

探索H5性能天花板

腾讯NOW直播极限优化实践

腾讯 IVWEB 谢清贵



NOW直播

个人简介

2014年 携程

旅游事业部

2017年 ~ 至今 腾讯

- 腾讯NOW直播
- IVWEB成员
- 主要负责移动Hybrid APP内业务以及前端性能优化



<https://ivweb.io/>

APP开发演进

迭代缓慢

必须跟客户端版本

开发&学习成本较高

安卓&iOS需要两端分别开发一套

客户端安装包很大

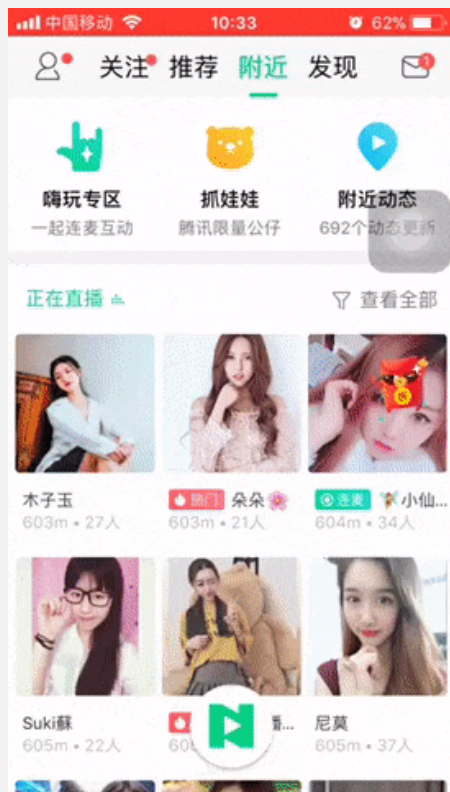
