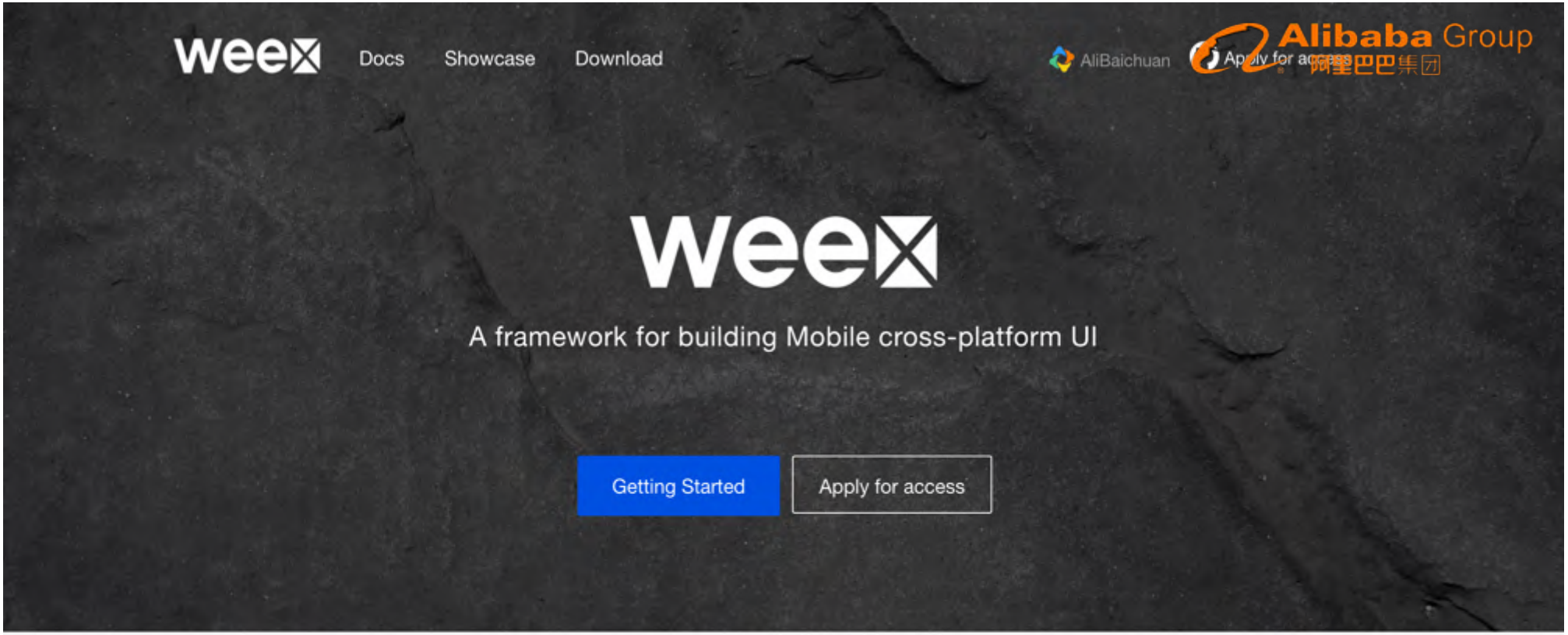
Weex 架构简介和性能优化实战

手机淘宝：郑士汉 (饮源)



