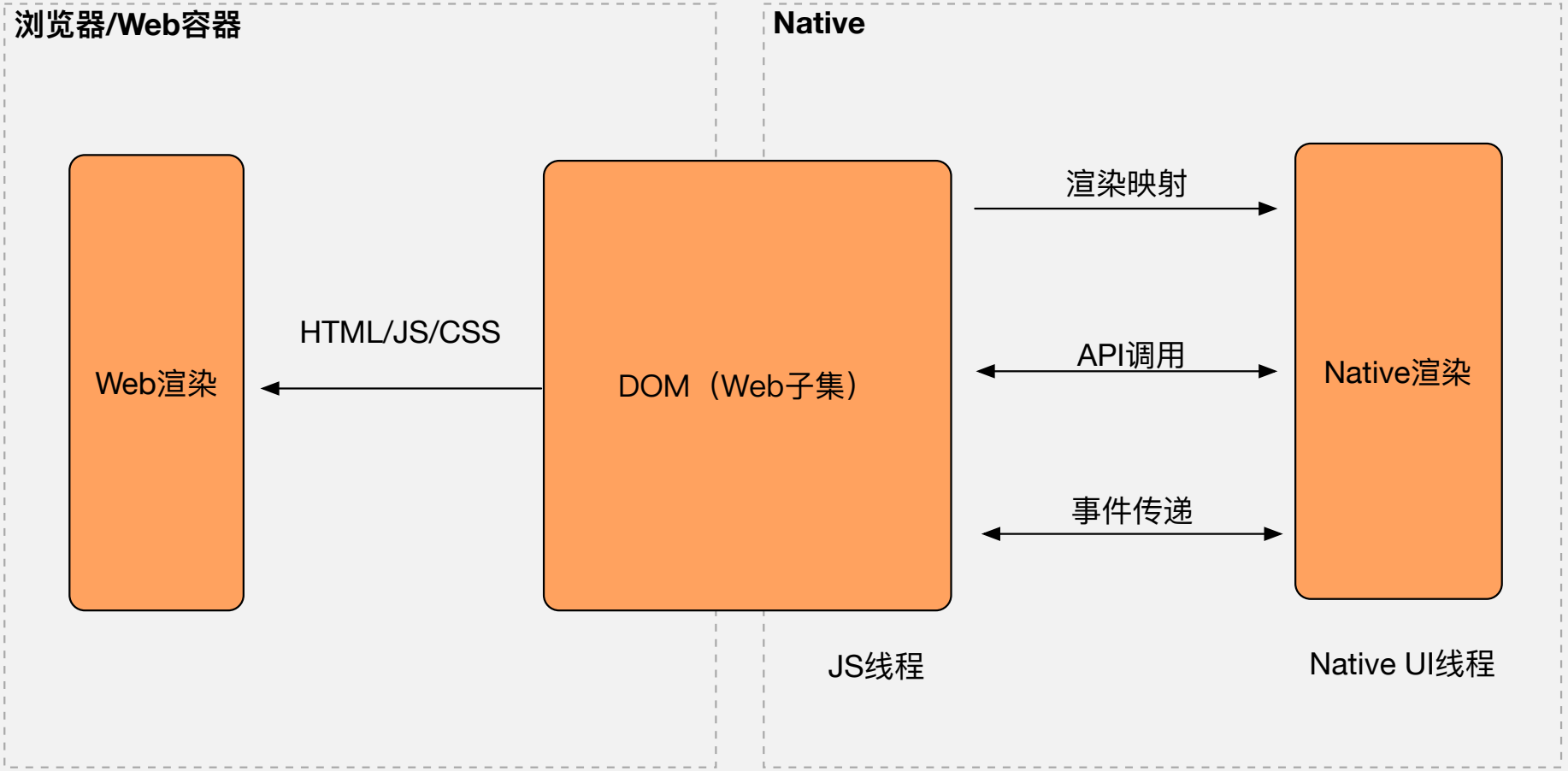
react.js(JSX)

```
import Vue from '../../dist/xcore-vue.js';
import App from './app';

new Vue({
  el: '#app',
  template: '<app></app>',
  components: {
    App
  }
});
```

