阿里文娱技术 | 阿里云 开发者社区

小程序 大世界

淘票票小程序进阶之路

2年接入6大主流应用平台

从开发技巧到避坑指南全掌握

藏经阁

阿里云开发者电子书系列

阿里云 开发者社区

加入交流群

阿里巴巴文娱技术
钉钉交流群

关注我们

阿里巴巴文娱技术公众号

更多独家电子书

阿里云开发者“藏经阁”
海量免费电子书下载

目录

第一章 淘票票小程序的进阶之路

作者 | 阿里文娱高级技术专家 丰安

小程序异军突起

2018 年是小程序爆发的一年，从国内手机厂商到 BAT，再到新兴的头条抖音，纷纷推出或者推广自家的小程序平台。为什么各家如此热衷小程序？每个平台的特点是什么，价值在哪里？这么多小程序我们该如何涉足？相信大家或多或少都有过这些疑问。

在 2019 年，阿里巴巴文娱的淘票票几乎涉足了当时市面上所有的小程序，上线了**支付宝小程序、手淘轻应用、快应用、字节跳动小程序、百度小程序以及微信小程序**。其中在不少平台上，我们是阿里第一批吃螃蟹的技术团队。回顾过往，我们做过很多尝试，也踩过很多坑，所以我们整理出系列文章，把技术经验分享出来。

就像标题写的那样，小程序里有着大世界！希望为你带来启发。

小程序核心特点——场景融合

小程序跟传统 H5 有什么区别？从产品和业务角度来说，小程序的核心特点在于“**场景融合**”。无论是手机厂商，还是支付宝、手淘、微信、头条、抖音、百度等第三方应用，每个平台都有着既定的用户和使用场景。例如，支付宝就是生活服务类场景，头条是内容资讯类场景。每个小程序要思考的，就是如何将自己的产品功能嵌入到平台的使用场景中，实现场景融合。对于小程序本身来说，能够借助于平台的流量和用户做大做强；对于平台来说，则可以借助各种小程序，丰富使用场景，增加平台的用户粘性，最终实现双赢的结果。这也解释了为什么各家都如此热衷于小程序平台的搭建。

场景定制

场景融合另一个层面意味着场景定制，支付宝跟抖音的产品定位跟使用场景差别很大，各自平台上的小程序也需要有所差异。这也给我们产品技术层面带来了很大的挑战。如何通过有限的人力，同时支持这么多小程序成了我们面临的重要问题。对此，我们主要做了这么几件事：

1) 区分出各个渠道的优先级和重要性，确定重点投入的渠道。2018 年我们更多侧重于支付宝、手淘这两个渠道，在资源方面有所倾斜；2019 年我们则重点融合头条抖音、百度、微信等渠道；
2) 将主要渠道和次要渠道做了归类，产品形态上进行聚合，减小差异。例如，在支付宝和手淘这两个主要渠道上，我们主要是对标淘票票客户端，提供相对丰富的功能和玩法；而在其他渠道上，我们做了功能精简，仅保留核心的购票功能，并且产品形态上趋同；
3) 技术架构方面，对各个平台的 DSL 进行分层优化，将业务逻辑跟界面展示分离，抽取出通用业务逻辑和 Util 方法，提高代码的复用度。之后的技术篇会有详细的介绍；
4) 为了缩短开发周期，采用 DSL+H5 混合应用的方式开发，将 UV 较高的页面用 DSL 实现，以便提供较好的用户体验；UV 较低的页面采用原有的 H5 页面，以便减少开发量。后续将根据实际情况，调整 DSL 和 H5 的比例；
5) 终端全栈模式，打破传统 Android、iOS、H5 技术分工的界限，让 Natvie 同学也能够开发小程序。小程序整体语法较简单，学习难度相比传统 H5 较低。实际开发中，更有两位服务端同学通过两周的学习和培训，也参与到小程序的开发中。

场景运营

上线了这么多平台的小程序，我们有个很深切的体会：**小程序相比传统 App 需要更多的细分场景运营，需要产品技术和运营一起探索分场景的运营策略。**例如，通常我们会通过活动、优惠等形式做运营推广，以便增加流量。但是我们发现在头条和

抖音平台上，主要流量不是来自于固定入口，而是来自于优秀内容推荐。好的内容往往会带来比平时多数倍甚至数十倍的流量。因此，在头条抖音平台上，内容运营的重要性就大大增强了。

另外，由于每个渠道小程序平台能力和成熟度不一样，实际工作中会遇到诸多问题。因此在团队组织上，可以采用业务、产品、技术“铁三角”的小闭环模式，小步快跑，快速调研，快速反馈，快速上线。这样在小程序上线初期能够极大地提高整体效能。

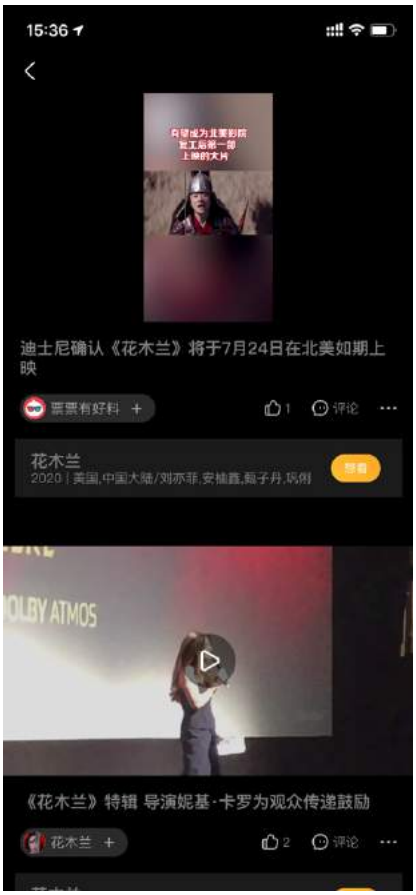
覆盖 6 大热门平台的淘票票小程序

回顾淘票票的小程序矩阵的搭建过程，我们走了一条“进击之路”。接下来我会

简单介绍淘票票在各个平台上小程序，之后的系列文章会详细介绍每个平台的场景、实现、遇到的问题以及沉淀的经验。

支付宝小程序

在支付宝平台上，淘票票从最初 Natvie 版本、H5 版本，最终转变为小程序版本。支付宝的生活服务类场景跟淘票票是最契合的，平台的用户心智也是最稳健的。2018 年 4 月份，我们正式启动支付宝小程序项目，期间经历了基础购票版本，视频 Feed 流版本，春节五福版本等数个版本迭代，最终形成淘票票小程序矩阵中功能最丰富的一支。其中视频 Feed 流是我们跟支付宝小程序团队共同推进的，支付宝方面为我们提供了功能丰富的视频组件，使得小程序的 Feed 流功能成为可能。如果有这方面需求的团队，可以在支付宝小程序平台上尝试。



