

移动端广告落地页预加载技术实践

张博

字节跳动 前端工程师

InfoQ官网 全新改版上线

促进软件开发领域知识与创新的传播



关注InfoQ网站
第一时间浏览原创IT新闻资讯



免费下载迷你书
阅读一线开发者的技术干货

自我介绍



2018 年加入字节跳动，主要负责商业化广告制作、展示效率等方面工作。

曾就职于百度，先后负责和参与了百度脑图、百度 H5 等产品，在 Web 富交互应用、前端性能优化方面积累了一定的经验。

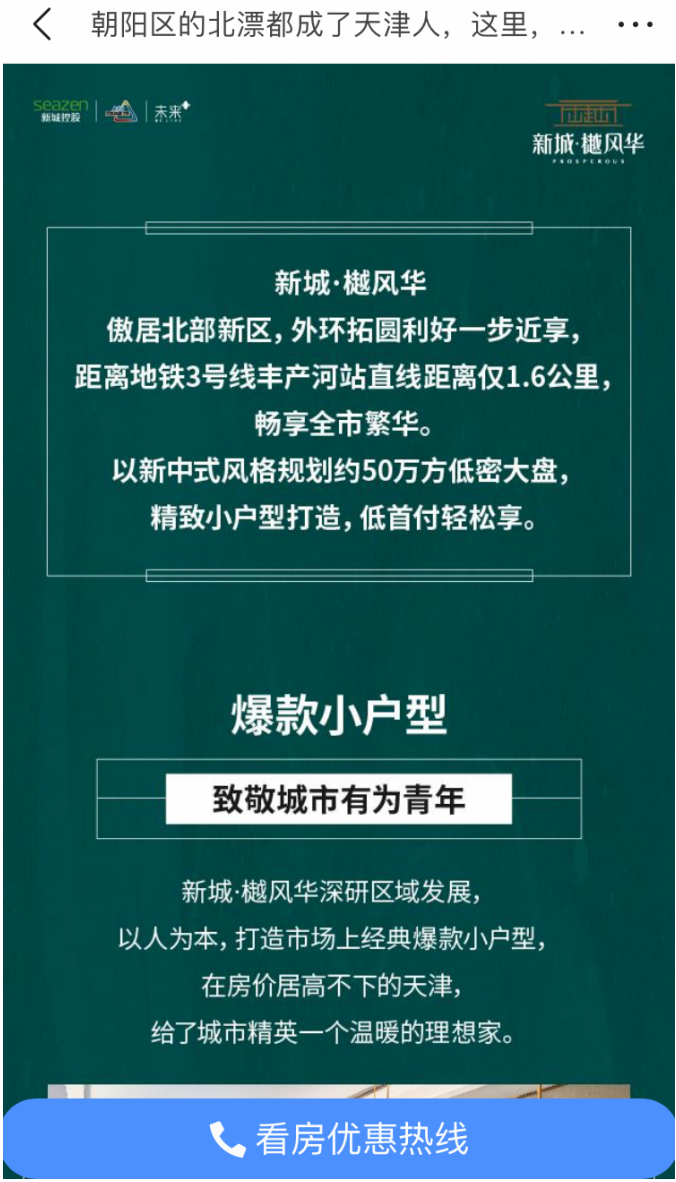
目录

- 背景介绍
- 预加载技术与实践
- 预加载的机遇和挑战

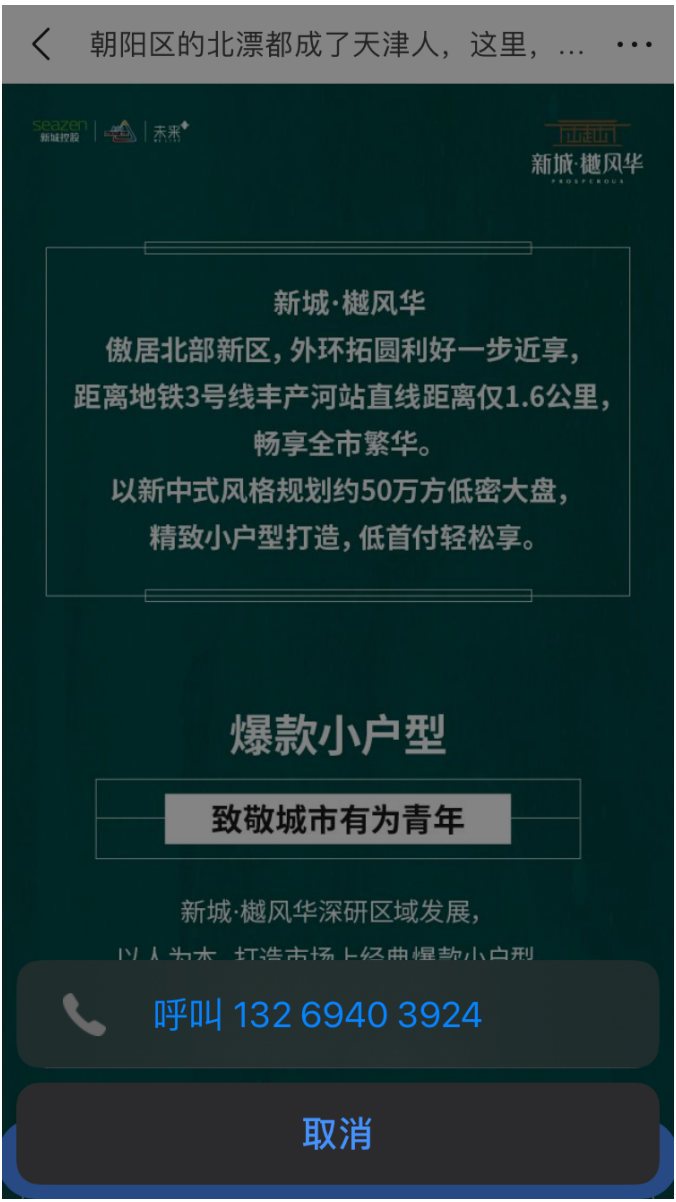
背景介绍



创意



落地页



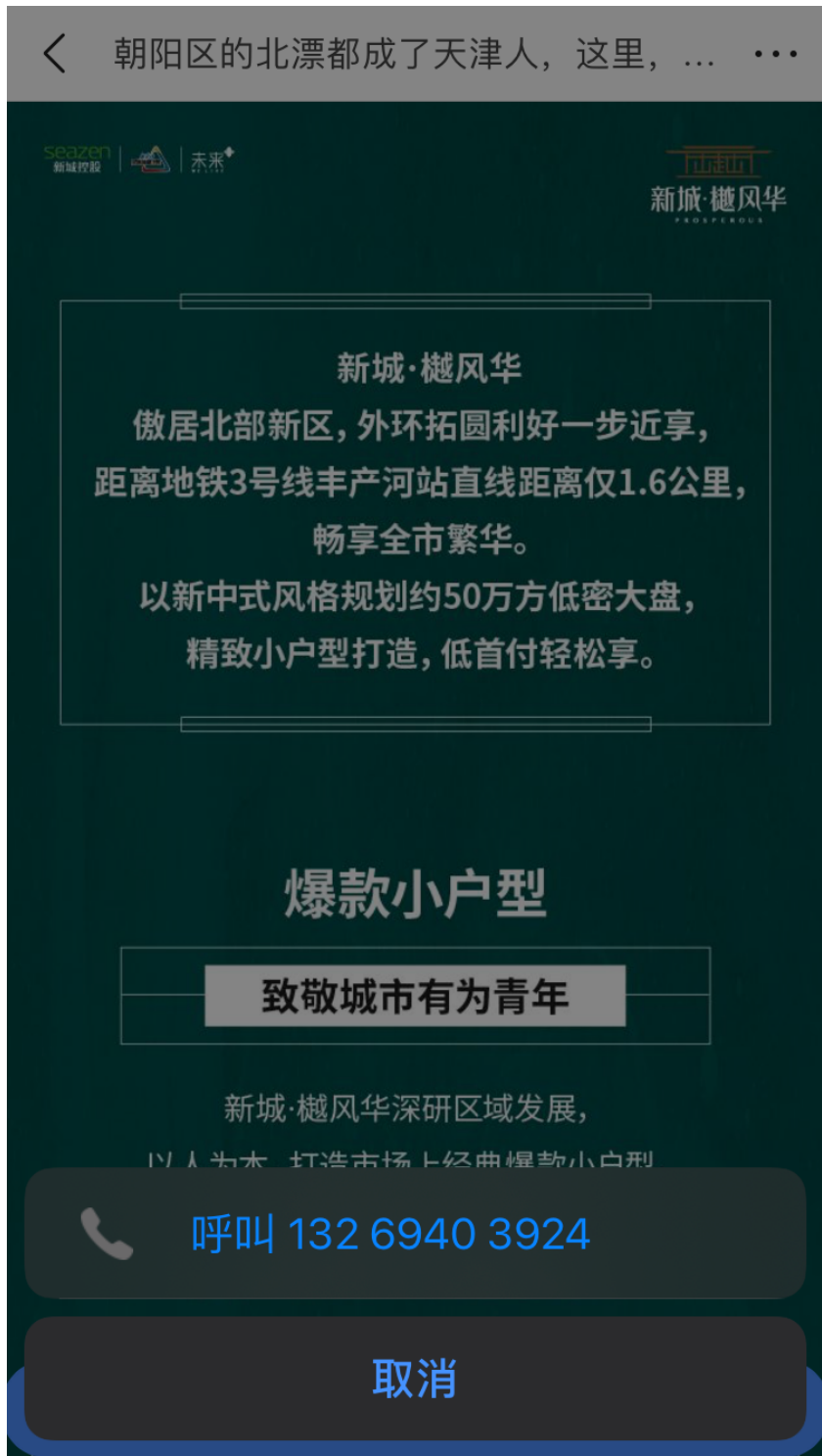
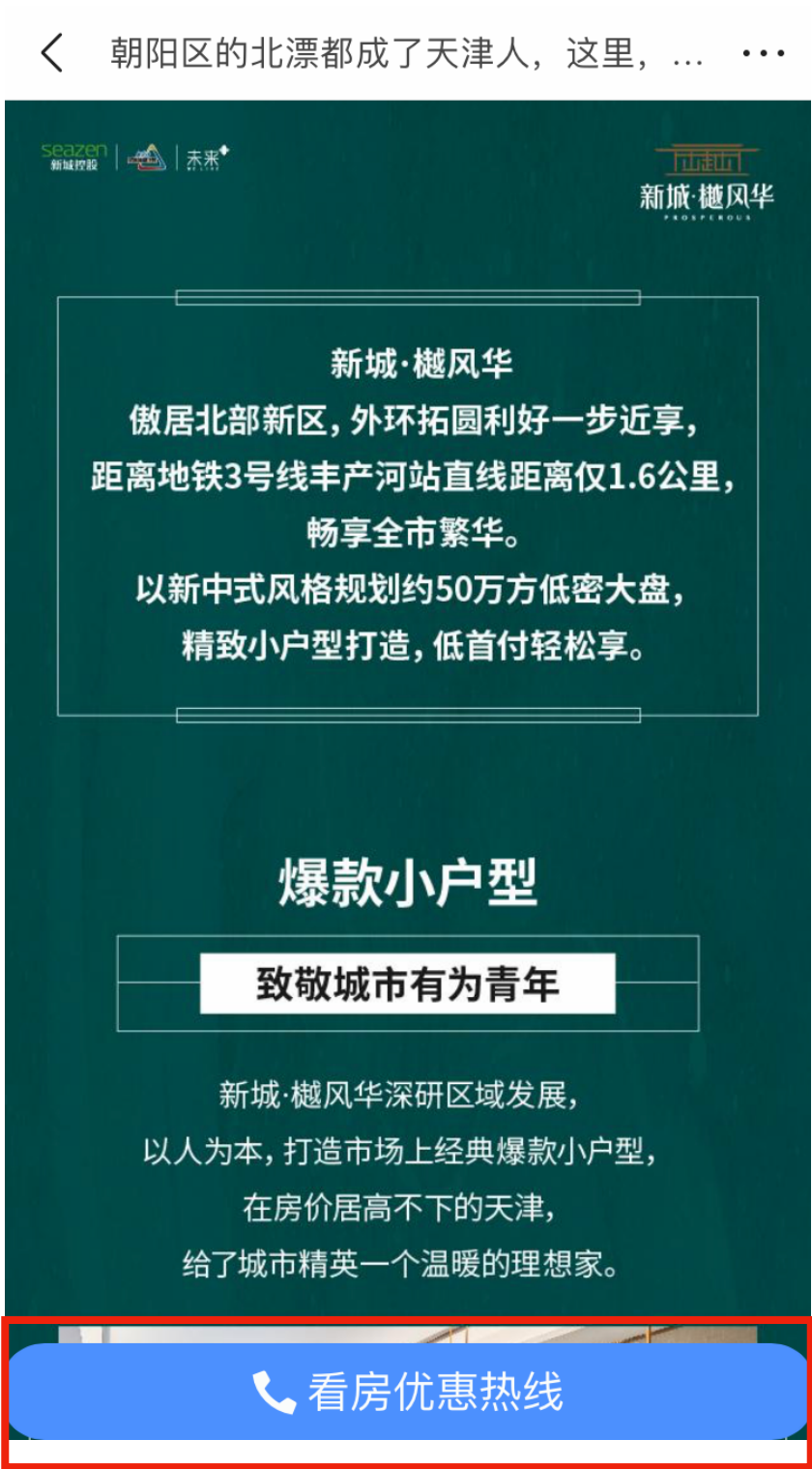
转化：用户和广告交互后，发生了业务上有意义的行为（购买了商品、拨打电话等）