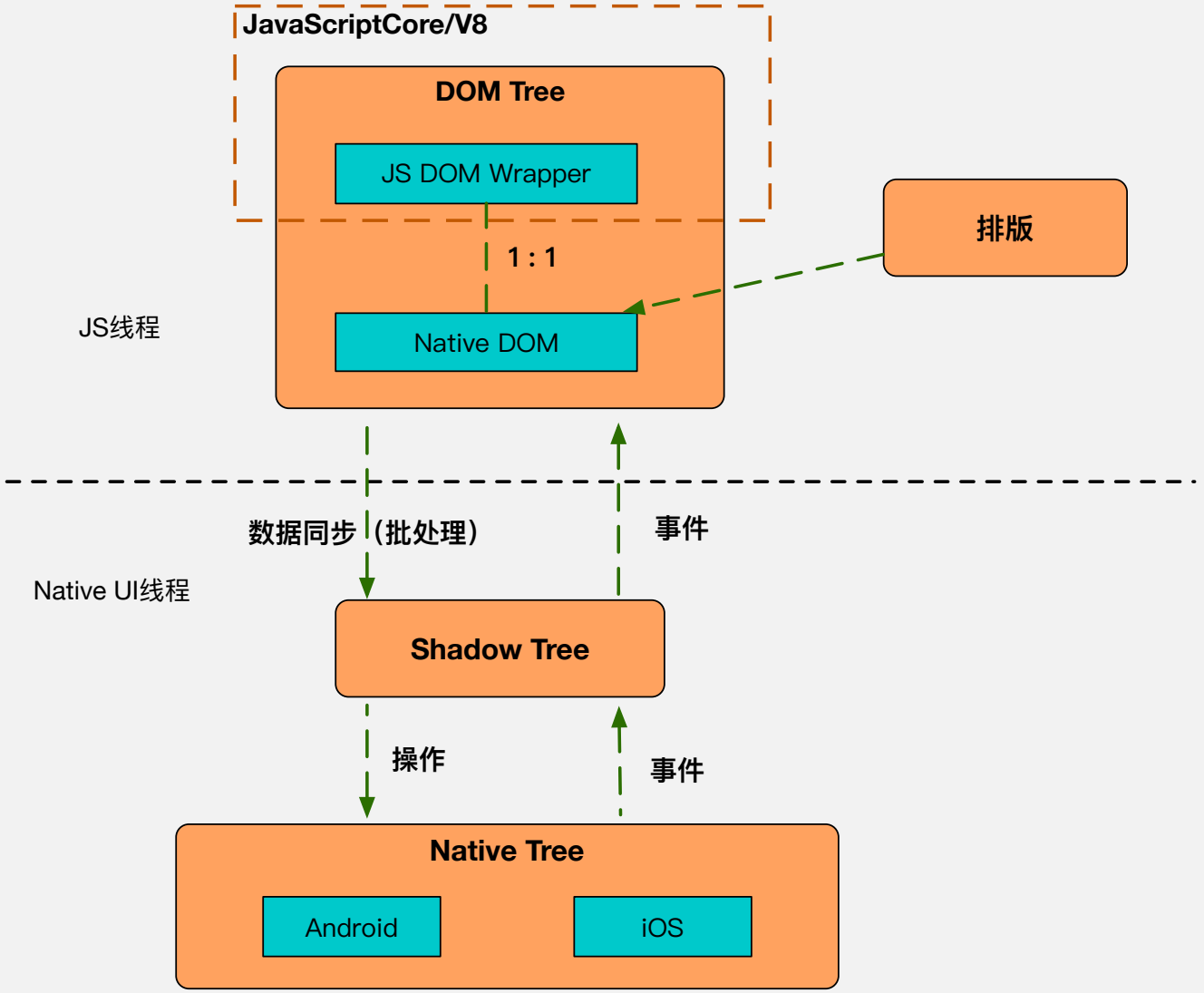
```
<template>
  <view class="headBnner">
    <image :src="bannerImage" style="height:1246"/>
    <view class="activityRule" @click="alertRule">
      <image :src="ruleImage" class="ruleImage"/>
    </view>
    <!— 促销规则弹出框 —>
    <template>
    <view class="mask" v-if="showRule">
      <view  class="tips">
        <label class="tipsWord">{{labelData}}</label>
      </view>
    </view>
    </template>

  </view>

</template>
```