所有业务都必须打包



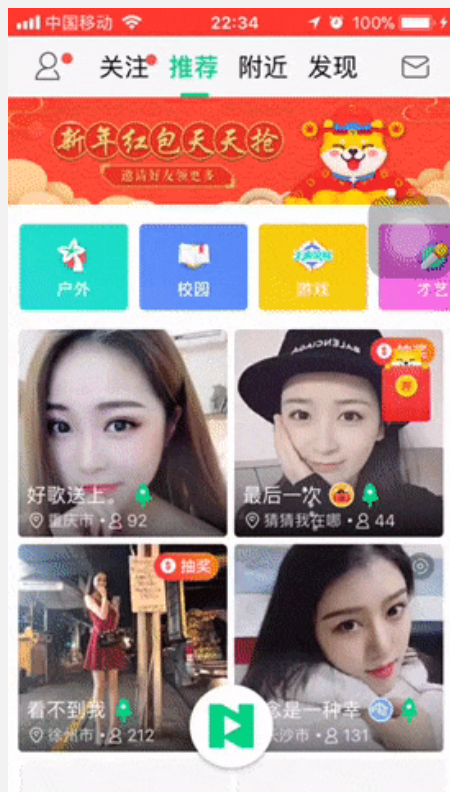
H5页面打开速度慢

用户等待白屏时间长

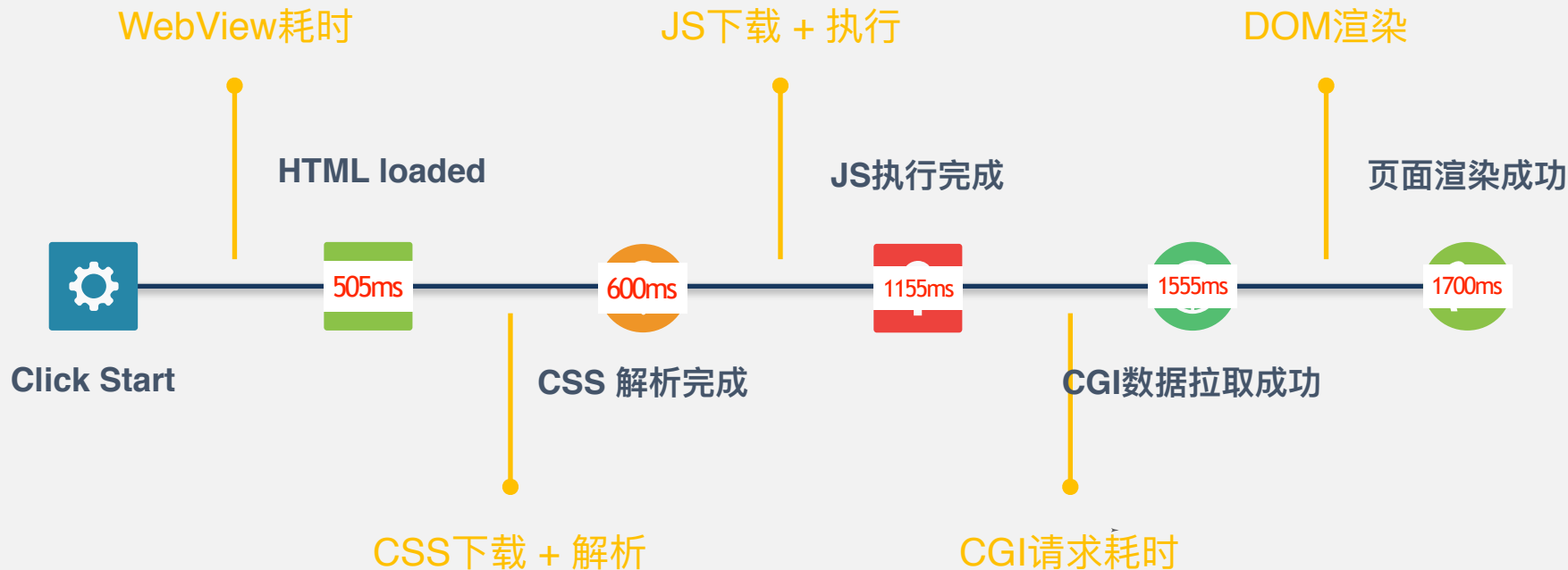
原生



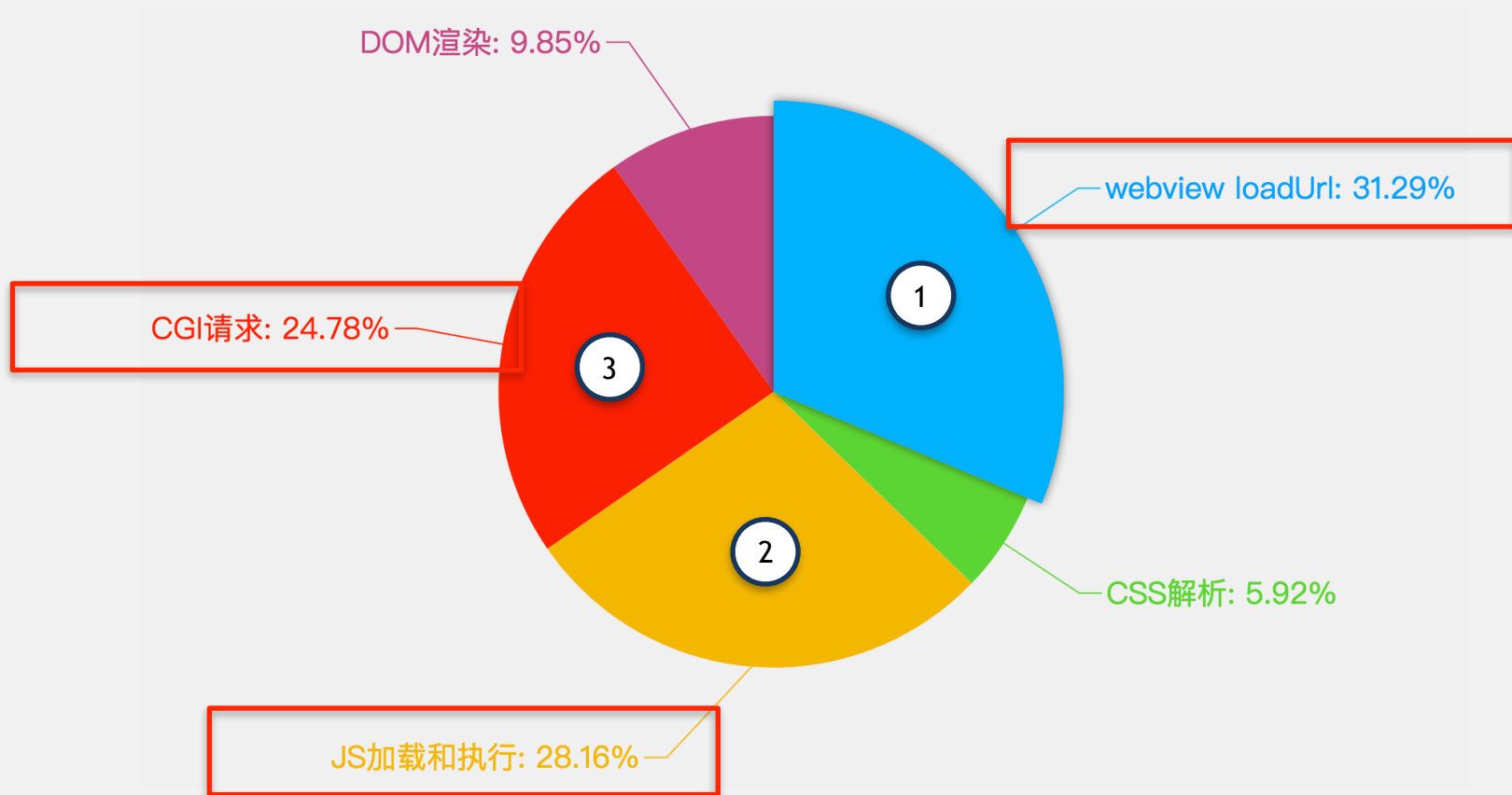
H5



用户点击到显示首屏的过程

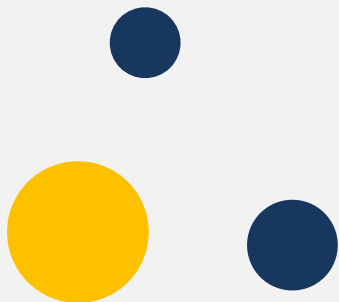


首屏耗时分布



01