https://github.com/alibaba/weex

Alibaba Group 阿里巴巴集团

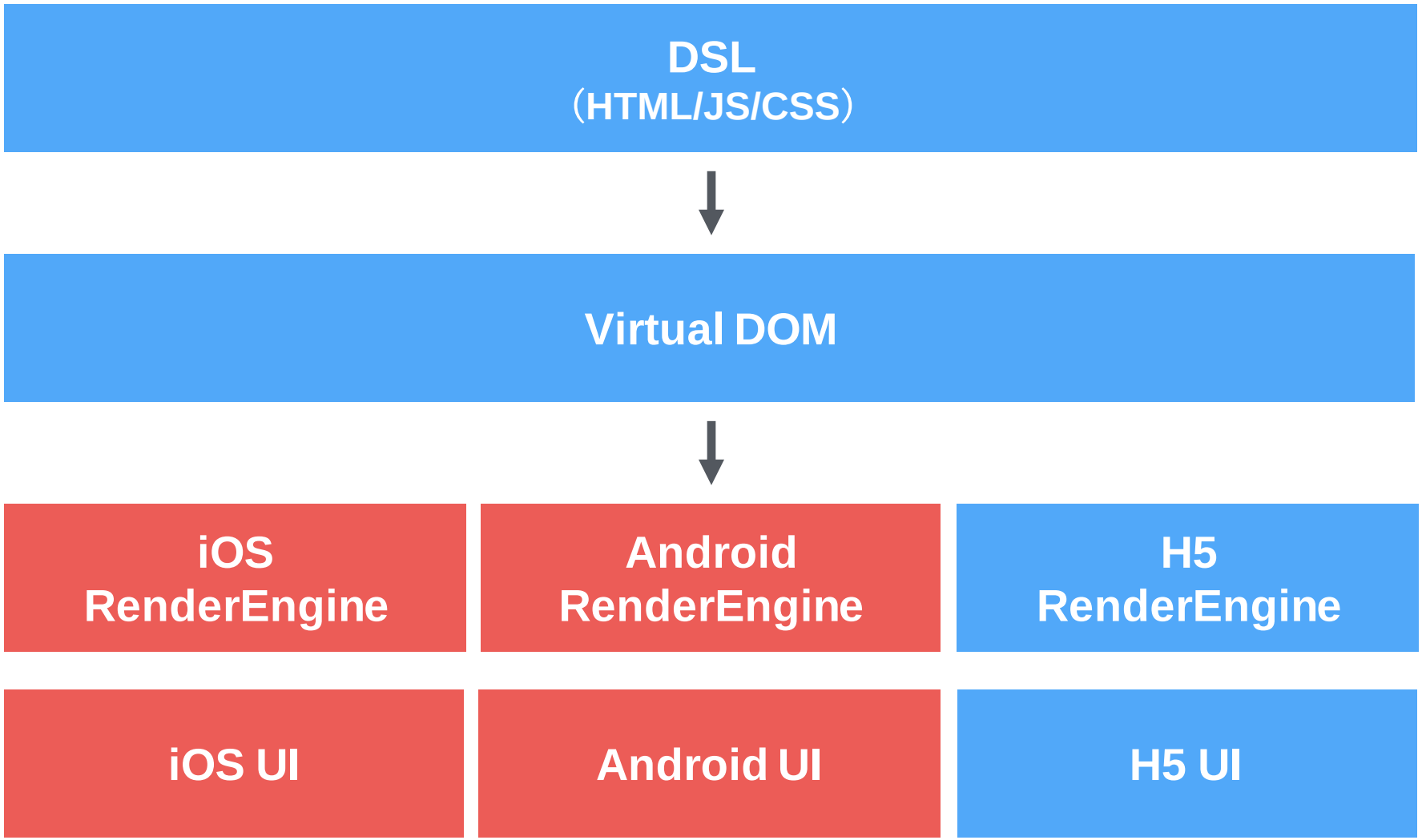
目录

- 1、Weex框架简介
- 2、Weex性能优化实践
- 3、最佳实践

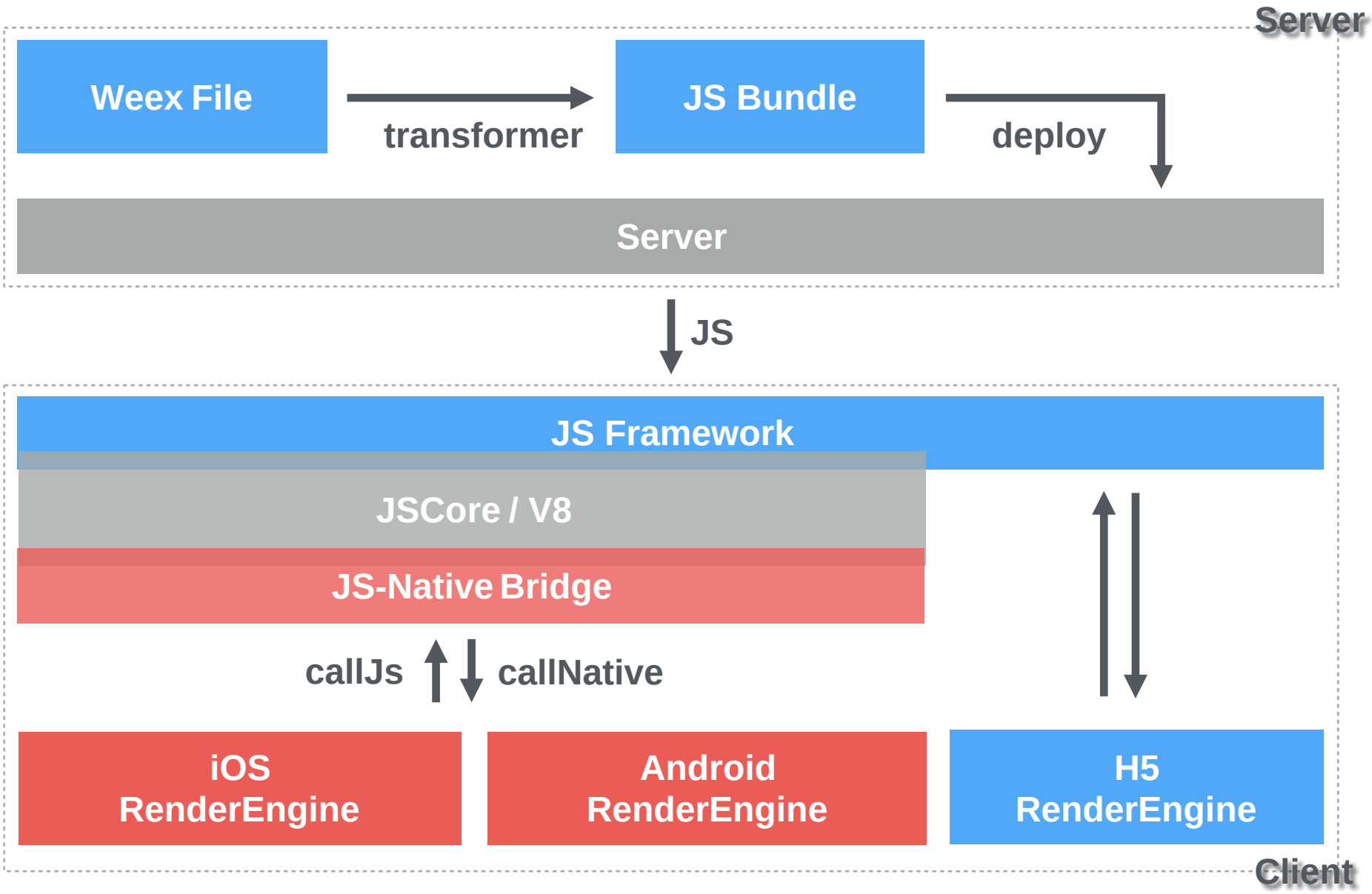
Alibaba Group 阿里巴巴集团

Weex框架简介

简明架构图


DSL
（HTML/JS/CSS）
Virtual DOM
iOS
RenderEngine
Android
RenderEngine
H5
RenderEngine
iOS UI
Android UI
H5 UI

简明架构图

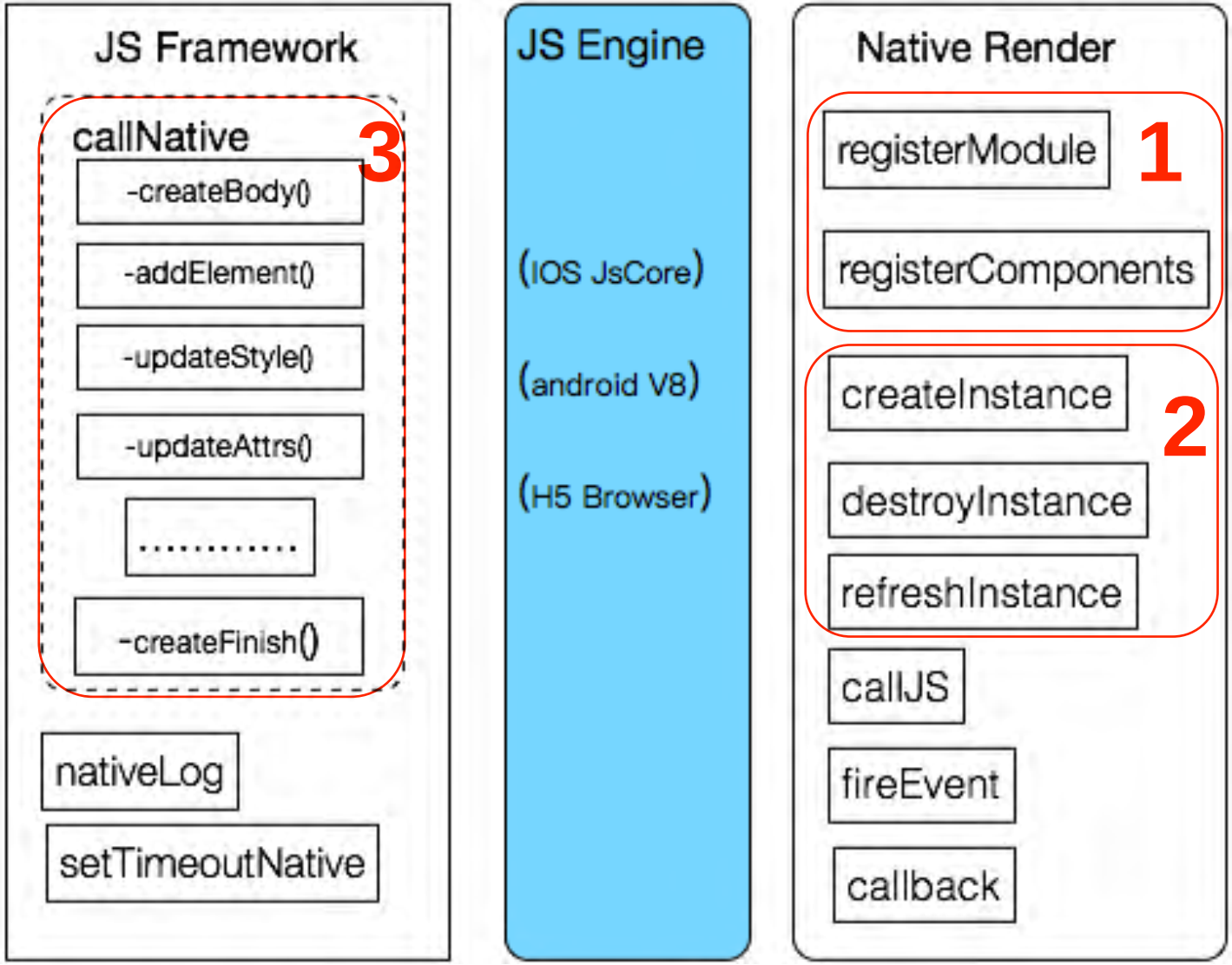

Server
Weex File
transformer
JS Bundle
deploy
Server
JS
JS Framework
JSCore / V8
JS-Native Bridge
callJs
callNative
iOS
RenderEngine
Android
RenderEngine
H5
RenderEngine
Client