vue.js 2.0

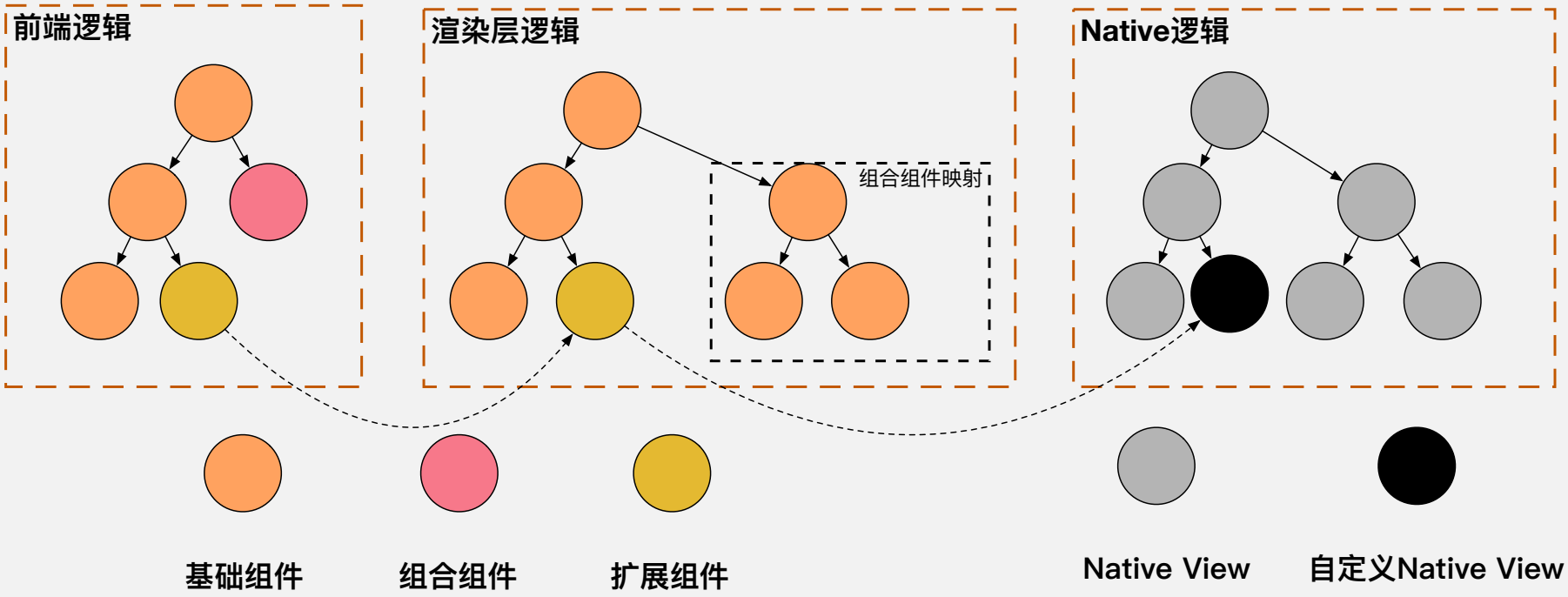
渲染模型


浏览器/Web容器
Native
Web渲染
HTML/JS/CSS
DOM（Web子集）
渲染映射
API调用
事件传递
Native渲染
JS线程
Native UI线程

渲染模型


JavaScriptCore/V8
DOM Tree
JS DOM Wrapper
1 : 1
Native DOM
排版
JS线程
数据同步（批处理）
事件
Native UI线程
Shadow Tree
操作
事件
Native Tree
Android
iOS