点击率 = 点击次数 / 展示次数

转化率 = 转化次数 / 点击次数

eg. 创意有 1w 次展示，有 2k 次点击，有 100 次转化，点击率 = 2k/1w = 20% 转化率 = 100/2k = 5%
点击和转化等数据涉及计费，所以埋点准确性非常重要，这也是广告落地页相比于普通网页的不同。

预备广告知识



落地页主要有两种形式：

- 1. Native：端上提供模板，下发数据做拼合。体验好，但是不够灵活。
- 2. Web：页面由 HTML 实现。足够灵活，但是可能存在性能问题。

背景介绍



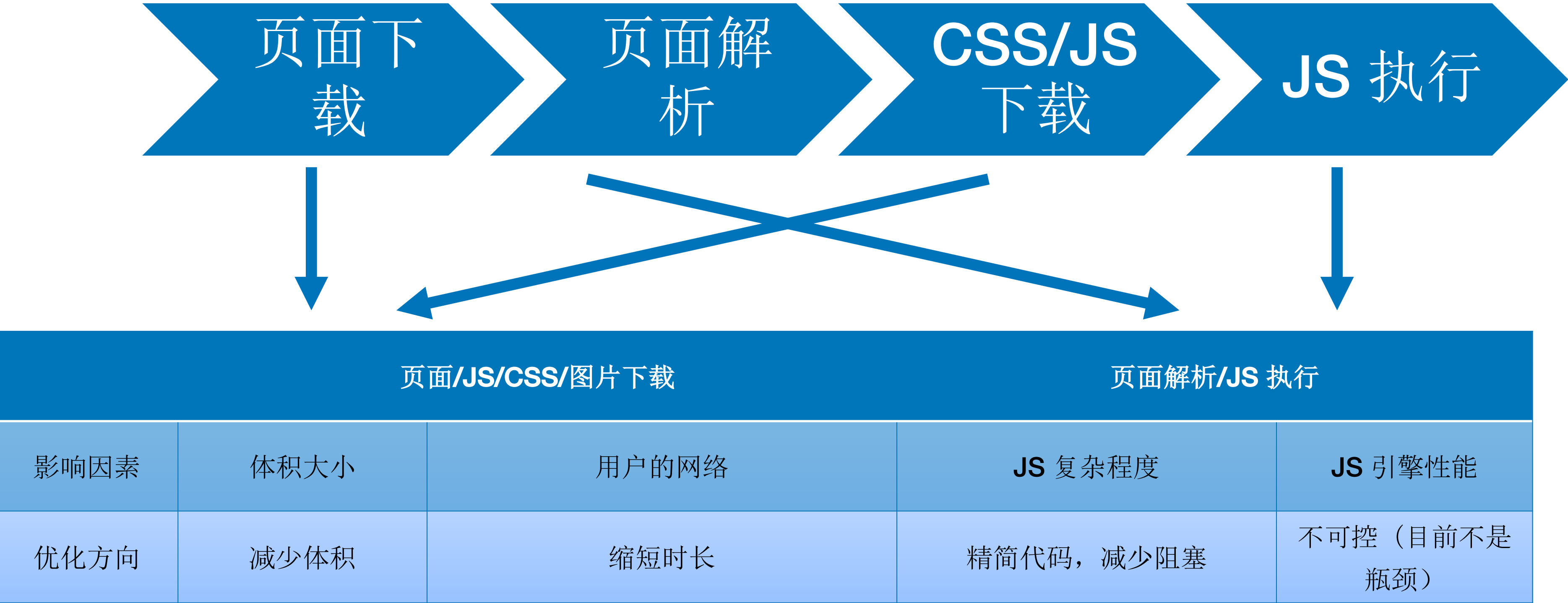
加载时长在 0 – 1.5s 内，转化率都在 15% 以上。

加载时长超过 1.5s 的落地页，转化率变化不大（后面的数据波动是由于样本数据较少）。

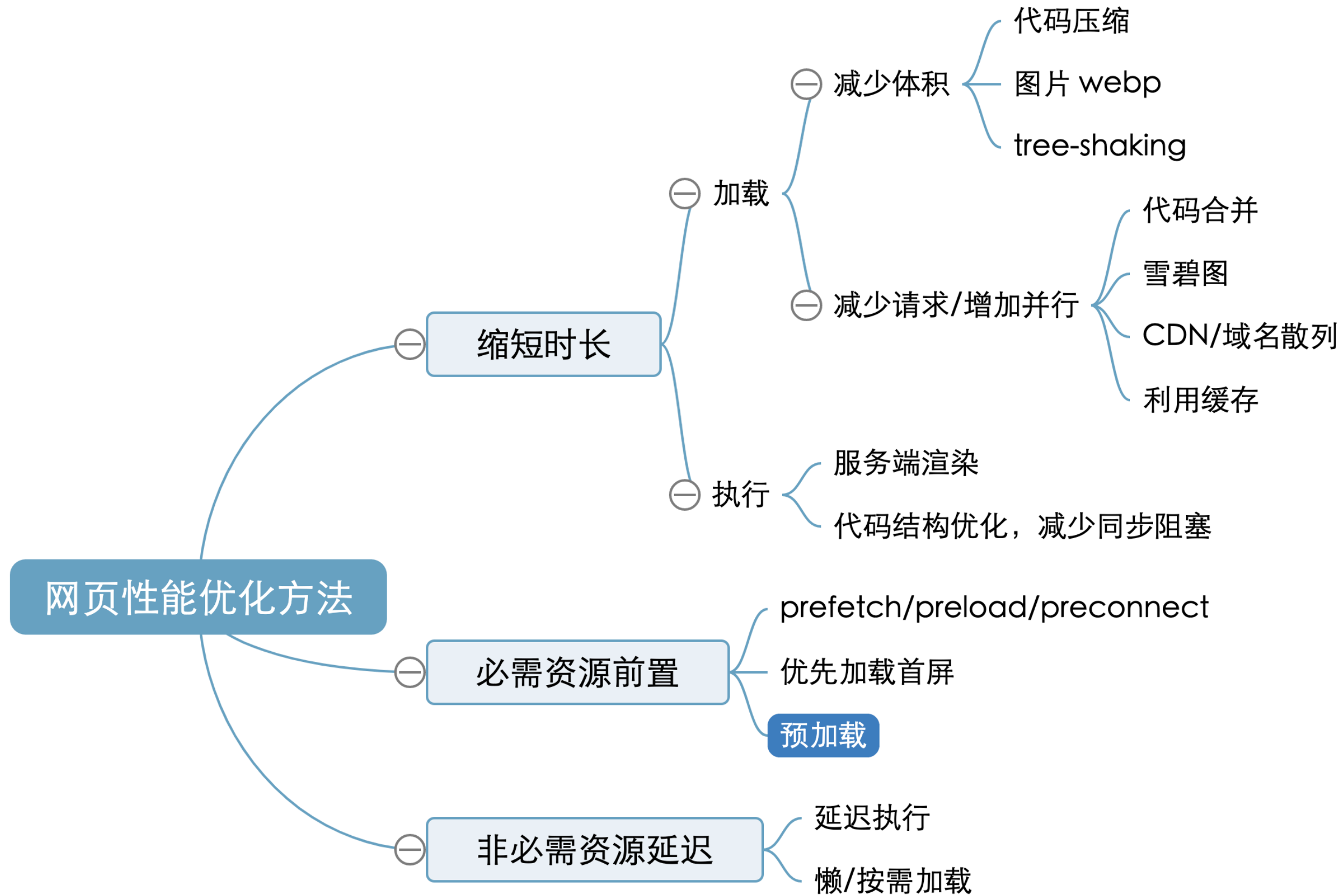
优化落地页加载时长到 1.5s 内，能显著提升落地页转化率。

广告落地页性能优化

广告落地页也是移动端的网页，常规的优化手段可以使用。