优化WebView耗时



WebView优化方案



页面预加载

打开当前页面的同时，提前预加载其他页面

页面预加载

10+个页面



预加载多少个页面?

- 预加载左右N个
- $N = f(\text{cpu}, \text{内存}, \text{机型} \dots)$
- f 可以是阶段函数

什么时机开始预加载其他页面?

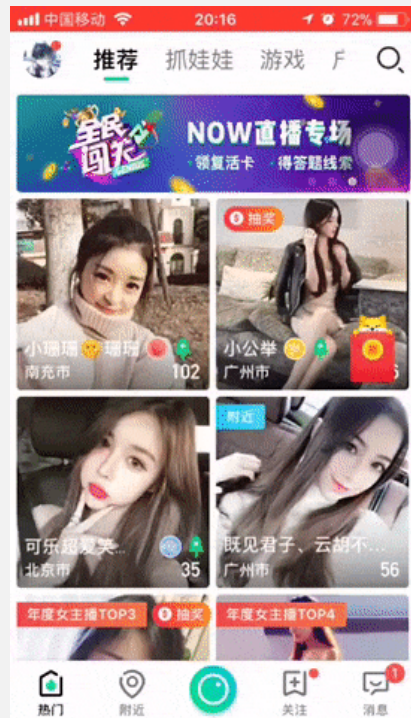
- 等待当前页面完全加载成功后，开始预加载
- 固定延时（如2s）后，开始预加载