手淘轻应用

2018 年底，我们启动了手淘轻应用项目，用两个月时间，经历两个版本迭代。在春节档，手淘平台给淘票票贡献了大量的流量。在手淘轻应用项目中，我们遇到的主要问题是技术选型。手淘轻应用技术实现方案有几种选择：Rax、SFC、AppX，其中 Rax 应用最广泛，也最成熟，上线的小程序多数采用这个实现方案。然而因为我们已经开发完支付宝小程序，使用的是 AppX，当时手淘容器对于 AppX 的支持相对较弱，最终为了保证赶上春节档，我们决定采用 Rax 来实现。随着 Rax 不断发展，能力也越来越强，我们近期也在研究基于 Rax 的统一小程序技术体系。

快应用

在快应用正式发布前，我们就一直关注着快应用的发展，因为这是小程序矩阵中唯一不依赖于第三方应用安装，就能直接触达用户的平台。2018 年初，在快应用 SDK 公开不久，我们就着手预研。当时还没有明确的使用场景，于是我们启动了创新孵化类项目，由技术主导，边探索边开发，最终推动产品上线。快应用也是淘票票小程序矩阵中唯一创新孵化类产品，为我们探索新技术的创新落地积累了经验。另外值得一提的是，目前快应用可以直接使用淘宝账号登录，我们也正在跟阿里其他团队一起探索快应用的联动。

字节跳动小程序

头条和抖音是近几年发展迅猛的 App，2018 年 10 月也推出了自己的小程序平台。头条的场景是内容资讯，抖音的场景是内容娱乐，从场景上说跟淘票票的购票场景相距较远。但是我们依然决定要进入这个小程序平台，因为对于阿里文娱来说，有一块很重要的业务是电影宣发。头条和抖音这两个场景在电影宣发领域有这比较大的价值。

我们从 2018 年 9 月份开始与头条、抖音方面沟通，10 月正式启动项目。这个项目过程中的坎坷也很多，面临内部外部各方面的问题。我们自身方面，项目周期跟淘票票的春节档重合，研发资源冲突很大。而春节档是我们极为重要的档期，这相

当于电影人的“双 11”。阿里内部共建方面，我们面临了类似于微信小程序的问题，账号和交易部分需要集团业务平台团队提供协助。但是项目周期跟集团的“双 11”、“双 12”重合，于是又遇到资源冲突问题，整体推进困难重重。不过经过各方努力，项目还是顺利上线。

百度小程序

在头条小程序项目的同期，我们又并行了百度小程序的项目。从技术方面讲，有前面多个小程序作为基础，同时我们又通过 Taro 进行小程序统一化改造，整体效率相对较高。值得提一点的是，在百度 App 的各个子场景中，贴吧的入口很值得去运营。跟小程序本身功能关联大的贴吧能够带来不错的流量。

微信小程序

这是我们最早启动调研的一个小程序，但是却是上线最晚的。2017 年 8 月份，我们开始进行微信小程序调研，然而技术方案上却遇到了巨大的阻碍。主要问题集中在账号、交易、支付等环节。直到 2019 年 6 月，我们磨练了两年的微信小程序终于上线。2019 年底，我们又为影院定制了一套面向影城的小程序方案，帮助影院进行微信渠道的私域运营。

回顾这条“进击之路”，其中的坎坷和苦痛也许只有我们自己知道。在最紧张的时候，我们有四个平台的小程序并行，然而作为阿里技术人，我们不惧艰辛，不惧挑战！用技术和热情，披荆斩棘，勇攀高峰！