广告落地页性能优化



常规的优化做完，再想进一步优化，就需要考虑广告落地页的自身特点
广告落地页运行在固定的环境内（字节系 App），我们称之为「端能力」。

广告落地页的预加载

字节跳动的广告落地页分为：

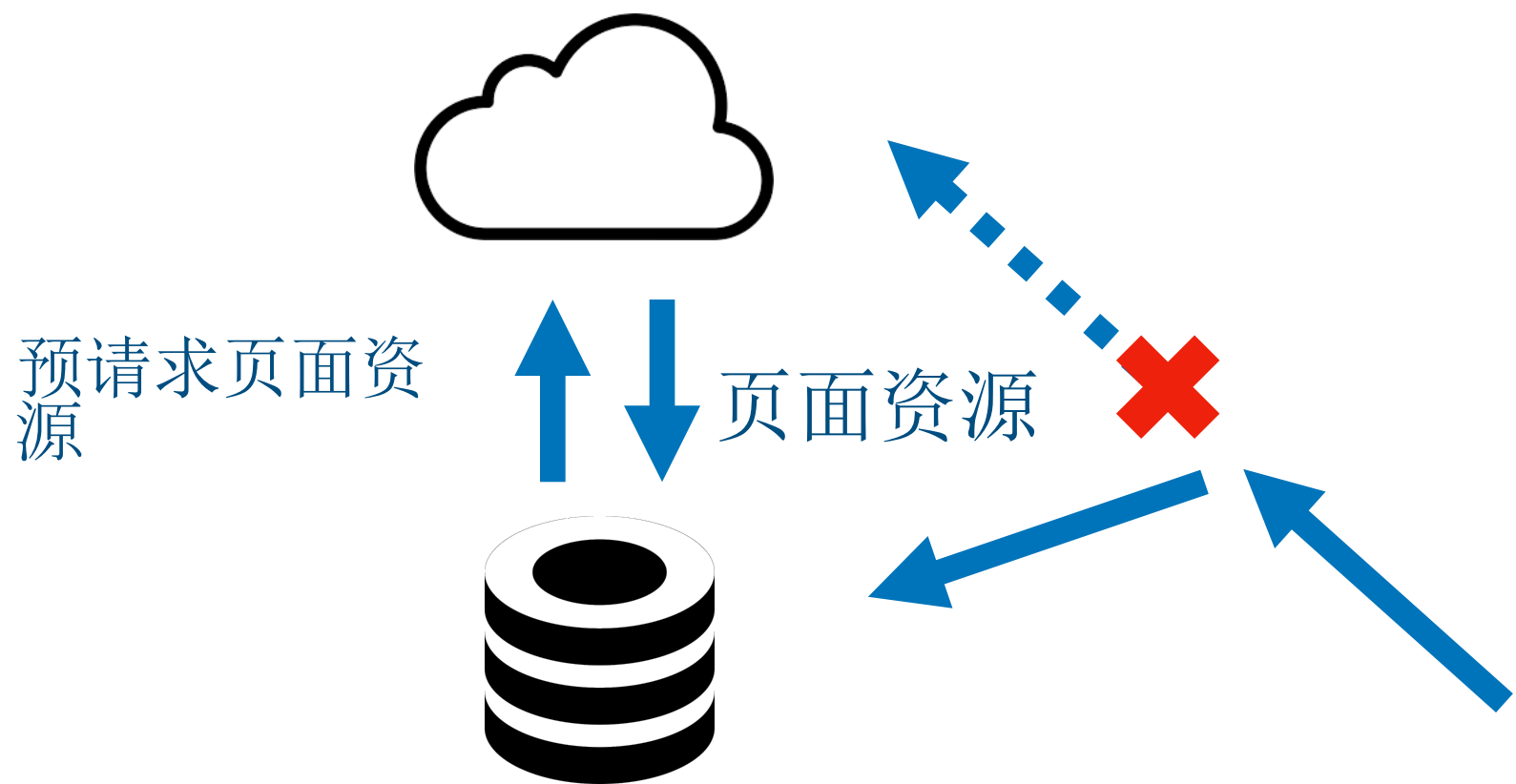
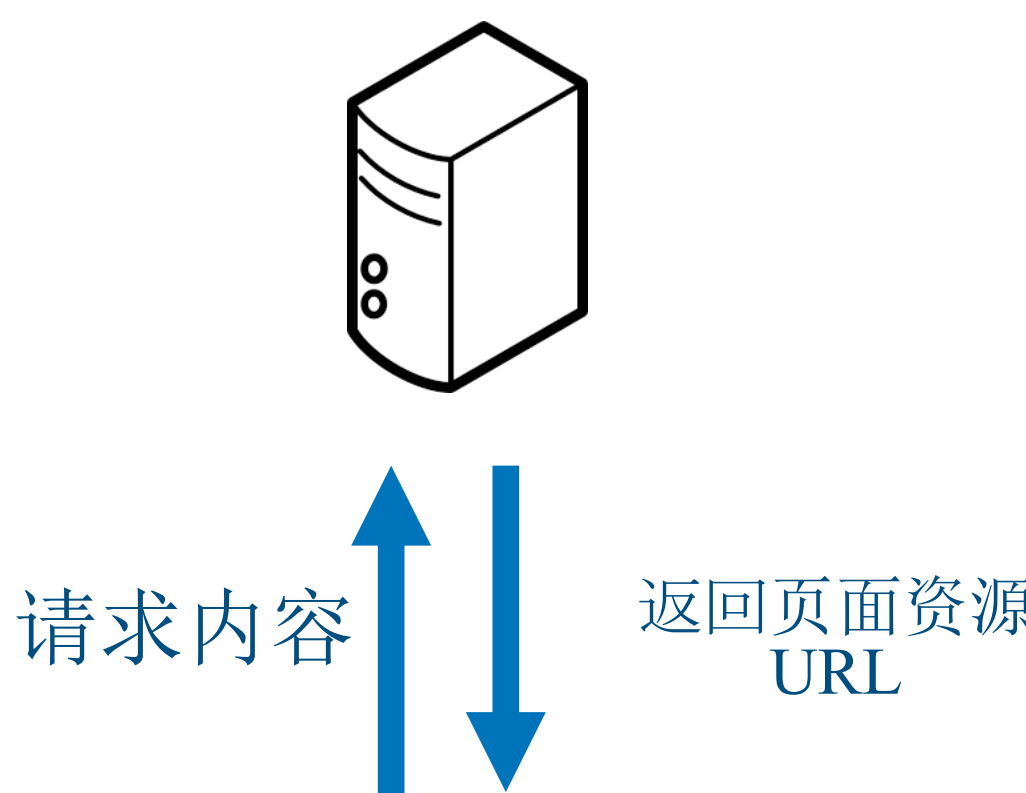
- 1. 自有落地页：通过橙子建站制作的广告落地页
- 2. 第三方落地页：广告主自己制作或者找代理商制作的广告落地页