页面预加载—效果



存在的问题

- 比较浪费用户流量
- 对机器的性能要求很高
- 对未被预加载的页面无效



WebView优化方案



页面预加载

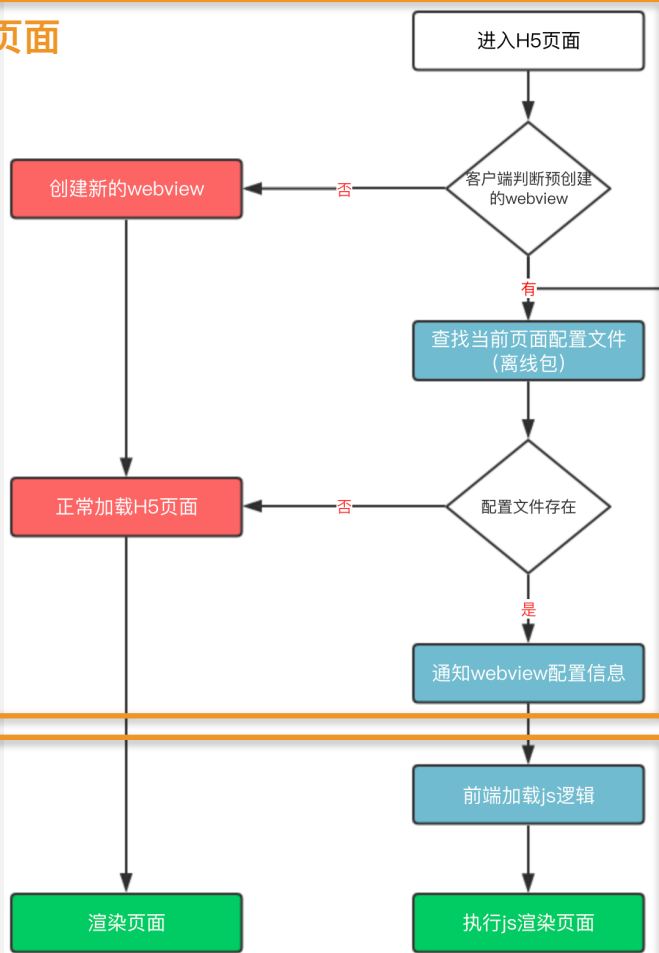
打开当前页面的同时，提前预加载其他页面



WebView H5框架预加载

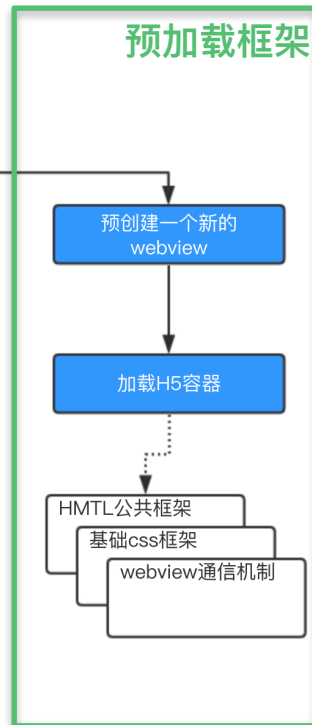
只预加载框架部分，前端异步加载js并执行业务逻辑

客户端打开页面

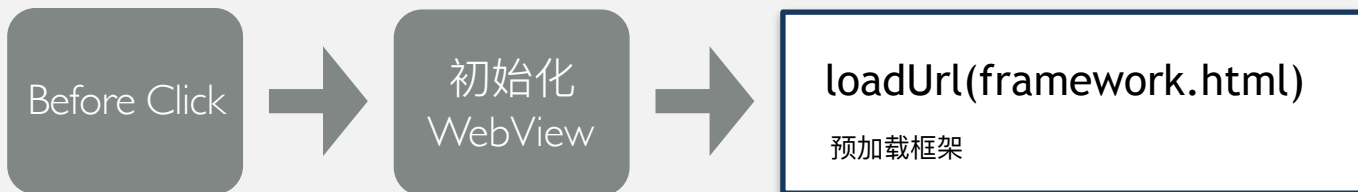


前端渲染

预加载框架



预加载框架



H5页面拆分



H5页面拆分 — framework.html

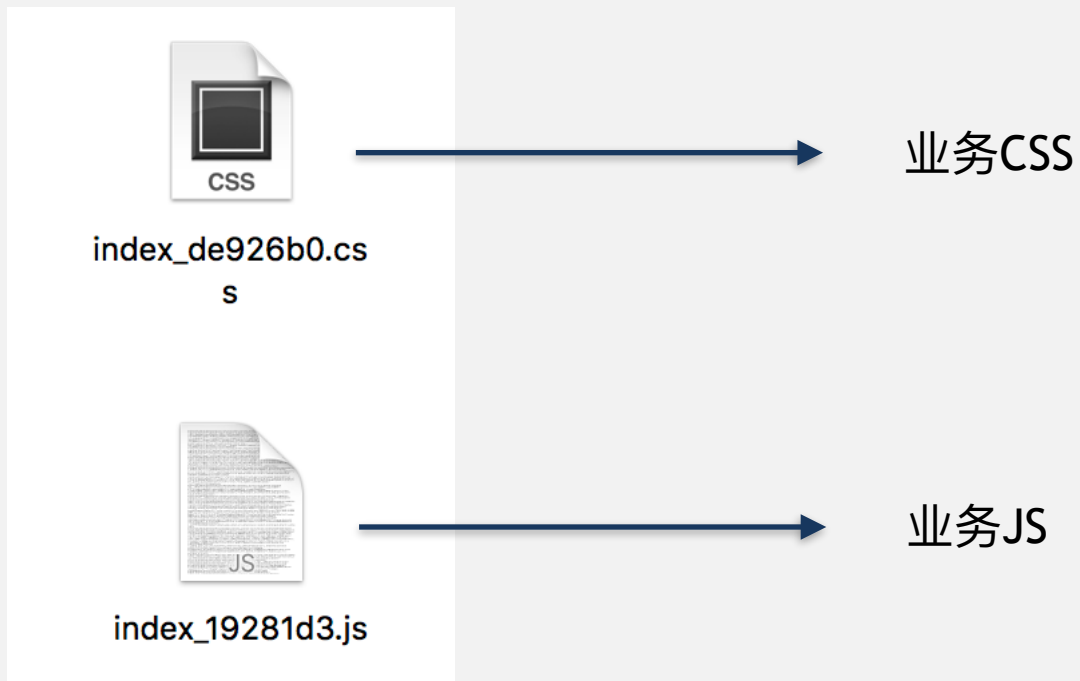
```
1 <!DOCTYPE html>
2 <html lang="zh_CN">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
6   ...
7   <title>NOW直播</title>
8   <script>
9     // 记录框架加载时间点
10    window._T = {
11      framework_page_start: new Date()
12    };
13  </script>
14  <style>
15    /* common style */
16  </style>
17 </head>
18 <body>
19   <div id="container"><!--HTML_PLACEHOLDER--></div>
20   <script>
21     window.__startLoad = function(url, config) {
22       // 1. css loader
23       ...
24       // 2. js loader
25       ...
26     }
27   </script>
28   window._T.framework_page_end = new Date();
29 </body>
30 </html>
```