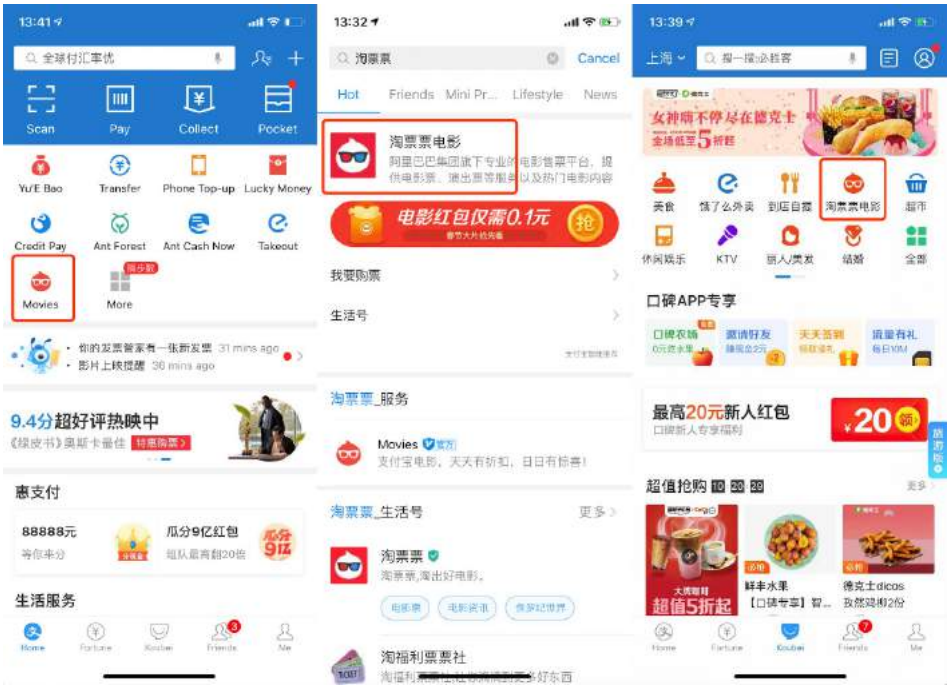
第二章　支付宝小程序实战总结

作者 | 阿里文娱高级前端工程师　亦铭

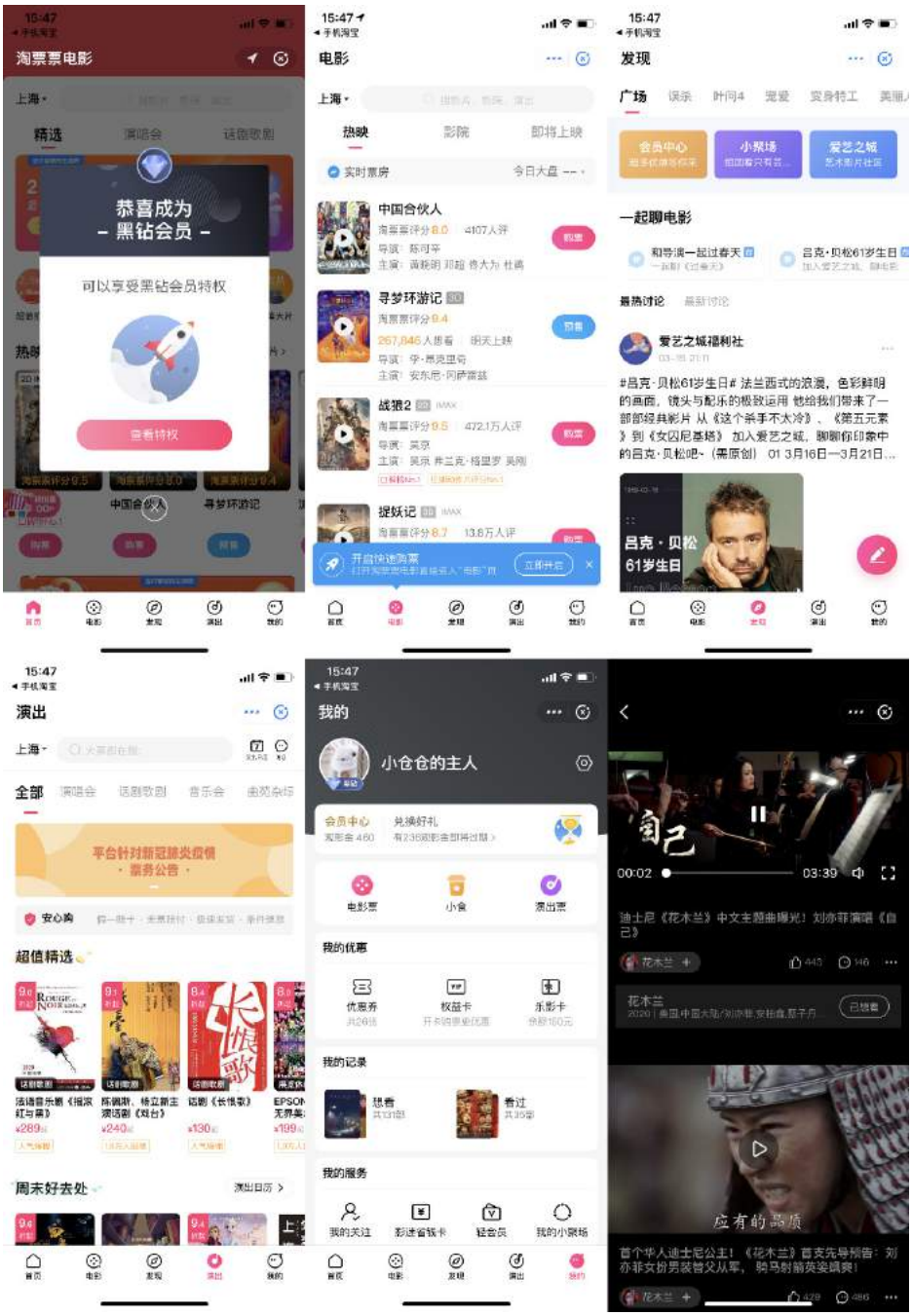
应用场景模块介绍

支付宝内的淘票票用户主要是以购票为主，工具属性比较明显；淘票票入口众多，均引导跳转到小程序，引导用户在小程序内进行购票、娱乐消费、收藏、添加到首页 / 桌面、分享等行为。

1. 入口：九宫格 (或首页 –> 更多 –> 第三方服务)、支付宝搜索、口碑、花呗、语音助手等其他入。



2. 内容：电影购票、资讯 feed 流、视频 feed 流、演出购票、会员业务等。



小程序开发实战

淘票票支付宝小程序，经历了近一年的起步开发，以及一年多的版本迭代，业务开发涵盖了购票、视频、资讯、社区等多个场景。涉及了很多小程序研发技术，以下是相关实战总结：

四项开发技巧

1）在核心业务中慎用 web-view

实际项目线上运行情况及用户反馈表明：web-view 初始化较慢、易受网络干扰、性能差（对比离线包及普通 H5 页面）、问题较多；建议不要在核心业务中使用 web-view 进行承载；

2）自有城市体系与支付宝城市组件的适配技巧

由于支付宝的城市组件是基于自身城市体系的，淘票票拥有独立的城市体系，所以需要城市选择组件适配不同城市体系的场景；经过几轮推动迭代，淘票票线上已使用城市选择组件，已支持复写当前定位城市、历史访问城市、热门城市、城市列表信息等。使用 my.chooseCity、my.onLocatedComplete、my.setLocatedCity 三个 JSAPI 可实现对应效果；

3）如何实现沉浸式效果（透明导航栏）？

首先在 page.json 配置“transparentTitle”为“auto”属性，开启沉浸式布局；其次，页面布局适配沉浸式顶部透明导航栏即可，通过 my.getSystemInfo 获取 titleBarHeight 及 statusBarHeight 可计算出顶部透明高度。注意：Android 5.0 以下由于不支持沉浸式状态栏，所以页面会从状态栏下开始布局；

4）小程序 tabBar 换肤、红点

主要使用的 JSAPI 及 event: my.setTabBarStyle、my.setTabBarItem、page.onTabItemTap，参数参考官方文档即可。注意事项如下：1）小程序触发 relaunch

时，tabBar 的样式会被清除，需要再次设置，目前建议在 app.onShow 里多次触发设置逻辑；2）尽量使用本地图片，在线图片有个下载的过程，体验不太好，且弱网下图片载入可能失败；3）my.setTabBarItem 的参数每一项均需要赋值，否则 Android 可能会报“invalid parameter”；

八项开发注意

1）不要使用 tnpm 安装依赖，tnpm 软连接目前支持有问题。

2）devDependencies 和 dependencies 需要分开，将 devDependencies 移到项目代码外层，否则会额外增加包大小。

3）设置 transparent/pullRefresh 等 window 配置时，跳转别的页面会被继承，需要在 app.json 初始化此类配置信息规避。

4）web-view 里面的页面会失去下拉刷新、resume 等特性。

5）Android 低版本不支持 sticky 属性。

6）某个值控制 dom 是否渲染，下次更新时此值若为 undefined 时不会销毁掉会被忽略掉。

7）window.atob、window.btoa 需要第三方库来替代。

8）lodash 某些方法不能直接使用，因为小程序构建时无 global 对象。

小程序监控

使用阿里云的 ARMS 平台，参考官方文档接入即可。优缺点：有实时大盘，排查用户日志方便，上报更自由、丰富；有接入成本、需要开发，增加包大小，且要收费。

小程序权限

小程序有权限管控，无论是申请 JSAPI 权限还是 H5 域名配置，都是需要打新的包上传才能生效。

下一步：跨端组件复用

淘票票在支付宝小程序上线初期就开始对接，坎坷不断，一路向前，最后成功上线了业务实现小程序化，也推动了小程序框架底层逐步完善及优化。由于阿里域内域外小程序爆发式增长，如何保证一套代码多个小程序输出，其次还要考虑 H5 及小程序的跨端组件的复用，只有解决了这个痛点，才能保证提升用户体验的前提下而不落下研发效率及灵活性。而我们团队也在不断探索及实践着统一化方案，目前已支撑到域外小程序的整体统一。