对比项	自有落地页	第三方落地页
制作方式	拖拽生成	编码
技术栈统一程度	统一	各异
可控程度（服务器资源）	完全可控	不可控
端能力协同	支持	少部分

结合端能力和落地页情况，我们将小目标定为：自有落地页的预加载

预加载技术

预加载原理



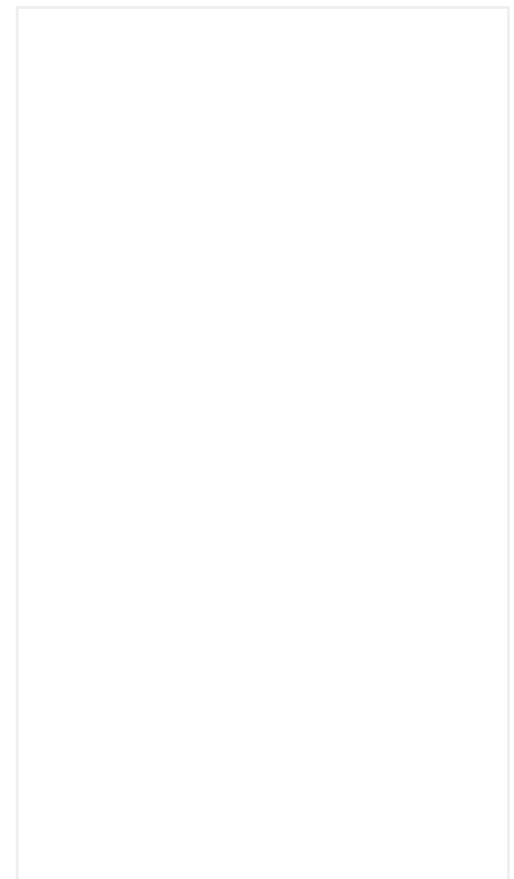
关键两步：
1. 预加载
2. 打开时拦截



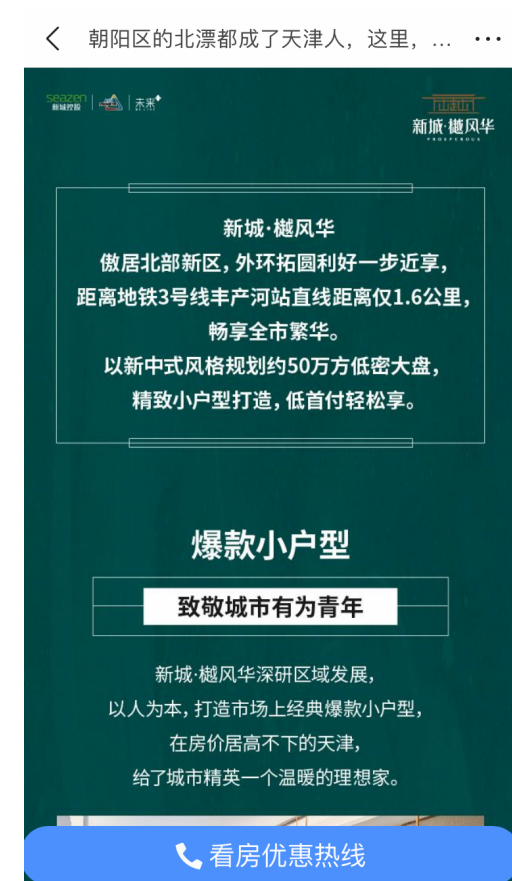
双击刷新



用户浏览



点击创意



广告呈现

基于资源列表的预加载

1. 橙子建站的每个落地页发布时，生成对应的 `preload_resouces.json`
2. 用户刷新 feed 时，端上根据得到的资源列表，依次下载资源
3. 用户点击创意
4. 端上拦截 HTTP 请求，检查是否在本地已存在，若是，则直接返回。