组件模型


前端逻辑
渲染层逻辑
组合组件映射
Native逻辑
基础组件
组合组件
扩展组件
Native View
自定义Native View

扩展组件

自定义扩展组件，弥补基本组件的能力不足

- 要求实现三端
- 如：轮播，下拉刷新

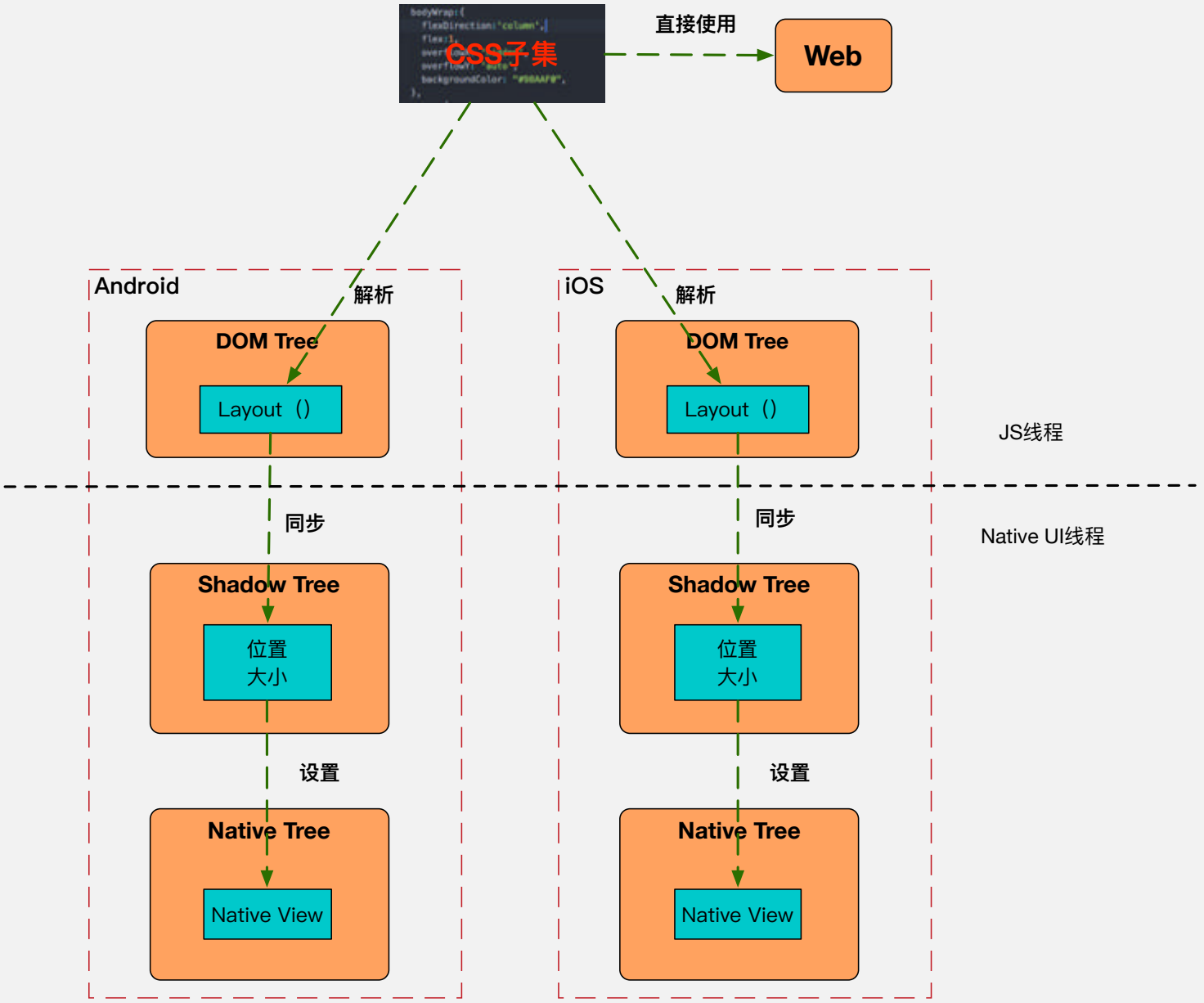
业界通常扩展方式的支持

- Cordova插件支持已完成
- 依据需求考虑支持更多插件，如RN的扩展组件

排版 — Flex布局

```
export default {
  bodyWrap:{
    flexDirection:'column',
    flex:1,
    overflowX: 'hidden',
    overflowY: 'auto',
    backgroundColor: "#98AAF0",
  },
  wrap: {
    position:'relative',
    // flexDirection:'column',
    flex:1
  },
  shopCar: {
    position:'absolute',
    flexDirection:'column',
    bottom:130,
    right:30,
  },
  shopCarImage:{
    width:100,
    height:100
  },
  fixedImage:{
    width:750,
    height:108
  },
  container:{
    position: 'relative',
    flexDirection: 'column',
    justifyContent: 'center',
    alignItems: 'center'
  },
  mask: {
    position:'absolute',
    left:0,
    bottom:0,
    right:0,
    top:0,
    alignItems:'center',
    justifyContent:'center',
  },
    tips: {
      paddingTop: 16,
      paddingBottom: 16,
      paddingLeft: 24,
      paddingRight: 24,
      borderRadius: 4,
      backgroundColor: 'rgba(0, 0, 0, 0.6)',
      maxWidth: 600,
      flexDirection: 'column',
    },
    tipsWord: {
      color: '#fff',
      fontSize: 28,
    }
}
```

排版计算


CSS子集
直接使用
Web
Android
解析
DOM Tree
Layout（）
同步
Shadow Tree
位置
大小
设置
Native Tree
Native View
iOS
解析
DOM Tree
Layout（）
同步
Shadow Tree
位置
大小
设置
Native Tree
Native View
JS线程
Native UI线程