第三章 主流应用小程序开发总结

作者 | 阿里文娱高级无线开发工程师 适其 / 井草 / 凌阁 / 怀虚

百度小程序

百度小程序概况

百度小程序，依托于百度 App 入口，其流量也是相当可观，攫取流量为淘票票提供入口，百度小程序是非常不错的官方渠道。

以微信小程序为蓝本的百度小程序，也同样具备相似的商业定位。依赖百度这样的老牌搜索门户，百度小程序更加偏向目的性强的搜索热词进行小程序的关联调起和互动，这也是百度小程序和其他小程序的区别。

由此，我们在 2018 年底进行了百度小程序的开发工作，由于在前期积累了小程序开发经验，百度小程序的开发更加的平稳和快速，不到一个月就上线运营了。

应用场景展示

百度小程序入口：

三种入口：百度 App 搜索关联、百度贴吧关联、其他百度生态搜索关联。

百度小程序开发指南

下面是 < 淘票票百度小程序 > 开发过程中遇到的坑和总结：

1）基础开发

百度小程序的开发与微信、头条小程序的开发方式和框架概念非常相似，都属于前端开发的一块子集，主要可以分为 4 块来构建百度小程序的页面：

第一块：HTML。构建页面框架：使用 xxx.swan 文件进行页面元素框架的搭建，具有独特的 HTML 标签如 view，scroll-view 等。

第二块：CSS。管理页面样式：使用 xxx.css 文件进行页面样式的管理，基本的 css 的样式都大部分支持。

第三块：JS。编写页面逻辑：使用 xxx.js 文件进行页面逻辑的书写，小程序具有其独特的生命周期管理方法。

第四块：JSON。组件注册：百度小程序支持通过组件的方式进行页面的搭建，通过在 xxx.json 中注册组件供页面使用。

2）template 模板使用

与其他的小程序相同，百度小程序也提供了模板 template 的能力，使用模板可以提高工程化和代码可维护性，开发者可以在模板中设计代码片段，向外暴露接口注入外界变量之后，可以在合适的时机去使用该代码片段。

但是在百度小程序使用 template 使用时，需要注意传递数据时需要使用 {{{ }}} 三层花括号包裹对象 , 否则数据注入时会出现异常。

百度小程序的 template 的使用：

```
<template is="xxx" data="{{{item}}}"/>
```

作为对比，头条、微信小程序 template 需要两层花括号：

```
<template is="xxx" data="{{item}}"/>
```

3）组件属性的 observer 使用

在使用组件（Component）是大概率会使用到属性的 observer 方法，当属性被改变时会执行属性对应的 observer 方法，此处需要注意在使用 observer 方法时，避免使用下划线开头的方法名，可能会造成 observer 方法的循环调用问题。

或者当发现 properties 中的 observer 方法被循环调用时，检查一下 observer 绑定的方法是否有下划线。方法命名移除下划线，大概率可以解决循环调用问题。

会出现 observer 循环调用的情况：

```
isShowLoadMore: {
                type: Boolean,
                value: false,
                observer: '_isshowChange'
      },
```

推荐的写法：

```
isShowLoadMore: {
                type: Boolean,
                value: false,
                observer: 'isshowChange'
      },
```

4）scroll-view 的使用

在使用 scroll-view 的开发过程中，对存在多个可滑动区域的页面且其中一个滑动区域为 fixed 样式时，iOS 机型会偶现 scroll-view 空白的问题。

可能存在异常的页面布局如下：

```
<view class='头部组件 ' style='position:fixed'/>
```

```
<scroll-view class=' 可滑动区域 1' style='position:fixed'/>
<view class=' 可滑动区域 2' />
```

其中“可滑动区域 2”为依赖内容撑开的页面 View，当内容到达一定长度时，页面 View 会提供滑动能力。如果使用上述写法可能会出现 scroll-view 空白的问题。

推荐的写法：

```
<view class=' 头部组件 ' style='position:fixed'/>
<scroll-view class=' 可滑动区域 1' style='position:fixed;height:44px' />
<scroll-view class=' 可滑动区域 2' style='height:80vh' />
```

5）小程序 DSL 页与 WebView 页的登录流程

小程序的页面实现方式可以分为两种：一种为小程序原生的页面；另外一种是使用 WebView 组件，将 H5 页面展示在小程序中。处理两种页面的登录时一般是先进行 DSL 页登录（小程序原生页面），完成 DSL 页登录后，再进行 H5 容器页的登录。

a）DSL 页登录：

先进行小程序的登录授权，获取到小程序的登录凭证，拿着登录凭证去自己的业务服务端获取真实的小程序登录信息，当开发者完成上述流程之后，将登录态信息加密后存储在本地。下次进行需要登录校验的页面时，进行本地登录信息的登录校验，则 DSL 页的登录流程就完成了。

b）WebView 容器页登录：

由于百度小程序无法操作到 WebView 容器的 cookie 信息，所以在 WebView 容器页进行登录时，势必要进行一次从服务端获取登录 cookie 的过程。目前可以在进入需要登录校验登录的 WebView 容器页之前，先发起获取 cookie 的服务端请求，服务端处理好用户登录信息校验之后就可以提供一个同步 cookie 的专用页面；当接口返回链接之后，小程序的 WebView 容器需要做的就是访问这条链接将服务端返回的 cookie 同步到 WebView 容器中，这样 WebView 容器就具备了可供校验的登录信息。

完成上述页面的登录操作之后，小程序登录流程就结束了。

本节着重描述的是开发过程中大概率会遇到的问题和解决方案，最好结合官方文档一起查看。

字节跳动小程序

字节跳动小程序概况

字节跳动小程序是基于字节跳动全产品矩阵开发，与图文、视频等场景有着天然的搭配性，带动小程序分发，由内容为小程序带量以及裂变。作为一个大流量且高度活跃的平台，具有很大用户增量挖掘空间。

对于头条系应用，同一小程序可以同步上线多个宿主端，目前已开放今日头条、抖音、头条极速版。无论是抖音，还是今日头条，都属于内容分发平台，相比公众号读者，抖音用户相对更年轻，而头条则拥有大量三四线城市读者，这正好契合了电影作为内容消费的特质，帮助淘票票更好的拉动下沉用户。基于头条、抖音平台自身的优势，我们在 2019 年上线了淘票票字节跳动小程序。