<template>

<text>{{message0}}</text>
<text>{{message1}}</text>

</template>
<style>
.hello {
flex-direction: row;
}
</style>
<script>
module.exports = {
data: {
message0: 'Hello',
message1: ' World.'
},
methods: {
clickHandler: function() {}
}
}
</script>
module.exports.template = {
"type": "div",
"classList": [
"hello"
],
"events": {
"click": "clickHandler"
},
"children": [
{"type": "text"...},
{"type": "text"...}
]
};
module.exports.style = {
"hello": {
"flexDirection": "row"
}
};
module.exports = {
data: function() {
return {
message0: 'Hello',
message1: ' World.'
}
},
methods: {
clickHandler: function() {}
}
};
DSL
JS-Bundle

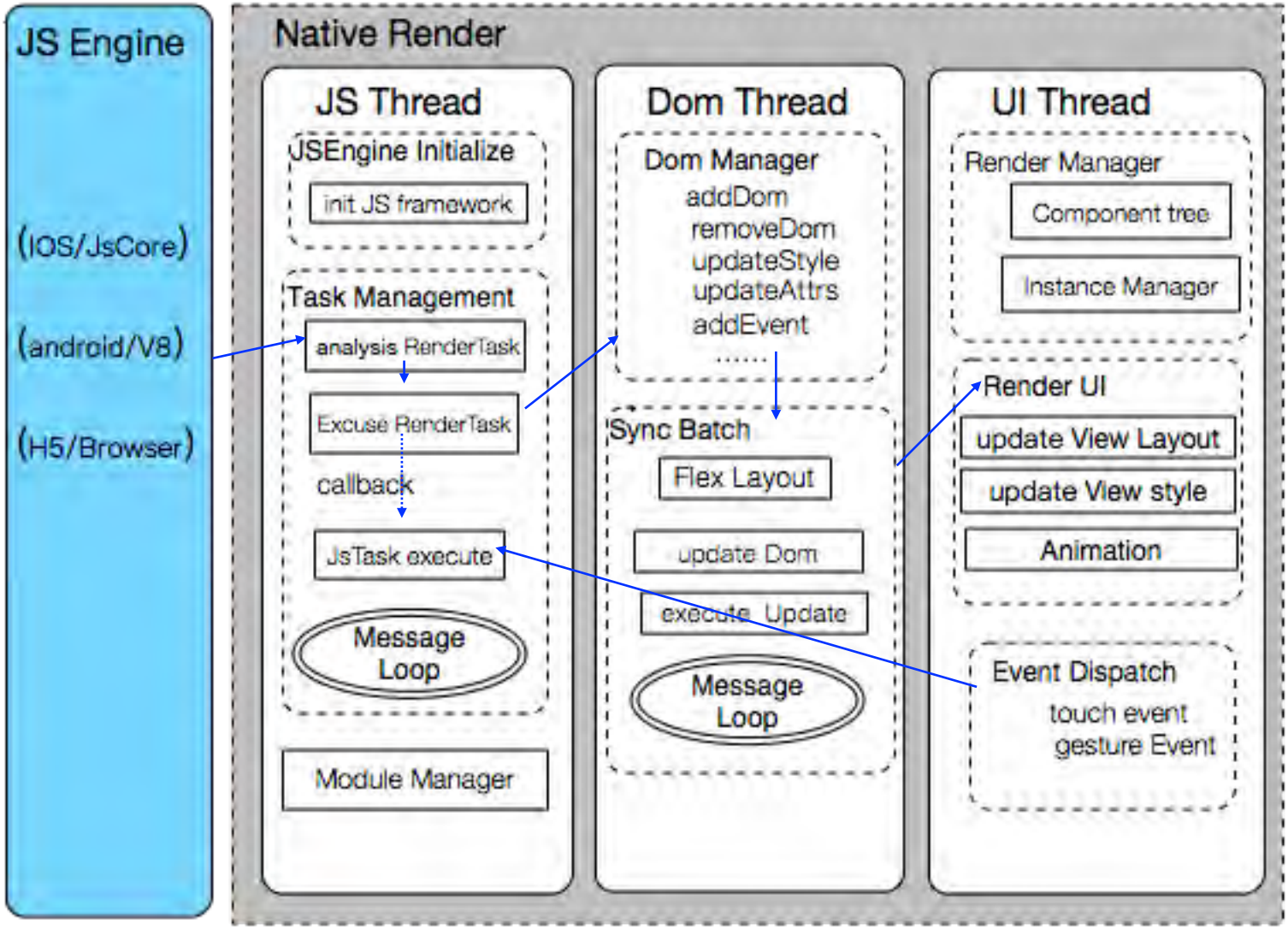
JS&Native 通信协议



Native Render

```
call Native >>>> instanceId:1, tasks:
[{"module":"dom","method":"addElement","arg
s":["_root",{"ref":"186","type":"div","attr":
{},"style":{}},-1]}], callback:185

callJS >>>> instanceId:5 tasks:[{"args":
["185","{}"],"method":"callback"}]
```