1 → 公共CSS

2 → 容器

3 → 入口函数

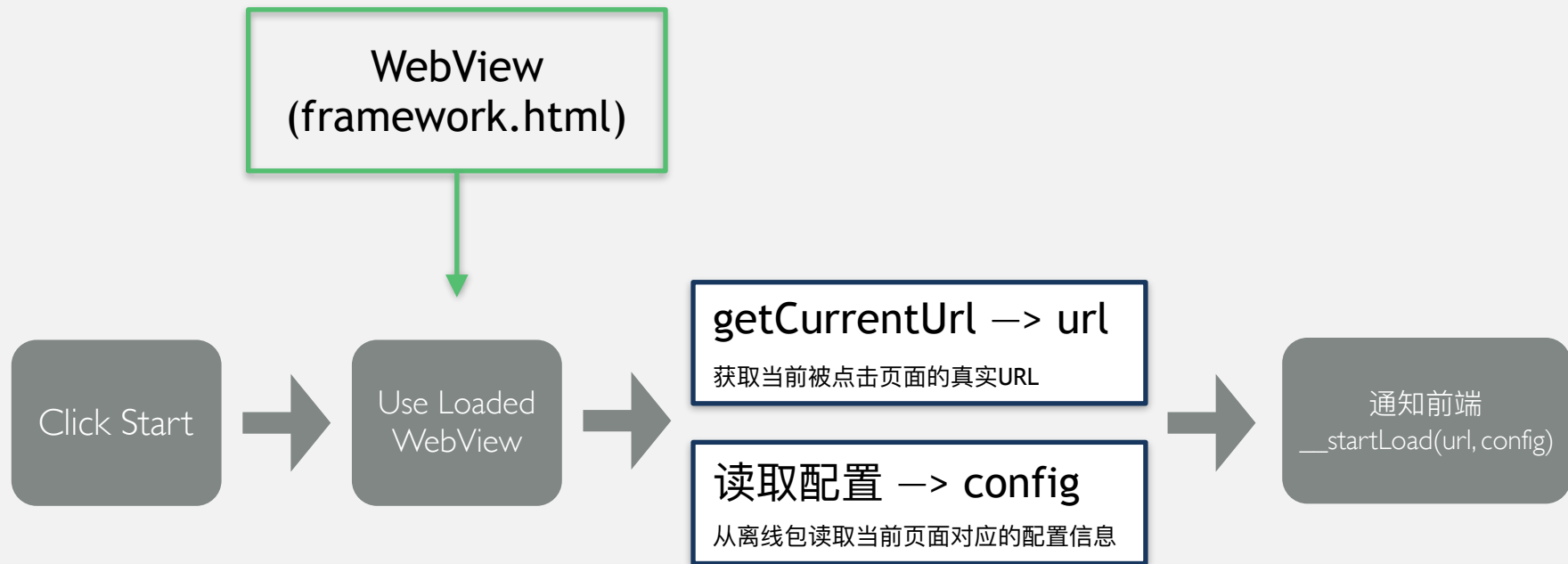
H5页面拆分 — 业务



客户端打开页面



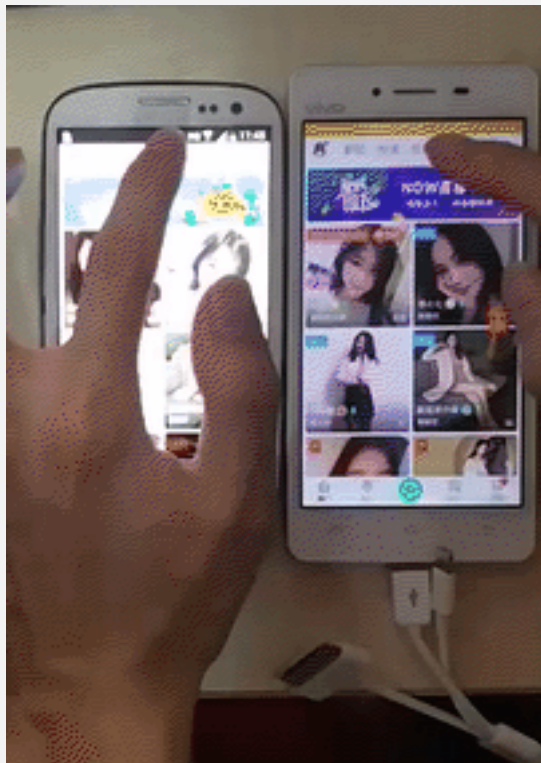
客户端打开页面



前端渲染

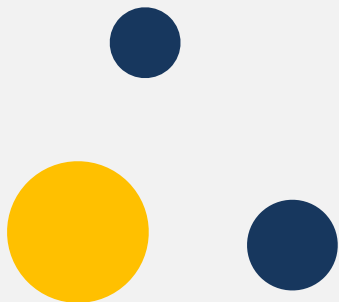


WebView H5框架预加载— 效果



02

优化JS加载&执行 耗时



优化JS加载方案



PWA

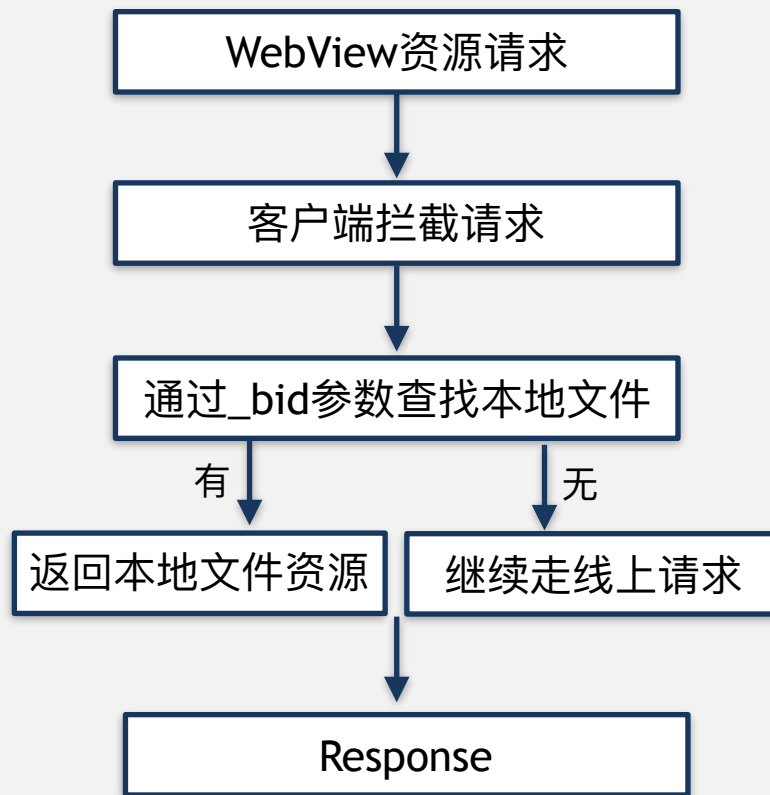
利用离线缓存的特性



离线包

静态资源APP内置

离线包机制



优化JS执行方案



优化框架

升级框架库，剔除无用模块等