应用场景展示

今日头条的六个主要场景：

1）信息流推荐

2）搜索直达

3）头条号挂载小程序

4）分享

5）中心化入口

6）留存入口

今日头条

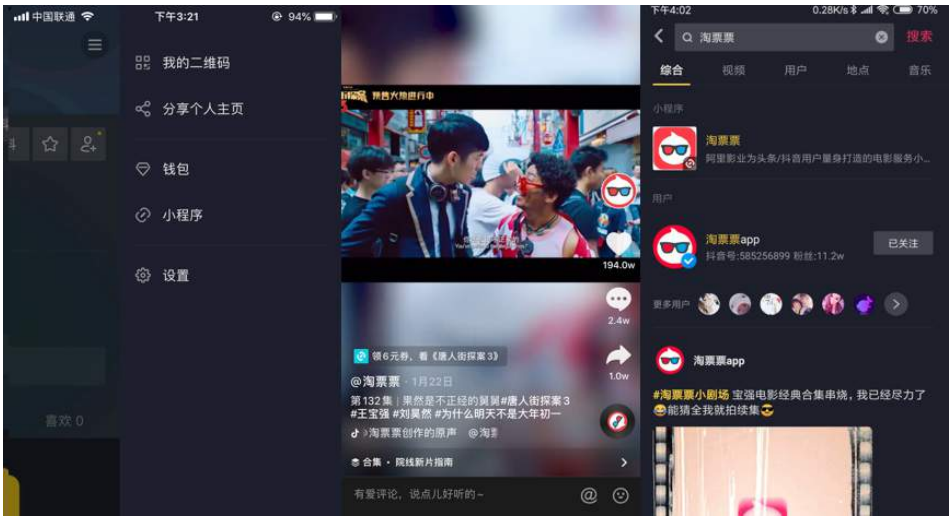
抖音的四个主要场景：

1）小视频挂载

2）企业号主页

3）搜索展示

4）留存入口



广告投放

开发思维介绍

字节跳动小程序基本开发思路类似于前端开发，并增强调用大量端能力，性能体验优于普通 Web。上层架构基于 JS 开发，可以辅助开发者进行良好得开发。框架结构和开放式类似于支付宝小程序、微信小程序和百度小程序。

目录结构：主要分为以下几类的文件：

1) .json 为后缀的 JSON 配置文件，这个文件配置了小程序所有页面的路径和界面展现样式等；
2) .ttml 结尾的模板文件，用来描述当前这个页面的文件结构，类似于网页中的 HTML 文件；
3) .ttss 结尾的样式文件，描述页面样式，类似于网页中的 CSS 文件；
4) .js 结尾的 JS 文件，处理这个页面和用户的交互。

```
|____app.css
|____app.json
|____project.config.json
|____pages
|       |____detail
|       |       |____detail.ttss
|       |       |____detail.ttml
|       |       |____detail.js
|       |____index
|       |       |____index.js
|       |       |____index.swan
|       |       |____index.css
|____app.js
```

目录结构

以上是基于淘票票小程序开发过程中的总结，由于小程序技术本身的快速迭代，有些内容难免有些纰漏，希望读者在阅读时能结合官方文档。此外，随着字节跳动旗下其他应用对小程序功能的逐步开放，可以最大化的利用各自平台优势为产品赋能降低用户使用门槛，完成服务闭环，提升用户体验。

快应用卡片

快应用小程序概况

当前，基于超级 APP 推出的各种小程序，对手机厂商的分发能力及话语权有明显削弱趋势。因此国内各手机厂商在推出快应用后，也逐渐对外开放手机负一屏的能力，为快应用及其他服务提供直接的入口。目前来看这是一个对 APP、手机厂商、用户三赢的场景。对于用户，手机负一屏直接给到服务，使用更方便；对于手机厂商，提升厂商应用分发能力；对于内容提供方，增加流量入口，减少入口深度，提升内容价值。

卡片类型

快应用的卡片类型可以分为：应用类型的卡片、服务类型的卡片和其它类型的卡片。

1）应用类型的卡片：是用户订阅的一种卡片，内容相对固定；

2）服务类型的卡片：针对用户关心数据的状态，内容实时变更；

3）其它类型的卡片：自定义卡片，根据实现对应内容展示及跳转。

1. 应用卡片的具体接入

卡片的开发基于快应用，所使用的 API 是快应用的子集，部分 API 不能在卡片中使用。目前已知的 vivo，OPPO，NUBIA 都需要卡片的开发不依赖主 rpk，也就是需要保证卡片能脱离主 rpk 独立渲染，为保证卡片的独立渲染，不能使用 this.$app 相关对象及文件 app.ux 中声明的工具类或生成的对象。

1）卡片的初始化配置

a）配置文件

所有的卡片都需要和快应用中声明页面一样在 manifest.json 中声明。具体是在 router.widgets 中配置，各厂商之间有部分差异，但差异不大。

b）卡片配置文件中需要注意的是：

【widgets】中配置的 key 值请使用路径名字，如果路径为两级（例：/A/B），则 key 值配置为”A/B”，且该值最终对应到厂商后台上传卡片时填入的卡片路径，基于此厂商才能正确解析到上传到联盟上统一的快应用包中对应的卡片。

卡片的属性【features】中需要声明该卡片会用到的系统 API，这些 API 在外层应用的 [features] 中已经声明过，此处需要再次声明，否则不能使用。

2）应用类型卡片接入（以 vivo 为例）

负一屏卡片线上效果图

a）卡片的声明

在 manifest.json 中的除了上面提到的配置之外，对于卡片需要注意卡片属性中的以下字段：

【 params 】字段用来配置卡片更为详细的参数，及特有的支持参数，需要按照文档进行配置。

【 targetManufactorys 】为对应厂商适配标明该卡片匹配的厂商，具体参看文档。

b）卡片的开发的不同点（所使用的 API 及组件为其子集具体参看官方文档）：

因为 vivo 卡片是单独工程，不能配置在快应用工程中，所以需要另外建立新的工程；对应包打出也需要单独配置和主快应用不相同，需要用到 vivo 给出的相关工具；卡片有单独设置主题的功能；卡片有折叠功能；卡片视觉稿由内容提供方给出；卡片开发只需开发内容区域，title 区域无需开发（由 vivo 负一屏容器完成绘制）同时