Alibaba Group 阿里巴巴集团

性能优化实践

Alibaba Group 阿里巴巴集团

网络＋渲染：秒开

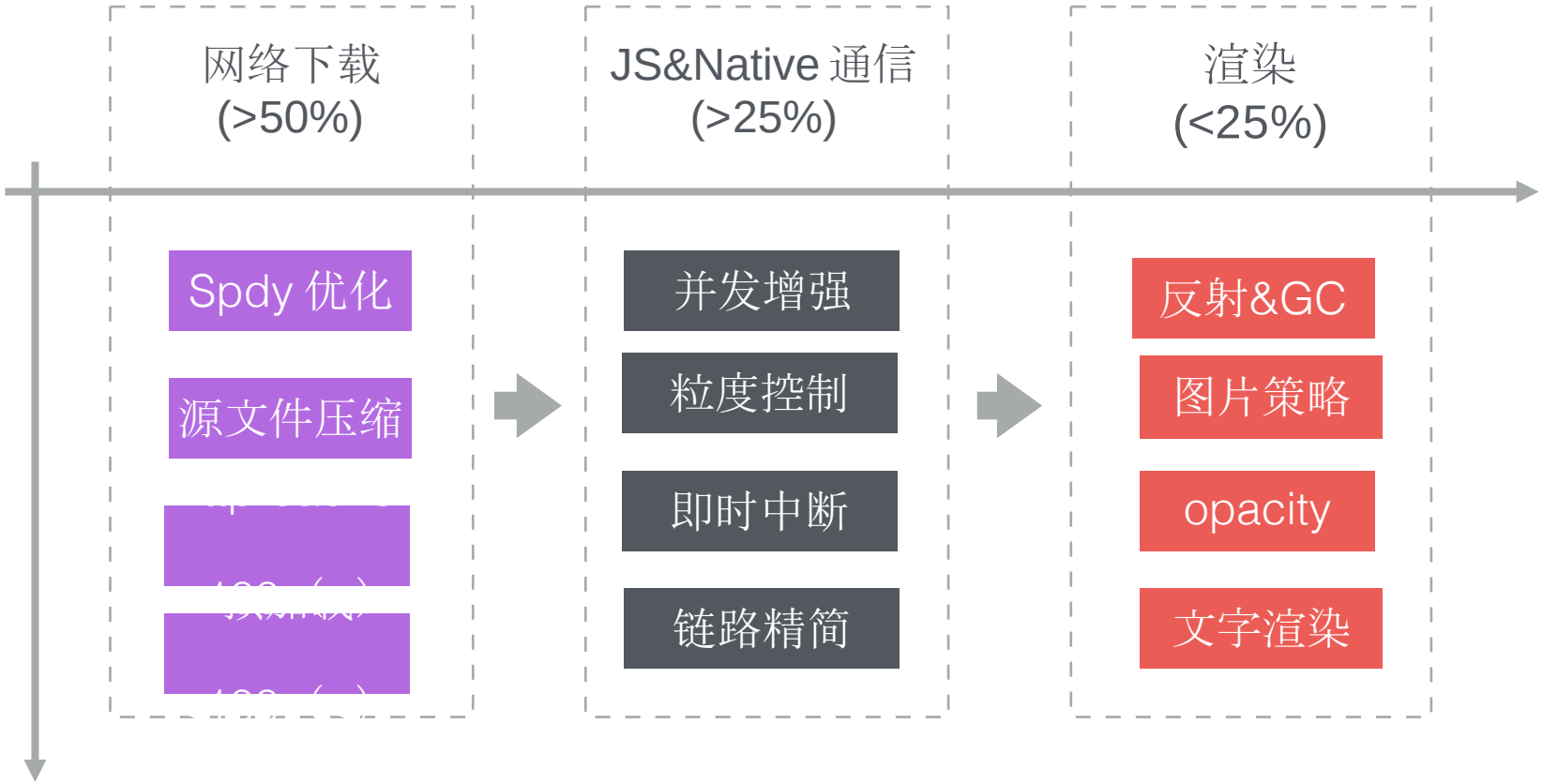
Alibaba Group 阿里巴巴集团

问题与挑战

- 1、JS bundle下载慢（压缩后60k， NetworkTime>800ms）
- 2、JS和Native通信效率低，拖慢加载时间
- 3、长页面VDom渲染时间慢，占首屏时间40%左右
- 4、JSThread过于繁忙（Json解析&反射调用）
- 5、JS Task无法抢占执行，导致新页面打开慢
- 6、复杂页面滚动帧率低的问题