ListView

复用Native ListView组件，实现高性能

- Android ListView
- iOS UITabView

使用姿势产生的问题

- 前端开发往往把逻辑组件当成ListView的Item
- ListView以Item的粒度回收，Item约大性能越差

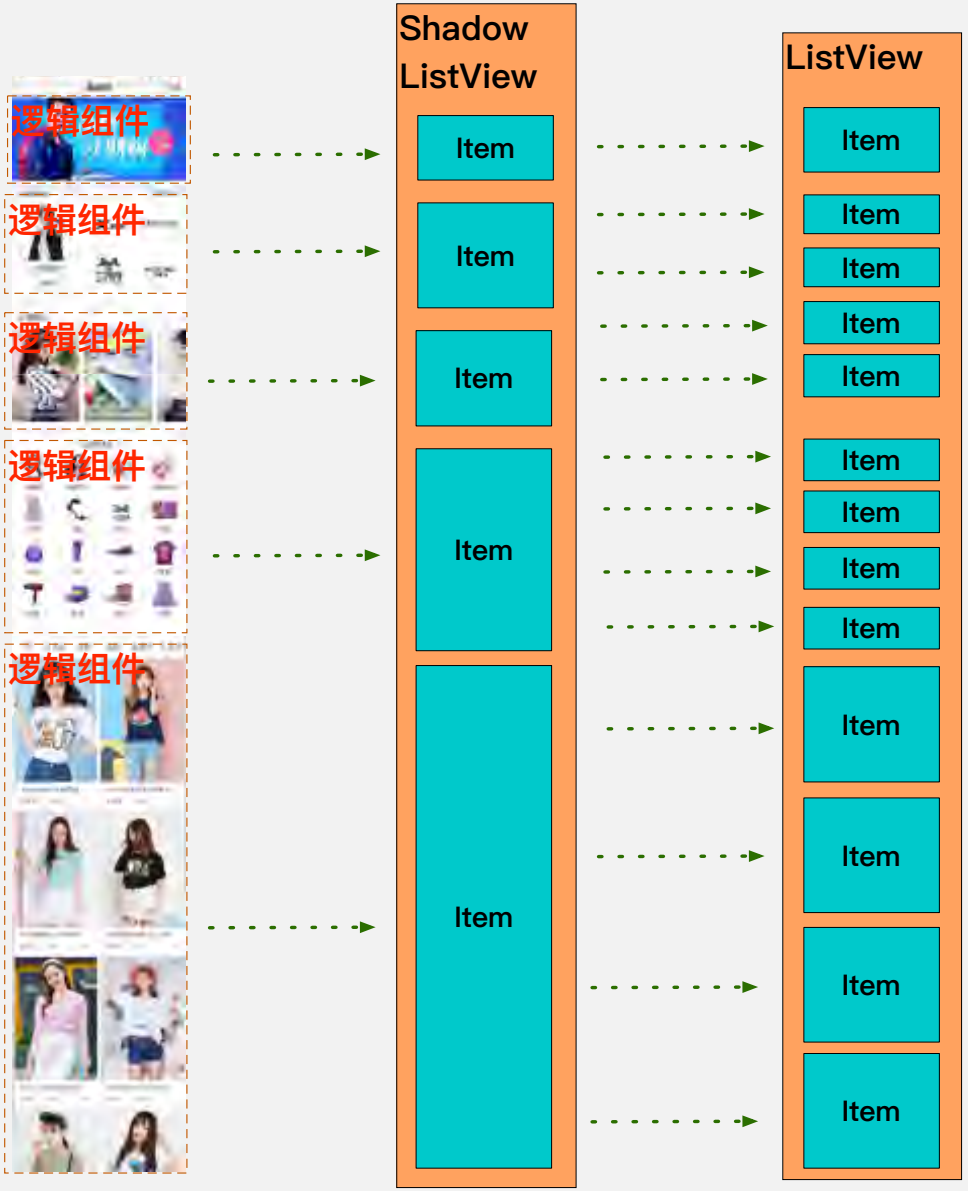
ListView

引入Shadow ListView

按行拆分逻辑组件，并自动将拆分的行映射为真是的ListView Item

优化

- View Pool
- Item构建不需要重排版



通信方式

iOS JSC API

Android — 复用NativeScript

案例1 — 蘑菇街616大促良品会场

案例2 — 蘑菇街品牌站


品牌站
分享
品牌集中营
BRAND IS AN ATTITUDE
品牌热榜
查看更多>
小桃泳衣
SEMIR
森馬
凡兔旗舰店
店铺焕新


品牌站
分享
品牌精选
韩国馆
国货馆
欧美馆
精选特卖
女装
女鞋
内衣
包包
美妆
个护
运动
男友
家居
配饰
食品
母婴
T恤
连衣裙
凉鞋
单鞋
单肩包
双肩包