意味着下半部分的圆角需要自己绘制；上传卡片包时需要提供对应的 icon。

c) 卡片调试

卡片调试需要使用 vivo 方提供的工具打出来的 rpk 文件，同时需要使用 vivo 方提供的专用内核及容器，具体按照文档执行即可。

d) 卡片提交

首先需要完成自测，自测之后需要使用 vivo 提供的专用打包工具，打包之后到 Jovi 后台地址提交，同时需要提供一个应用图标。

2. 负一屏服务类型卡接入

以下以 OPPO 服务卡接入为例：

触发场景：用户在淘票票快应用中购票之后，在影片上映前的固定时间内触发该卡片内容展示，进而提醒用户取票，即消息触发场景。

OPPO 负一屏卡片线上效果图

1）OPPO 服务卡卡片的声明

在 manifest.json 中的 router 字段中添加【 widgets 】字段，并在该字段中添加对应的配置，与 OPPO 应用卡片完全相同；

2）卡片开发

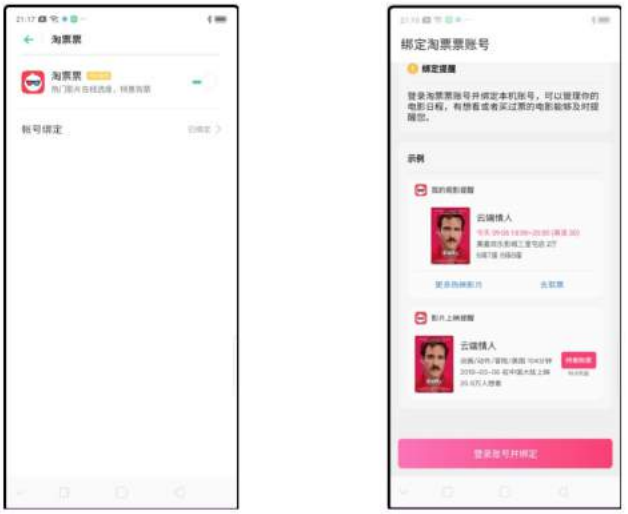
OPPO 服务卡开发涉及用户关心数据状态改变触发卡片的场景，因此整体需要解决以下几个问题：首先是触发时机问题，然后是要确认触发的卡片 ID，还要解决要触发哪一个 OPPO 用户；

3）OPPO 服务卡整体开发流程如下：

前提：要开通 OPPO 账号服务，保证在快应用中能够拿到 OPPO 当前登录的用户的授权码。

a）账号绑定，即 OPPO 账号和快应用账号的绑定

账号绑定的入口：该入口由 OPPO 负一屏容器统一提供，位置如下图左：



OPPO 服务卡绑定入口及自定义绑定页面

该入口对应一个快应用内的绑定页面。

b）绑定页面开发，该页面是快应用页面，主要提供绑定功能

作用是让内容商服务端知道自己的账号和 OPPO 测的对应关系，及换取发消息到 OPPO 端时所需要的 TOKEN 值。

c）触发对应 OPPO 用户负一屏的卡片

内容服务商在用户关心数据变更时，触发推送消息到 OPPO 服务端，该消息按 OPPO 文档约定，带上对应的 TOKEN 值，要触发的卡片 ID，消息内容，要触发的时机及时长，OPPO 服务端会根据该 TOKEN 找到对应的用户下发消息，并在需要的时机拉起对应 ID 的卡片。

d）卡片消失

由发送消息中定义的卡片时长决定，展示时间到点后，负一屏容器会自动移除该卡片。

e）调试

首先，需要 OPPO 服务端参与；其次，需要 OPPO 提供负一屏开发环境版本，以保证 OPPO 服务端日常环境的数据能够到达；再次，需要提供初步测试完成包含服务卡的 rpk 到 OPPO 侧，把该 rpk 配置到 OPPO 对应的环境。

f) 提交市场

测试完成可以在 OPPO 后台提交该卡片，同时同步正式生成的卡片标示到自己的服务端用来推送消息使用。

g）综上涉及各端的开发工作如下：

首先，涉及服务端账号绑定开发，TOKEN 刷新维护，触发的消息推送到 OPPO；其次，涉及前端的服务卡片开发以及绑定页面开发；涉及其他：OPPO 账号

服务开通。

3. 踩坑记录

1）负一屏的 UA 和快应用中不同如果有与 UA 相关的配置需要注意；

2）对于调试时更新了 rpk 之后，实际打开对应更改没有体现时，可以尝试清除对应卡片容器的 cache，同时保证该容器有相应的读取存储的权限；

3）对于同一个业务，由于各厂商适配不同及平台不同，需要多处代码编写，但基本业务逻辑基本一致，唯一不同是 UI 展示，所以在一开始还是需要抽离数据逻辑，不同厂商给不一样的 UI 展示即可。

淘票票小程序构建演进

在 2018 年我们做了很多的小程序，对多端同构也做了一些尝试。

从一端到多端

1）起步：小程序原生开发

2018 年，随着小程序平台爆发，我们首次踏出了淘宝域内，进行了头条小程序的尝试。这次尝试，使用的是原生的小程序 DSL 语法编写。为了方便复用已有的 H5 样式，加入了 Gulp，用来编译 Less。

这种开发方式轻快，但是同时也暴露出了很多问题：

a）包体积很难控制；

b）原生 DSL 没有任何复用性，并且需要重新学习；

c）无法使用 NPM，一些很常用的社区包，团队基础工具链无法使用；

d）机型兼容性不好，没有 babel 支持。

2）摸索：两个端，一套代码?

在开发百度小程序的过程中，吸取了第一次的教训，加入了 webpack 来做一层编译，一是解决了包引入问题；二是加入了 babel 插件，解决 JS 兼容性问题；开启 CommonChunk 插件，解决包大小问题。

总体上，从输出一端变为输出两端，所以出现一些差异。对这些差异，编写了一个插件，对业务层抹平。比如微信端引入 index.wx.js，头条端引入 index.tt.js。

脱离了刀耕火种，开发效率明显提升，但是还不够好，视图层还是两份，而且以后每新增一端就要新拉出来一份。

3）优化：走向社区，跨多端

在开发头条和百度小程序时，业内也已经有了在小程序 DSL 上封装的框架，但是当时看都不是很成熟，基本都是专注于一个平台，没有什么跨端能力，就没有用到生产环境，而是持续关注更新近况。

2019 年进军微信小程序，再次看市面上的框架，发展的很快，同时也注意到跨端开发这个需求点，选择了 Taro 作为主力框架。这种框架横评就不展开了，市面上很多，简单说几个选择 Taro 的原因：社区相对活跃、支持渐进式切换、TS、react like。

a）平滑迁移

Taro 支持渐进式切换，也就是 Taro 和 DSL 混写的能力，所以迁移成本可以接受。我们先将首页 Taro 化，后面慢慢迭代将所有的页面都切换为了 Taro，这里值得一提的是，Taro 的跨端差异化处理和我们之前的处理思路一样，因此 Util 迁移起来几乎 0 成本，成本主要集中在视图层。

b）好处是什么，缺点在哪里?