Alibaba Group 阿里巴巴集团

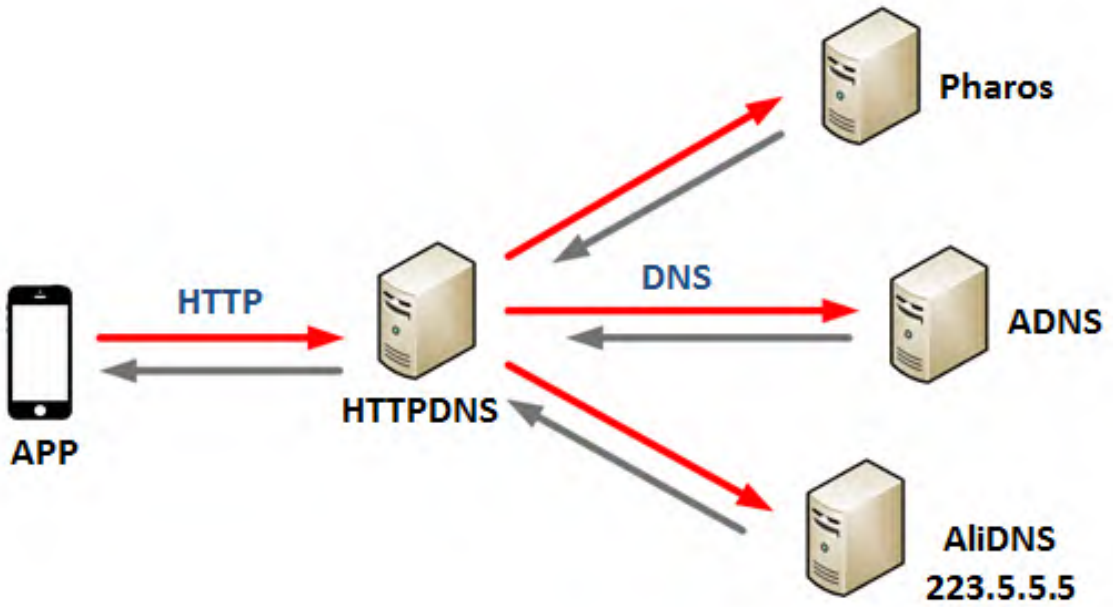
Weex TimeLine



Alibaba Group 阿里巴巴集团

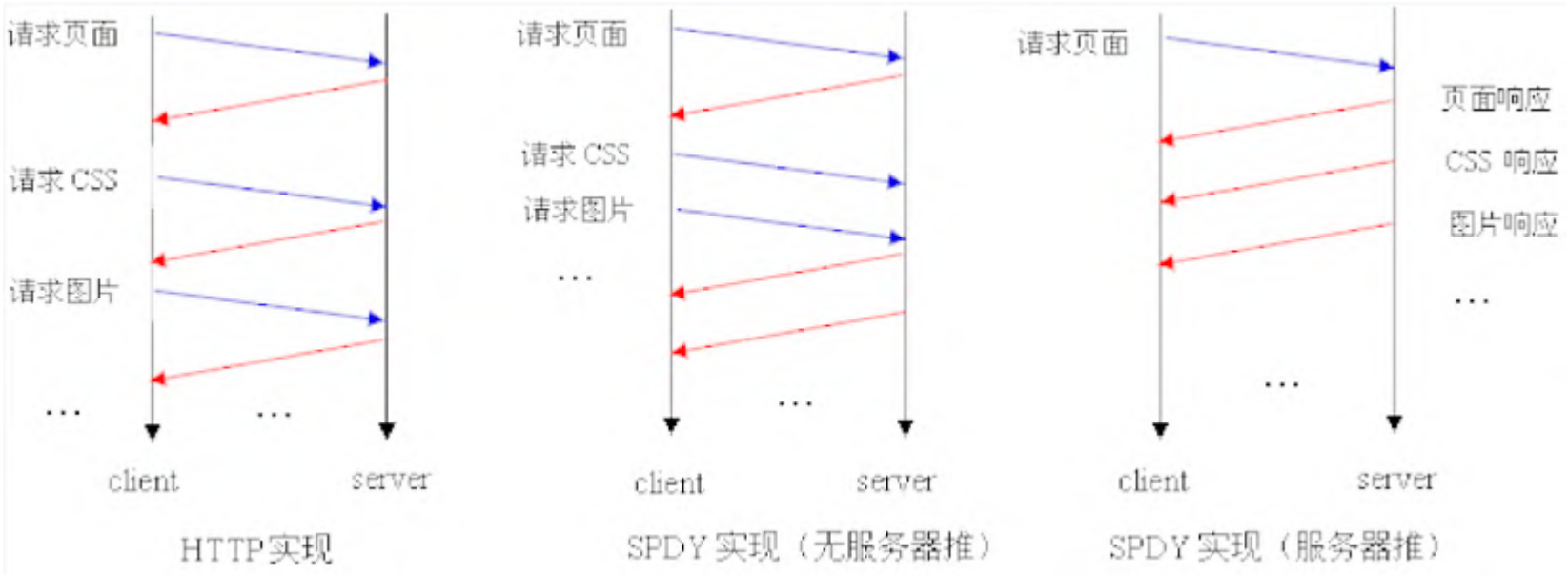
网络下载优化

HTTP-DNS



1. 防止域名劫持
2. 更精准的调度，更快的建连
3. 额外的域名相关信息

SPDY



1. **多路复用请求优化**
2. **服务器推送技术**
3. **SPDY压缩http头**

网络优化

Alibaba Group 阿里巴巴集团

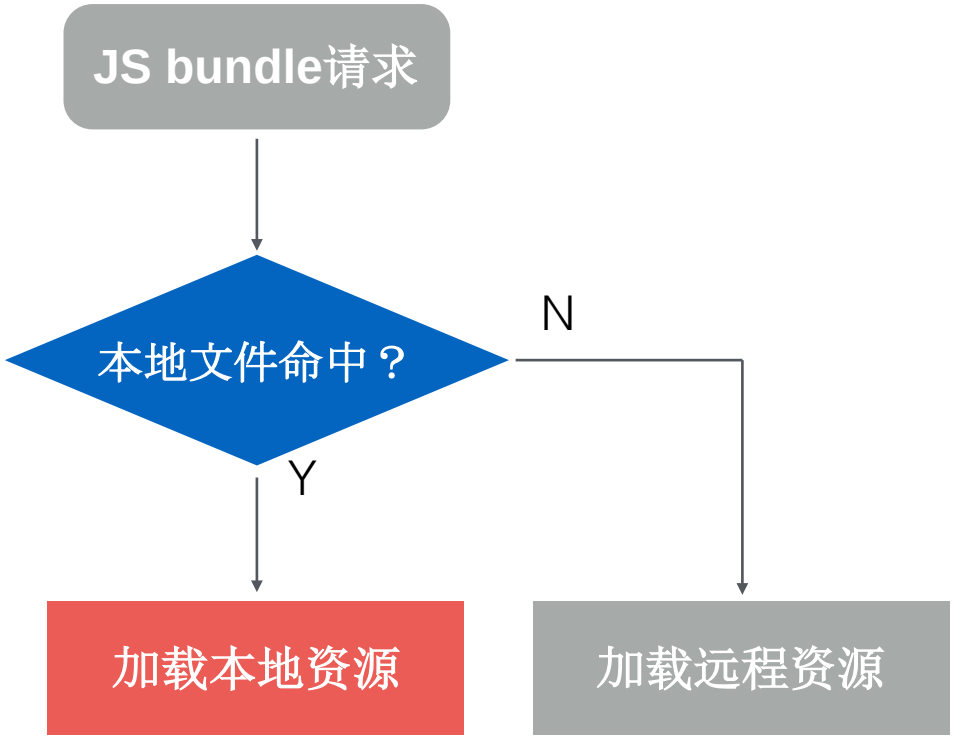
网络问题：

- 1、优化前状况：iOS 280~450ms Android 700~900ms
- 2 、业务JS文件过大，gzip压缩后（60～80k)
- 3、域名没有收敛
- 4、没有配置Spdy协议
- 5、没有httpCache，200请求

线上性能数据对比：

日期	App版本	页面名称	连接类型	样本数	networkTime	样本数占比（本页）
2016-10-09	6.0.0	DetailWeexFragment	cache	[illegible]	106.69	0.64%
2016-10-09	6.0.0	DetailWeexFragment	http	[illegible]	13159.59	0.01%
2016-10-09	6.0.0	DetailWeexFragment	https	[illegible]	10397.83	1.38%
2016-10-09	6.0.0	DetailWeexFragment	packageApp	[illegible]	40.19	23.55%
2016-10-09	6.0.0	DetailWeexFragment	spdy	[illegible]	248.30	1.28%
2016-10-09	6.0.0	DetailWeexFragment	spdy_0rtt_cdn	[illegible]	266.83	73.14%