优化代码

常用移动端优化手段等



优化效果一般

优化JS执行方案



优化框架

升级框架库，剔除无用模块等



优化代码

常用移动端优化手段等



前端本地缓存

将上一次的HTML片段和数据缓存到本地

前端本地缓存



前端缓存HTML + Data

缓存首屏的HTML片段和数据



页面打开时，采用本地缓存渲染

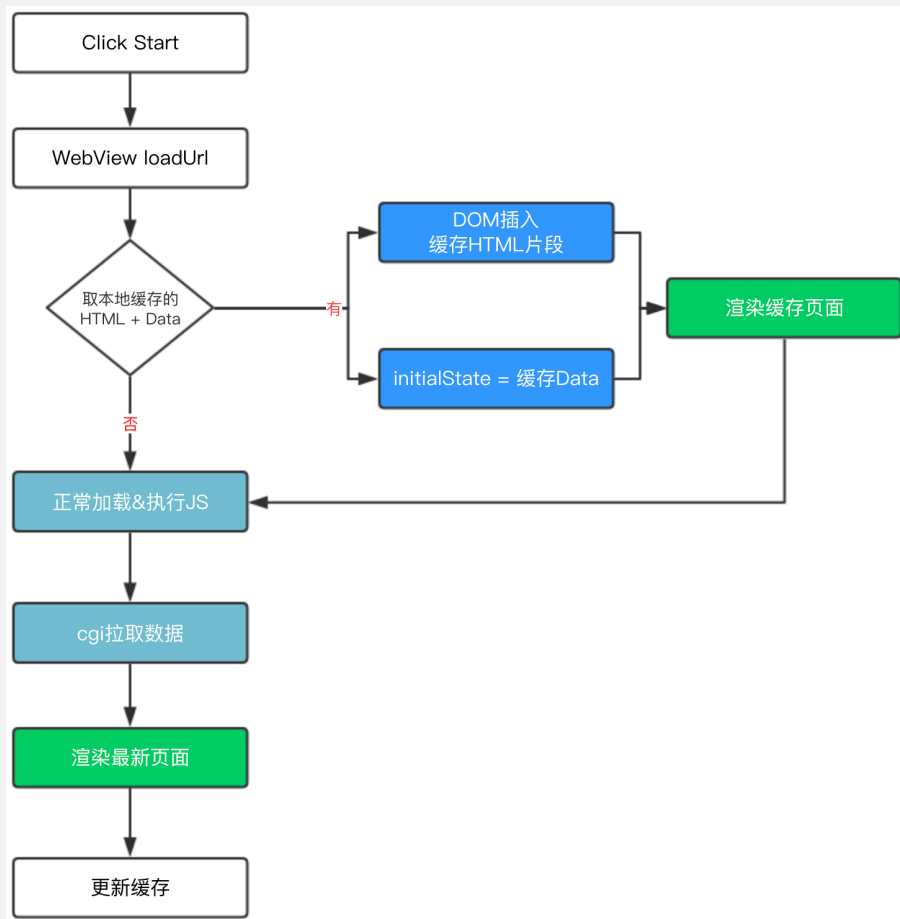
需要设置redux的initialState，否则会出现一次刷新



页面渲染后更新本地缓存

在合适的时机点更新

前端本地缓存



缓存机制：

- 客户端提供存储缓存的jsbridge (set)
- 客户端以内存和离线文件的形式存在
- 客户端提供获取缓存的jsbridge (get)