使用 Taro 的好处是解决了我们之前遇到的主要问题，是一个一揽子解决方案。

同时这种上层框架在扩展新端时成本低，机动性很高，框架提供了新平台包，适配成本低。

当然也遇到了一些新的问题，比较严重的是调试，因为代码被转译过一次，同时不支持 Soucemap，导致 debug 时体验很差。

多端差异

多端必然会有一些差异，业务的差别、端上 API 的差异等，比如微信上的分享能力，抖音上的抖音拍摄器，百度的 feed 流等。最终落在业务上，差别可以分为三部分，输出不同的页面、不使用同的组件 (有的端使用原生组件)，细到不同的逻辑。

1）输出不同的页面

在使用 Taro 时发现不支持，想到可以使用 babel-preval 来编译时输出页面配置，这样包体积也不会受影响，最后我们也反哺回社区。

使用不同的组件，不同的逻辑。根据端上不同的组件我们使用的最多的是多态模式，底层组件对外暴露相同的接口，端上调用时不需要考虑端上的差异，在 import 层会根据不同的端来引入不同的具体组件。

2）端上逻辑

如果是一些简单的逻辑差异，可以直接使用环境变量来做控制，走不同的逻辑。这种方式针对小一些的逻辑还可以，不过这种代码一多，就不容易维护。

3）针对维护性的建议

这里推荐几种维护性比较强的差异处理方式：

a）设计组件时插件化：比如路由，不同端在跳转前后需要有一些不同的操作，实现了插件化后，每一个端只需挂载不同的插件即可；

b）配置抽离：针对一些端上不同的配置，比如一些文案、固定内容等，可以抽

离到一个统一的地方维护，可以少很多 ifelse 逻辑；

c）用好函数 hook：针对不同端相同的逻辑放在函数中，有差异的逻辑可以单拆函数作为 beforeHook 和 afterHook。

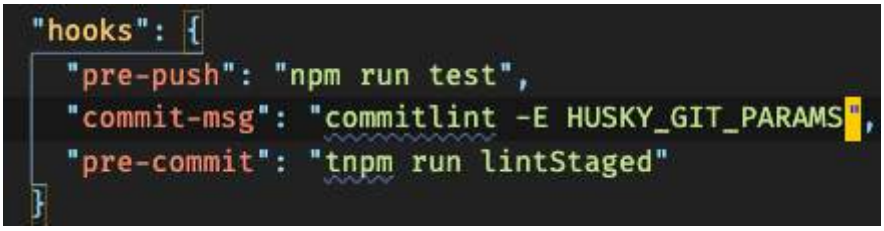
项目维护策略

项目输出多端后，每次改动回归就成为一个比较重要的问题，如何保证自己的代码不会再其它端上出问题？每次改一个小程序其他都要立即回归吗？如何快速整理其他端的改动？下面针对多端项目的维护总结了一些经验。

1）单测：针对核心逻辑编写测试，unit test 和 snapshot test，我们内部维护了一个针对端上 API 的 mock 测试库，整个测试可以在 node 环境中运行，保证运行效率。

2）Commit 规范：指定一个 commit message 规范，可一眼看出你在做什么，在改哪一个端，以及后面回归策略时用到。

3）git-hook



使用 githook 能保证上面的规范能够真正的遵循，保证每次提交的质量。pre-commit 时跑一下 Eslint，然后校验一下 commit-message 是否符合规则，最后 push 时会跑一次整体的测试。

4）多端的回归策略

没有做 E2E 的主要原因是小程序限制，case 编写难度比较大，并且维护性低，

无法自动化。

目前我们是人工回归的，如何保证代码不会再其他端上出问题？难道每一次改一个小程序其它都要立即回归吗？回答下这两个问题，编写代码时考虑影响面，提交时提供足够的改动信息，合并时主要测当前端即可，不需要回归所有端；等另一个端需要发布时，拉出版本的 commit-message，然后梳理出变更范围，在该端做回归即可。这样做减少对测试的集中消耗，保障质量。

如何处理跨段项目

以上是我们对跨端项目的经验总结，包含技术演进历史以及差异控制策略。跨端项目的难点就是处理差异。1）端上能力的参差不齐、业务针对不同场景的定制，一旦控制不好，整个人项目的维护性就会大大降低。2）业务方面要思考清楚，不同的端，是相似的更多，还是差异多。3）框架方面，最近看到有开发者已经给 W3C 提小程序的白皮书了，总体朝着良性方向发展，这是一个好的开始，期待能够标准化小程序框架。

第四章 淘票票小程序工程化技术总结

作者 | 阿里文娱无线开发专家 午天

作者 | 阿里文娱高无线开发工程师 适其

Taro 化之框架浅析

TaroJS 是一套遵循 React 语法规范的多端开发解决方案，由于多端代码一致所以大幅提高开发效能。目前业界有不少的小程序就是采用了 Taro 的方式进行了多端小程序代码融合。完成多端代码融合之后，对于开发效能是有了质的提升，目前淘票票小程序已经上线 Taro 化的版本。

Taro 原理浅析

下面我们来看一下 Taro 的大体实现（图 2-1）：

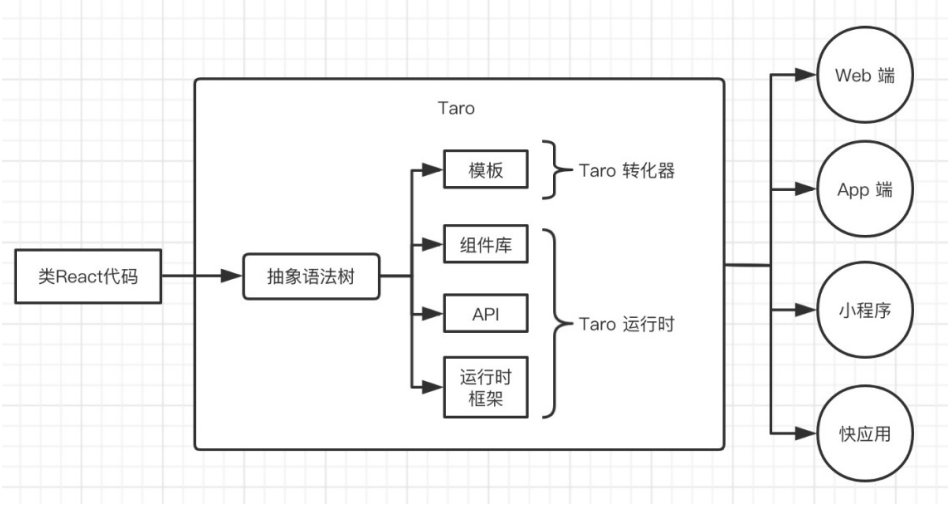


图 2-1

从图上可以得知，Taro 使用 React 编写语法进行功能开发。Taro 有一套自己的抽象语法树将编写之后的 React 代码转化到各个小程序平台。render 方法中的渲染语法会转义到各个小程序平台的模板文件中，如百度小程序的 swan 或头条小程序的 TTML。而对于 JS 层面的逻辑和 API 调用方面，在运行时会通过代码 hook 包装的方式，转义到各个小程序端的 JS 文件中，从而实现代码的多端统一。