预加载

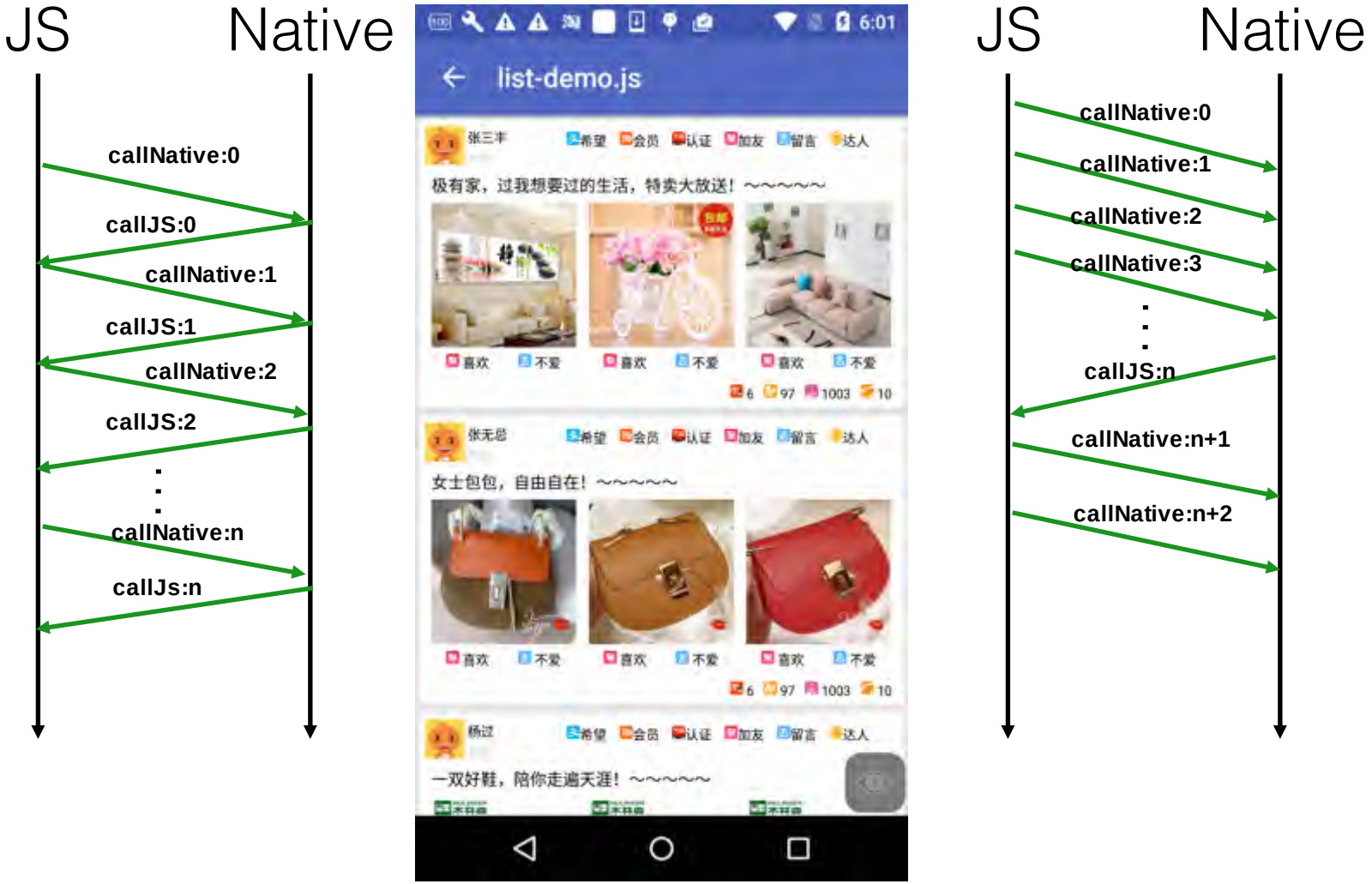


会场典型值	iOS	Android
走网络	400ms	850ms
预加载	**10ms**	**35ms**

Alibaba Group 阿里巴巴集团

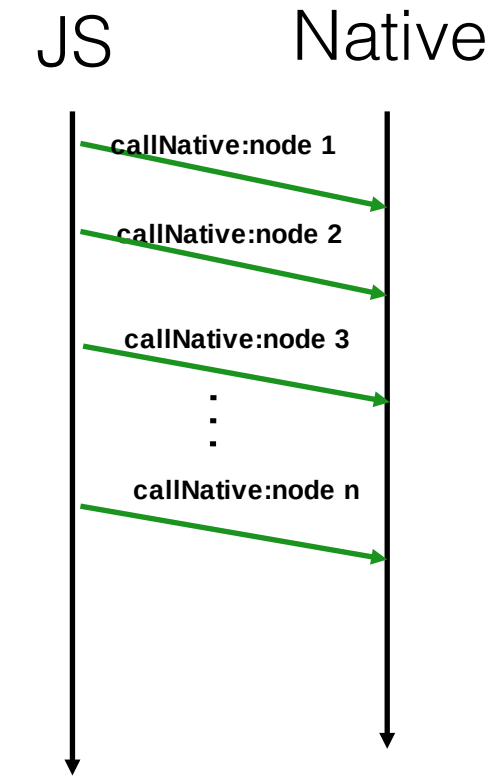
Weex 通信机制

并发增强