预下载：端上根据资源列表下载

拦截：打开时，端上拦截请求并返回本地已有资源

基于资源列表的预加载

Url : 资源地址

Size: 资源体积

Content-type: 写入格式

Level: 可缓存级别, 1 为通用资源, 2 为独特资源

Screen: 图片在第几屏

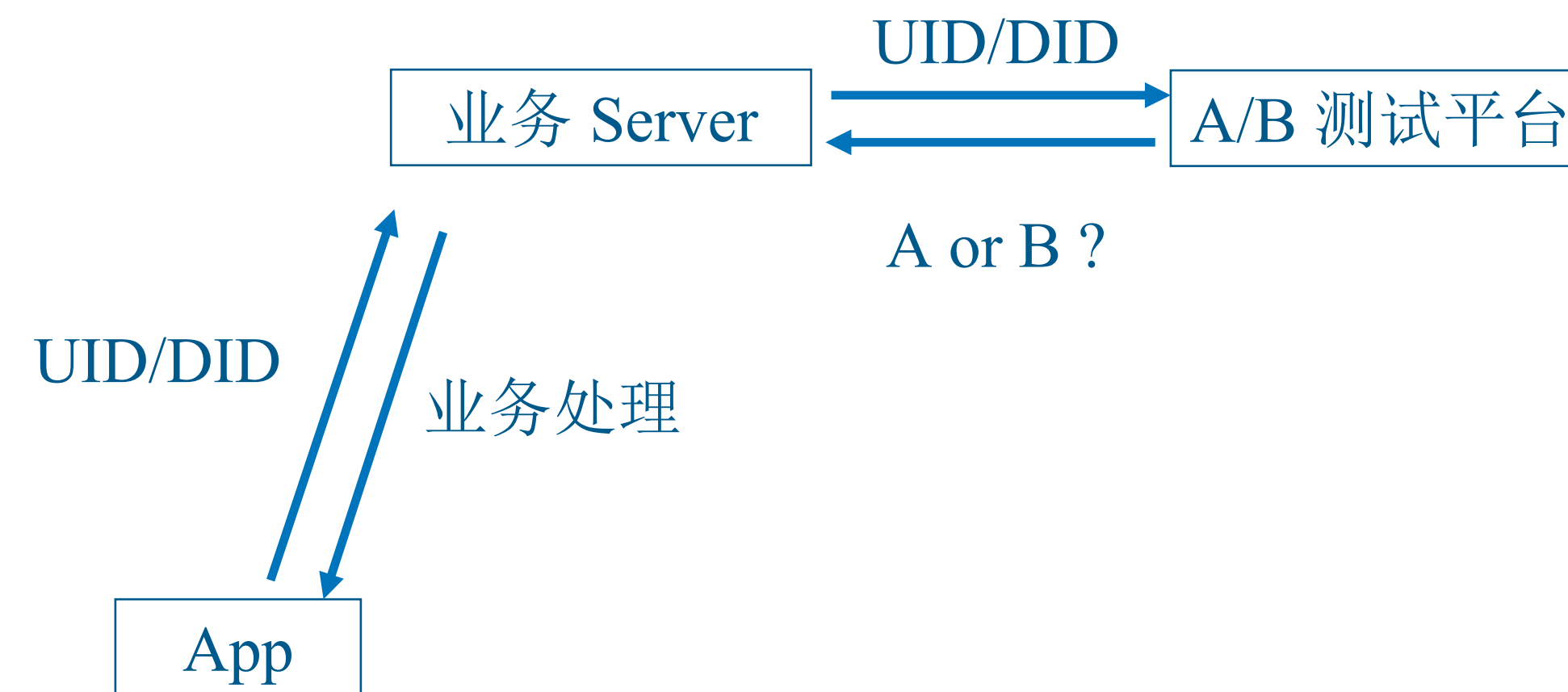
```
[
  {
    "url": "https://xxx.com/tetris/page/<id>",
    "size": 2334,
    "content-type": "text/html",
    "level": 2
  },
  {
    "url": "https://<cdn>/vendor.af23.js",
    "size": 5233,
    "content-type": "application/javascript",
    "level": 1
  },
  {
    "url": "https://<cdn>/xxx.jpeg",
    "size": 234343,
    "screen": 2,
    "content-type": "image/jpeg",
    "level": 2
  }
]
```

preload_resources.json 示例

“字节跳动是个只会 A/B Test 的公司”？

A/B 测试基础知识

使用一部分总流量如（20%），基于用户（User ID）或者设备（Device ID）进行分流，分为实验组和对照组。



基于资源列表的预加载 A/B 测试方案

实验组和对照组：

对照组：不进行预加载 vid = A

实验组：实施预加载的 vid = B



基于资源列表的 A/B 测试方案数据结果

指标:

- 1. 技术指标:
 页面加载时长（实际加载时长）、预加载完成率（预加载是否充分）
- 2. 业务指标:
 转化数、转化率、广告主价值

网络	4G	WiFi
预加载时长	1.62s - 2.47s	1.21s - 1.72s
网页加载时长	3.9s -> 3.0s	3.4s -> 2.8s
转化数	增加近 4%	小幅波动
广告主价值	最高提升近 5%	小幅增加

结论:

- 1. 技术指标:
 页面加载时长都有不同程度的缩短 10% - 20%、预加载完成率 99% 以上
- 2. 业务指标:
 转化数和广告主价值一定程度提升，转化率提升比较有限