品牌站
分享
单肩包
双肩包
内衣
泳衣
面膜
防晒

与RN，Weex的对比

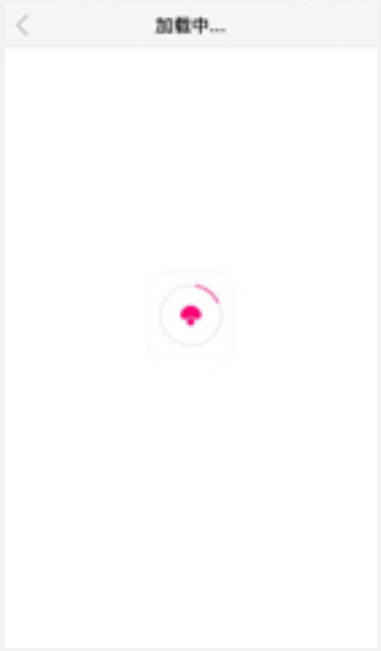
定位与目标不同

- 特定业务场景驱动
- 轻量、高效、灵活的独立内核

架构设计不同

- 不绑定前端框架
- 微浏览器架构

页面资源离线化



HTML

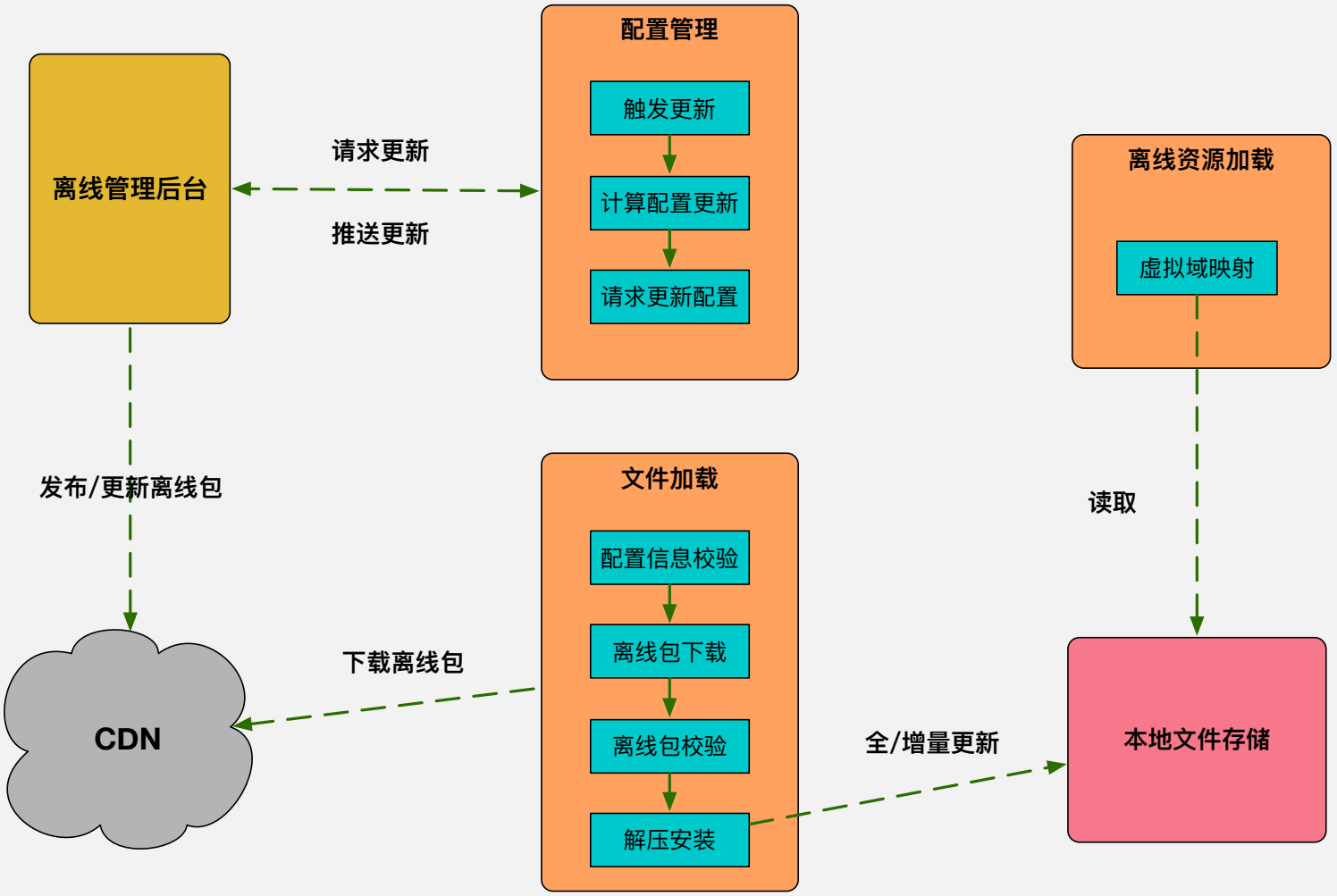