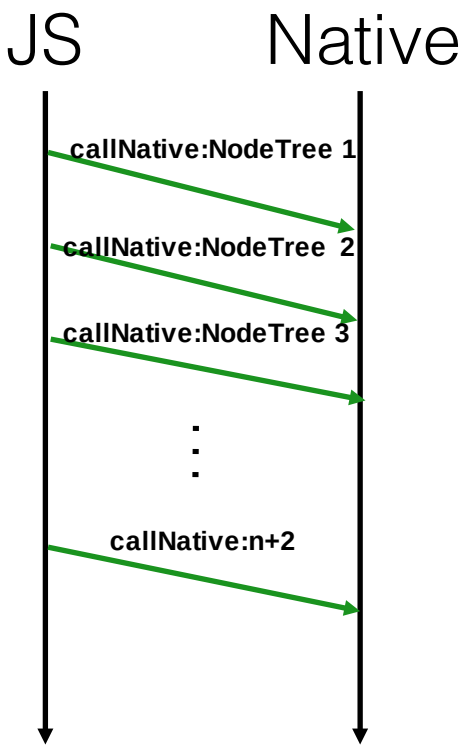


压测页面加载性能：加载性能提升 30%

粒度控制



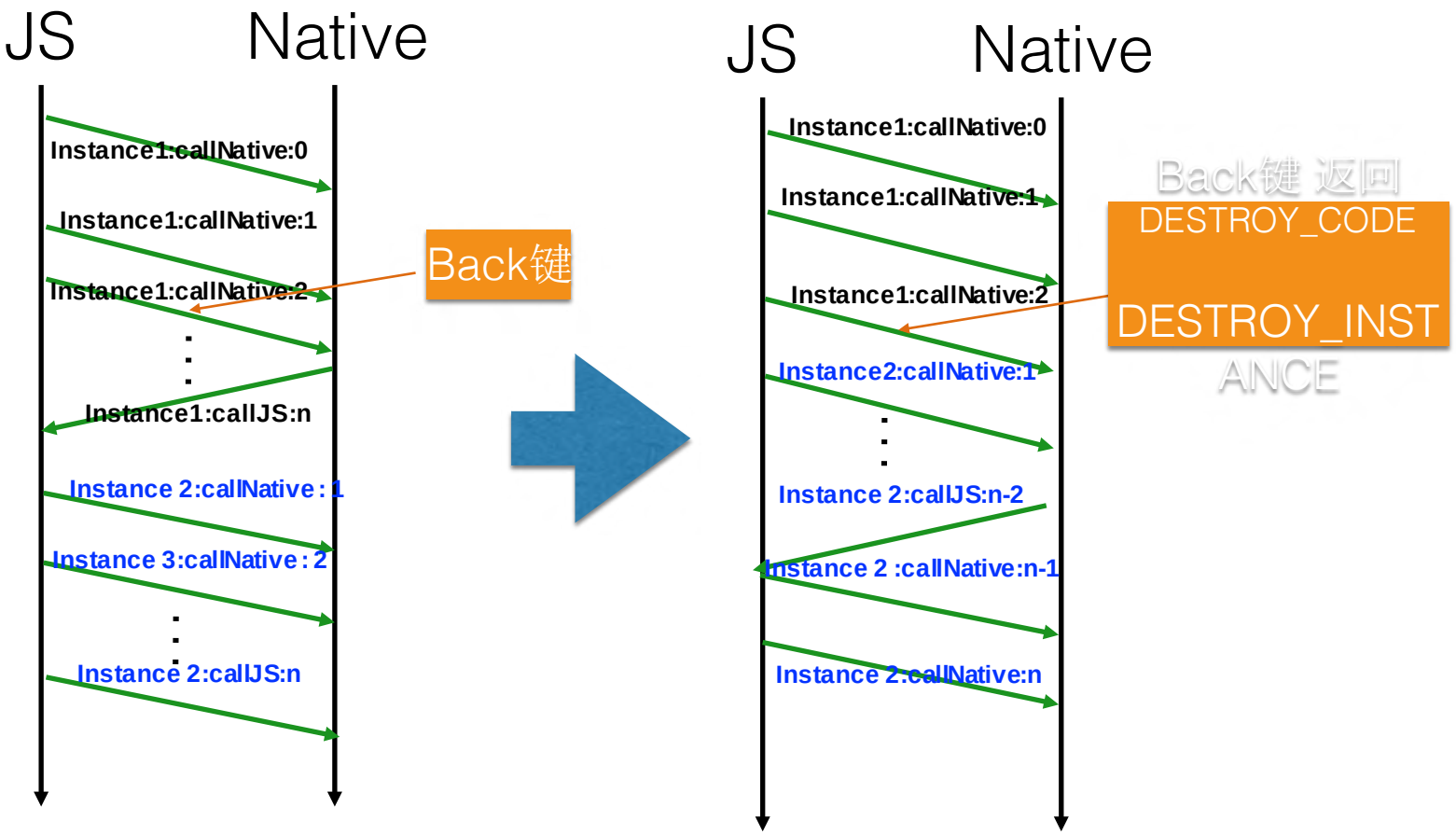
append＝node



append＝tree

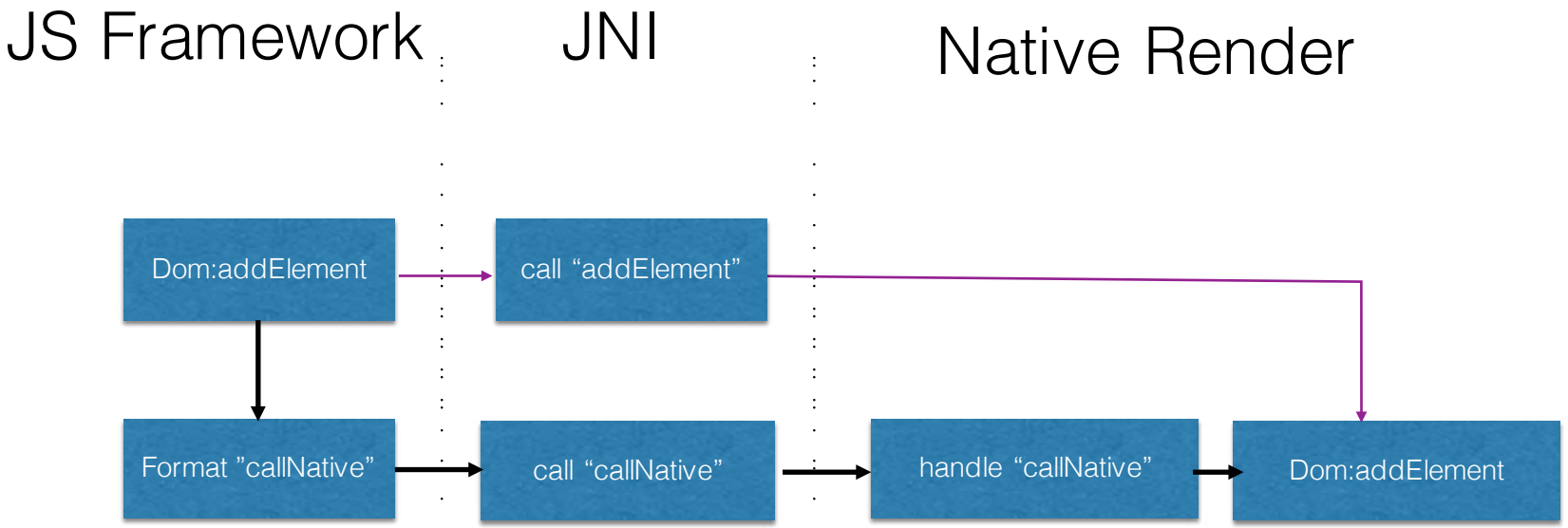
压测页面加载性能：append＝tree方式比append＝node 提升10%