基于资源列表的预加载方案总结

优点：

1. 对于资源的控制把控力度强（何时何种条件预加载），能清晰还原状态
2. 不依赖于外部特性（cache header 等），直接可用

缺点：

1. 开发成本比较高，特别是埋点方案。开发->测试->发版->灰度->实验，过程至少1个月。

基于 Webview 的预加载原理

核心是依赖于浏览器的缓存机制，手段是借助 prefetch/preload 实现

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Loader HTML</title>
    <link rel="prefetch" href="<cdn>/example.js" as="script" />
    <link rel="prefetch" href="<cdn>/example.css" as="style" />
    <link rel="prefetch" href="<cdn>/example.jpg" as="image" />
  </head>
  <body></body>
</html>
```

tips :

1. prefetch 是获取下一个页面要用到的资源，preload 是获取本页面用到的资源，优先级更高。
2. iOS 不支持 prefetch，可以考虑使用 preload 替代；
3. iOS 11.2 及以下不支持 preload，可以考虑使用 ajax 或 fetch，不过会涉及跨域问题

* 参考：<https://zhuanlan.zhihu.com/p/77144845>

基于 Webview 的预加载

```
<html lang="en">
<head>
  <meta charset="UTF-8"><title>Loader HTML</title>
  <link rel="prefetch" href="<cdn>/example.js" as="script" />
  <link rel="prefetch" href="<cdn>/example.css" as="style" />
  <link rel="prefetch" href="<cdn>/example.jpg" as="image" />
</head>
<body></body>
</html>
```

loader.html

点击



```
<html lang="en">
<head>
  <meta charset="UTF-8"><title>Landing Page</title>
  <link rel="stylesheet" href="<cdn>/example.css">
  <script src="<cdn>/example.js"></script>
</head>
<body>
  
</body>
</html>
```

落地页

实验过程:

1. 橙子建站发布时, 每个落地页生成对应的 loader.html
2. 刷新广告时, 若在实验组里, 初始化一个不可见的 webview 加载 loader.html
3. Webview 会在空闲时下载 loader.html 里面写明的资源
4. 用户打开广告, webview 基于资源的 response header 判断是走缓存还是网络。

预下载: 利用 HTML 自身机制下载

拦截: webview 利用浏览器的缓存策略「拦截」

基于 Webview 的预加载方案总结

优点：

1. 开发成本比较低
2. 如果 Webview 复用，能做到 cache from memory
3. 「可伸缩性」较好，很容易地支持从单 Case 到规模化产出的落地页

缺点：

1. 对于资源把控力度较弱，如开始预加载时机、当前的状态以及过程中埋点
2. 依赖于浏览器机制，不符合条件的资源需要改造 HTTP Header

方案对比

两个方案各有优缺点，对比如下：

对比项	基于资源列表	基于 Webview
开发成本	大	小
对于资源的把控度	强	弱
I/O 速度	Disk Cache	最快可以 Memory Cache
对端能力要求	高	低

其他相关的方案

代码打包到 App：提前将需要用到的资源打包到 App 的安装包中，可以做到页面彻底的「离线化」，很多 Hybrid App 在使用。

预渲染：刷新广告时，初始化一个不可见的 webview，直接加载落地页。用户点击打开时，将 webview 设置为可见。

四个方案各自的特点以及渐进式

预加载方法	打包到 App	基于资源列表	基于 webview	预渲染
特点	简单，但是要做拦截	复杂，需要处理各种情况	简单，基于 HTML 机制	简单粗暴
问题	随包发版，不够灵活	开发成本高，需要处理缓存一致性问题	对资源把控度不够，加载细节和进度不可控	初始化需要运行的脚本以及音/视频自动播放等



资源把控度减弱，开发成本减少

预加载的机遇和挑战

预加载的机遇和挑战

问题：如何比较快的推进预加载项目？

解决：小步快跑，规模化成本较高，步子不用迈大、可以从单个页面、单个广告位先开始。

eg. 先用少数页面做实验，评估出可取得的收益，然后衡量 ROI

预加载的机遇和挑战

问题：广告位形态各异，预加载方案能否通用化？

因地制宜，围绕广告位的特点进行设计，同时可以多种方案结合

eg. 开屏广告位 vs. 信息流广告位

eg. 通用性资源打进 App，个性资源使用 webview 预加载

预加载的机遇和挑战

问题：如何找到资源消耗和收益的平衡？

对资源消耗和收益做精确量化，同时利用技术手段进一步优化资源消耗。

eg. 结合广告行为预测，对预估点击率高于一定阈值的才进行预加载，并且反馈修正预测模型。

预加载的机遇和挑战

问题：第三方落地页如何做预加载？

解决：可基于现有方案扩展，额外需要解决第三方落地页服务器抗压能力、资源变动不可控以及干扰检测埋点的问题。

eg. 提前抓取第三方落地页资源，并定时更新。干扰埋点可以增加特殊标识并告知广告主。

总结

- 预加载技术是结合「端能力」的全新尝试，是解决落地页性能问题的「杀手锏」。
- 预加载关键词：「预下载」和「拦截」，所有预加载都是如此；
- 基于资源列表的预加载：通过资源列表，端上完成下载和拦截；
- 基于 webview 的预加载：通过 loader.html，webview 完成下载和拦截；
- A/B 测试是一种科学的验证策略优劣的方法；
- 预加载的机遇和挑战：小步快跑、因地制宜、精确量化、扩展范围。



关注 QCon 公众号

收获国内外一线大厂实践 与技术大咖同行成长

✓ 演讲视频 ✓ 干货整理 ✓ 大咖采访 ✓ 行业趋势





扫码查看

60 天学透算法与数据结构

600+ 名企内推通道向你打开！

✓ 大厂内推 ✓ 实战演练 ✓ 闭环体系 ✓ 社群连接

THANKS