JS

CSS

数据

图片

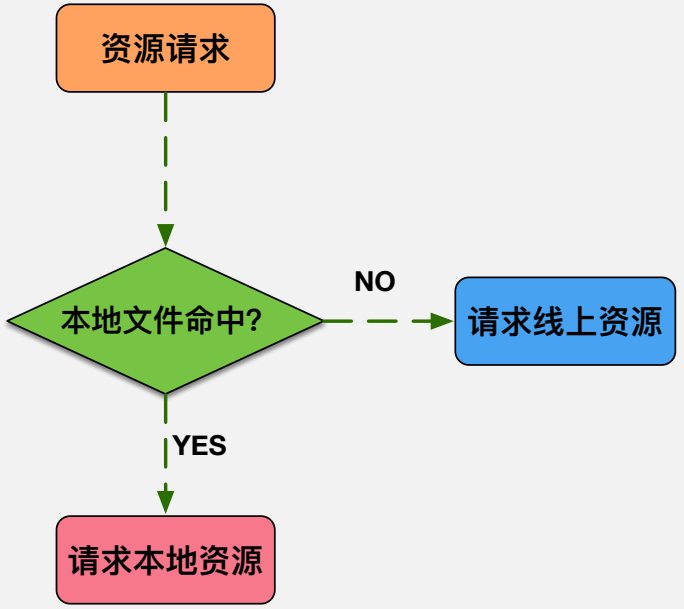
离线化的整体结构


离线管理后台
请求更新
推送更新
配置管理
触发更新
计算配置更新
请求更新配置
离线资源加载
虚拟域映射
发布/更新离线包
文件加载
读取
配置信息校验
离线包下载
离线包校验
解压安装
CDN
下载离线包
全/增量更新
本地文件存储