即时中断



解决长页面快速进出，无法中断的问题

链路精简



高频通信接口上浮，减少中间数据转化消耗，加载性能平均提升15%

Alibaba Group 阿里巴巴集团

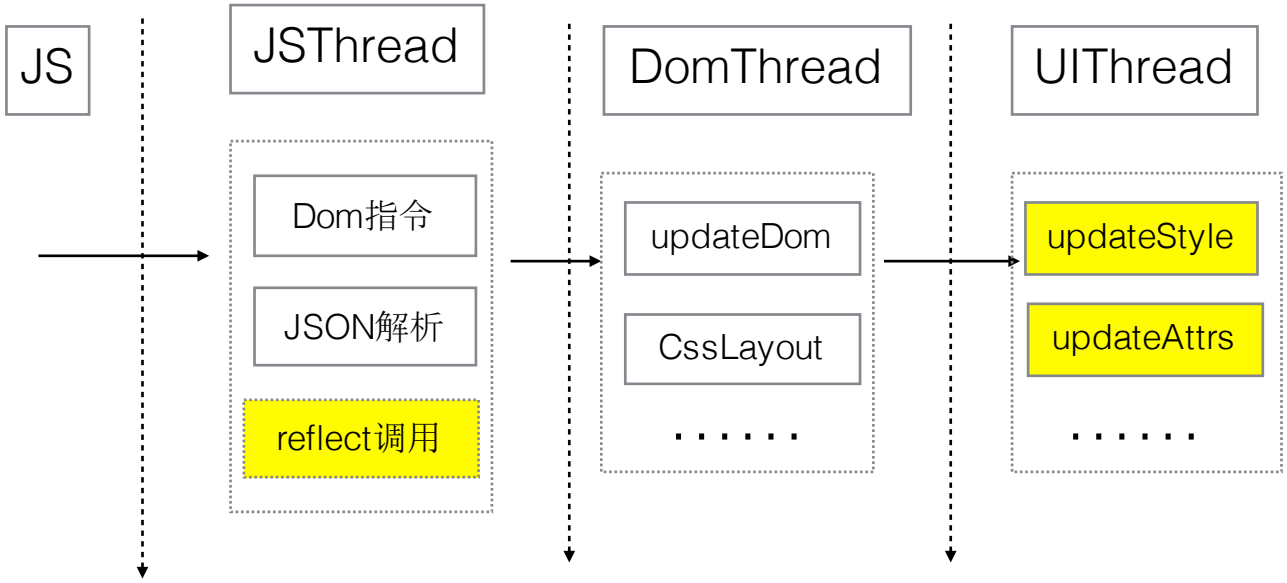
渲染优化

Alibaba Group 阿里巴巴集团

渲染优化

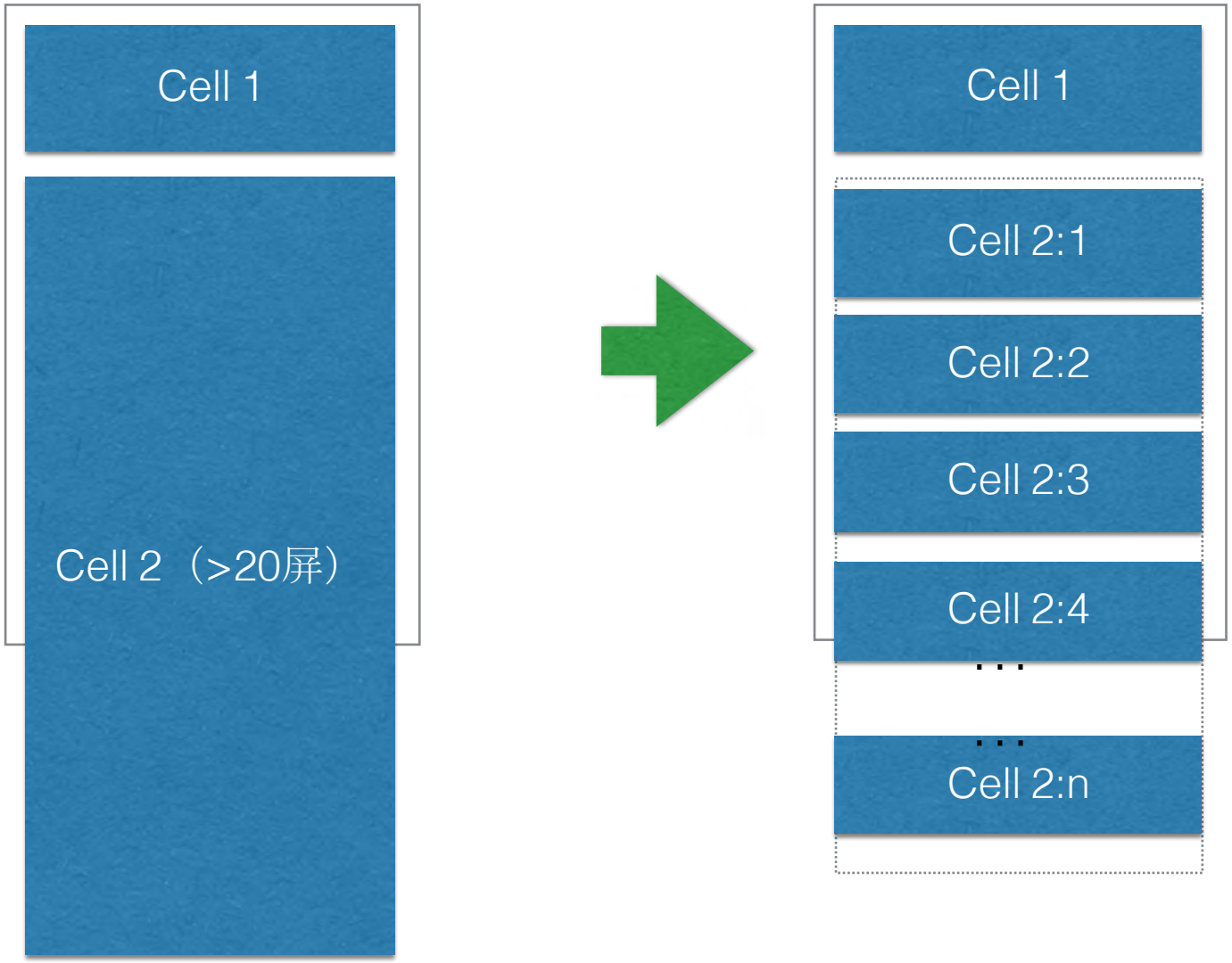
- 帧率优化
  - 图片加载策略（域名收敛、webp收敛、尺寸复用等）
  - 反射&GC优化（Android Device Monitor）
  - List最佳实践（一个List 一个10屏的Cell？）
  - opacity属性优化（大坑）
- 文字渲染优化（截断、性能）

反射&GC优化



优化效果：加载时间提升10%，帧率提升5帧

List最佳实践



Opacity 属性问题

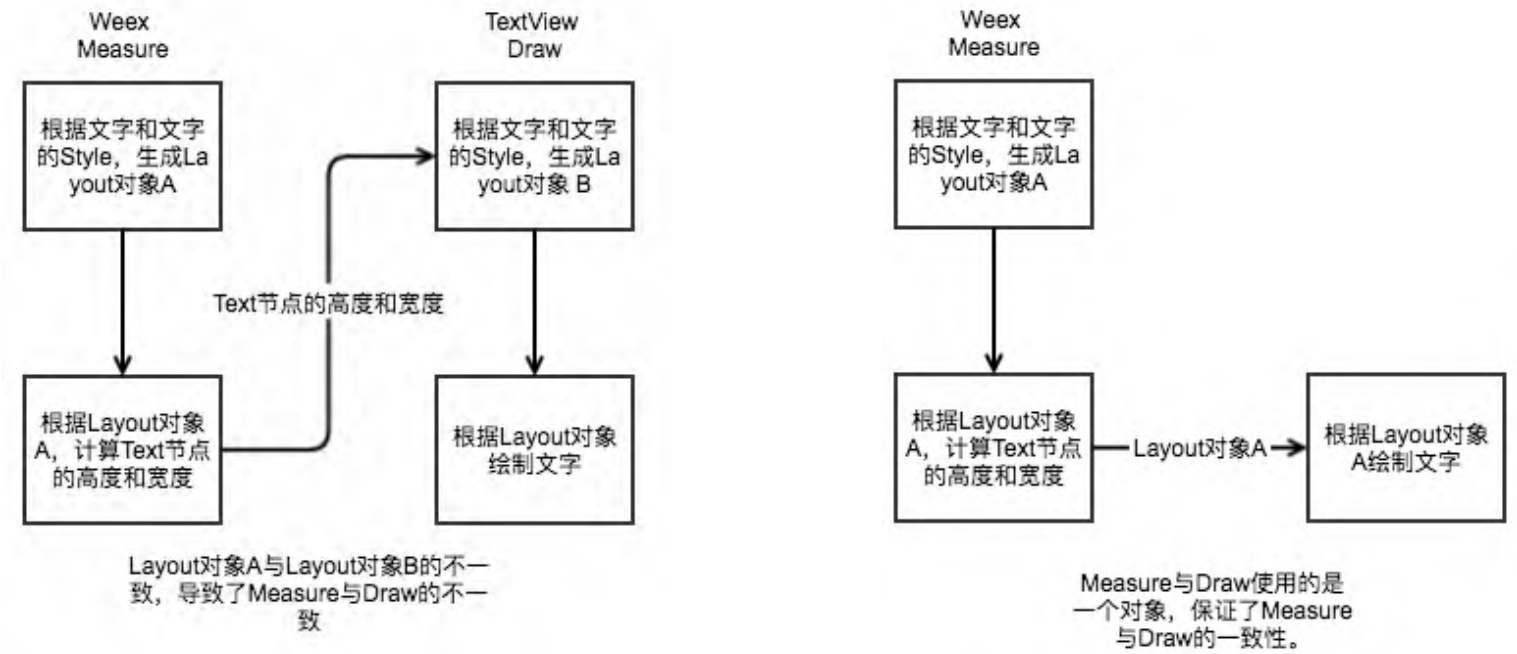
解决方案：setLayerType(View.LAYER_TYPE_HARDWARE, null);
优化效果：38帧－>55帧

Text优化

优化前(文字被截断)

优化后(正常渲染)

Text优化



Measure和Draw使用了同一个对象，避免了额外的对象创建，预计提升10%左右

最佳实践

- 1、dom 数量多会严重影响性能，因此必须尽可能减少同时间产生过多的 dom ；
- 2、长列表建议使用 list，每个Cell尽量拆分；
- 3、使用 list + loadmore 方式实现长列表分页加载；
- 4、实现tab切换功能的时候，注意list标签的个数；
- 5、点击跳转的场景建议使用 a 标签而不是透过 onclick 绑定的方式跳转。

Alibaba Group 阿里巴巴集团

Q&A

Weex Github地址： https://github.com/alibaba/weex

THANKS

SequeMedia 盛拓传媒 | ChinaUnix | ITPUB