优点VS缺点



首屏时间 \approx WebView loadUrl时间

首屏完全省去JS下载和执行时间 + CGI拉取数据时间，页面秒开



不适用于数据频繁更新的场景

如直播场景等

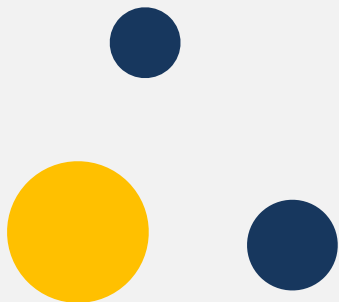


肉眼可以感知的画面重绘

由缓存HTML到最新数据render会有一次画面刷新

03

优化数据拉取耗时



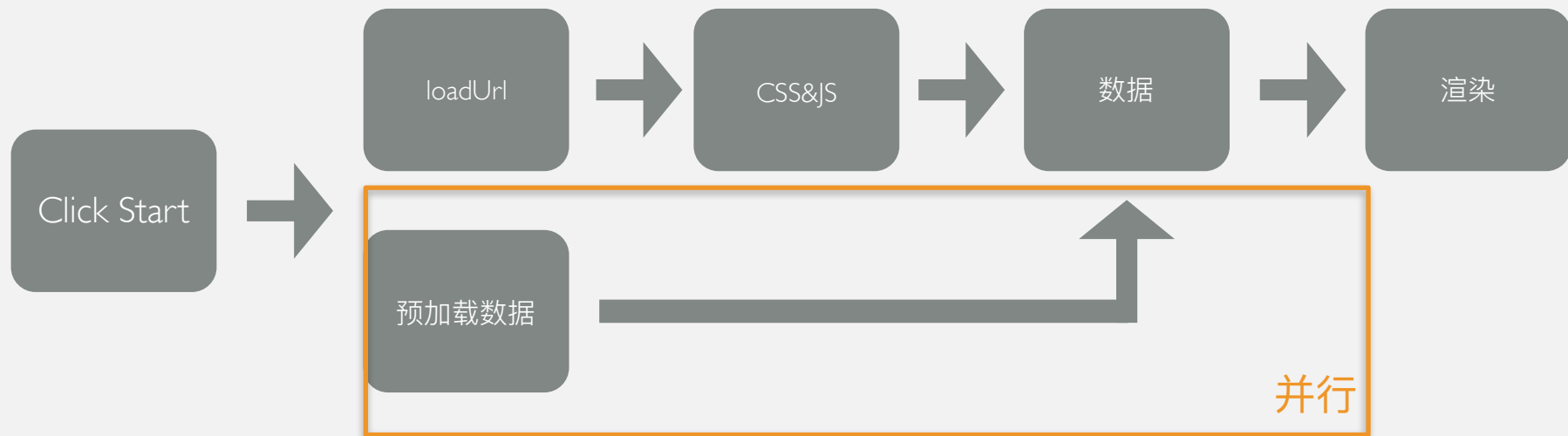
优化方案



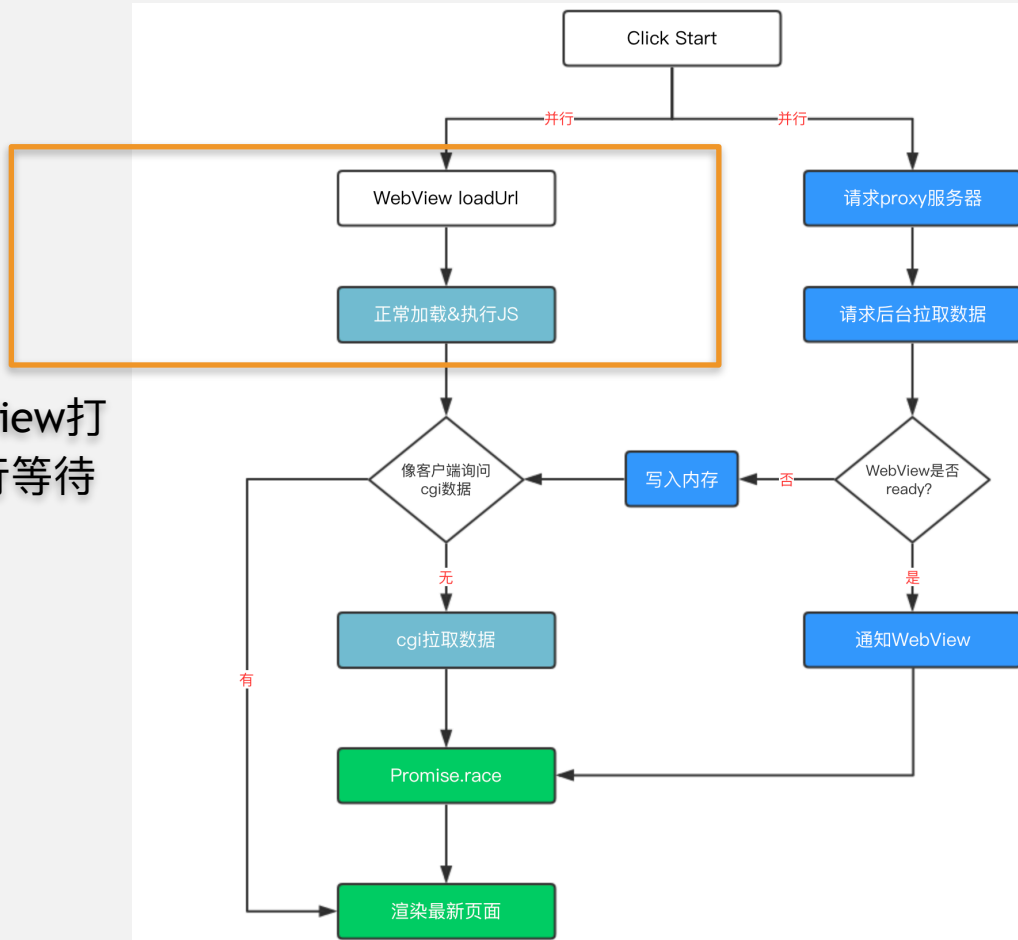
客户端预加载接口数据

客户端提前帮忙预加载接口数据

客户端预加载接口数据



客户端预加载接口数据



充分利用WebView打开和js加载执行等待的时间

小结

优化WebView耗时

页面预加载

WebView H5框架预加载

优化JS加载&执行
耗时

PWA

离线包

前端本地缓存

优化数据拉取耗时

客户端CGI预加载

谢谢聆听！

微信公众号



QQ交流群



更多干货，请扫码加入腾讯IVWEB交流群

官方Github: <https://github.com/iv-web>



Github: <https://github.com/feflow/feflow>