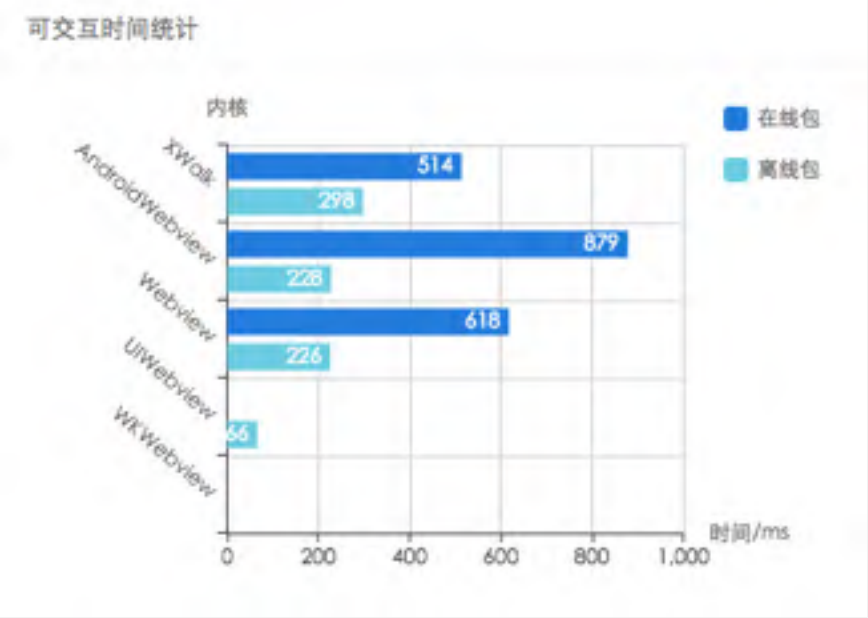
虚拟域的映射拦截

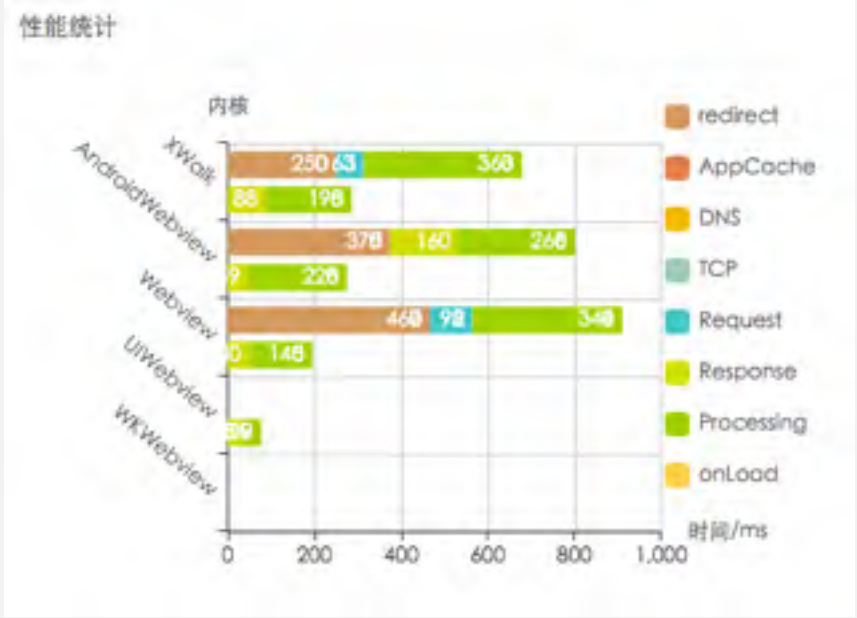
iOS — URLProtocol

Android — ShouldInterceptRequest



离线化效果


可交互时间统计
内核
在线包
离线包
XWalk
AndroidWebview
Webview
UIWebview
WKWebview
514
298
879
228
618
226
36
0
200
400
600
800
1,000
时间/ms


性能统计
内核
redirect
AppCache
DNS
TCP
Request
Response
Processing
onLoad
XWalk
AndroidWebview
Webview
UIWebview
WKWebview
250
63
360
88
198
378
160
266
9
226
460
92
348
0
148
89
0
200
400
600
800
1,000
时间/ms

计划

整合与融入前端开发体系

离线化体系的深入完善

XCore与前端开发体系的整合

展望

移动端开发方式的变革

THANK YOU