整体的转义流程如下（图 2-2）：

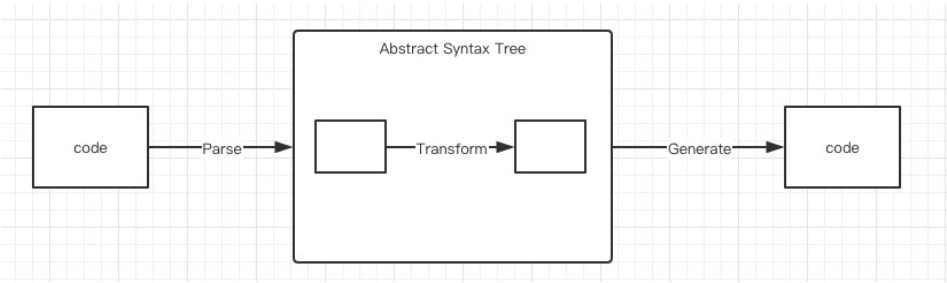


图 2-2

所以在使用 Taro 开发小程序时的流程基本是：

1）编写类 React 的业务源代码；

2）通过 Taro 编译命令进行代码的转义；

3）在对应小程序端的开发工具中打开对应生成的小程序代码进行编译。

我们再来分析一下 Taro 的 runtime 转义逻辑，下面来看一下 Taro 是如何在运行时进行代码包装的。Taro 库文件结构如下：

```
├── @tarojs
│   ├── async-await
│   ├── cli
│   ├── components
│   ├── mobx
│   ├── router
│   ├── taro
│   ├── taro-h5
```

```
|   ├── taro-swan
|   ├── taro-tt
|   └── taro-weapp
```

在 Taro 库文件中，有对应的各个端的适配逻辑，在进行输入编译命令进行编译时 Taro 会通过小程序类型使用对应端的转义代码。

我们来看一段转义之后的小程序代码：

```
var _TaroComponentClass = (_temp2 = _class = function (_BaseComponent) {…
}(_index.Component), _class.$$events = ["anonymousFunc0", "anonymousFunc1"], _class.$$componentPath = "pages/

exports.default = _TaroComponentClass;

Page(require('../../npm/@tarojs/taro-tt/index.js').default.createComponent(_TaroComponentClass, true));
```

这是一段小程序某个页面的 JS 代码。在 Taro 的设计中小程序的每个页面都是一个 TaroComponentClass ，Taro 将页面的相关逻辑代码都收归到了 TaroComponentClass 这个对象之中。

再看 TaroComponentClass 这个对象中的实现：

```
var _TaroComponentClass = (_temp2 = _class = function (_BaseComponent) {
  _inherits(_TaroComponentClass, _BaseComponent);

  function _TaroComponentClass() {
    var _ref;

    var _temp, _this, _ret;

    _classCallCheck(this, _TaroComponentClass);

    for (var _len = arguments.length, args = Array(_len), _key = 0; _key < _l
      args[_key] = arguments[_key];
    }

    return _ret = (_temp = (_this = _possibleConstructorReturn(this, (_ref =
      navigationBarTitleText: '我的'
    }, _this.anonymousFunc1Map = {}, _this.customComponents = [], _temp), _po
  }

  _createClass(_TaroComponentClass, [{
    key: "componentDidShow",
    value: function componentDidShow() {
      this.getUserInfo(false);
    }
```

如上图，TaroComponentClass 中将源代码中的方法都通过 map 映射到了自身，对于 React 框架中的生命周期方法也是通过 map 的方式映射到 TaroComponentClass 中去，进行二者的相互关联。

最后在 @tarojs/taro-xx 的库方法中会有对 TaroComponentClass 的处理，如下：

```
function createComponent(ComponentClass, isPage) {
  var initData = {};
  var componentProps = filterProps(ComponentClass.defaultProps);
  var componentInstance = new ComponentClass(componentProps);
  componentInstance._constructor && componentInstance._constructor(componentProps);

  try {…
  } catch (err) {…
  }

  initData = Object.assign({}, initData, componentInstance.props, componentInstance.state);
  var weappComponentConf = {…
  };

  if (isPage) {
    weappComponentConf.methods = weappComponentConf.methods || {};

    weappComponentConf.methods['onLoad'] = function () {…
    };

    weappComponentConf.methods['onReady'] = function () {
      this.$component.__mounted = true;
      componentTrigger(this.$component, 'componentDidMount');
    };

    weappComponentConf.methods['onShow'] = function () {
      componentTrigger(this.$component, 'componentDidShow');
    };
```

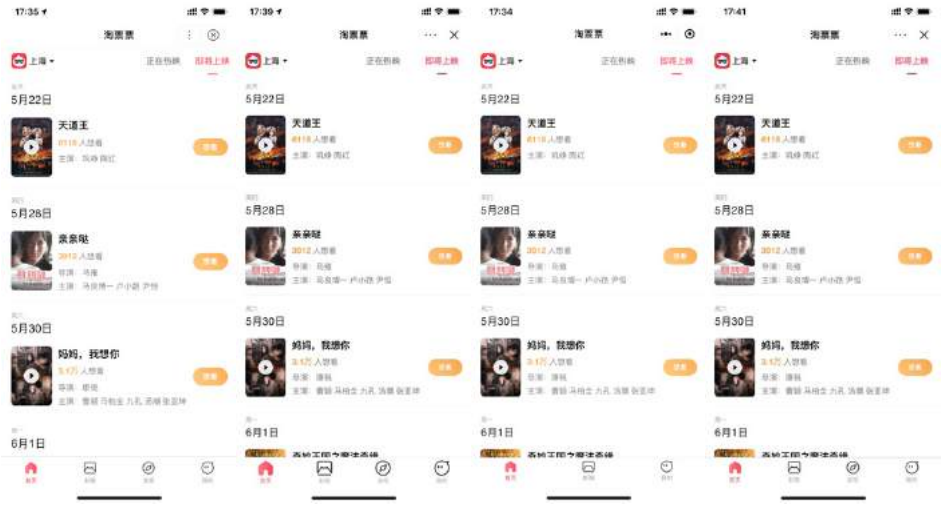
在 createComponent 方法中会将小程序的生命周期方法如 onShow,onHide 等方法映射到之前注册到 TaroComponentClass 的 React 框架方法中，从而实现 React 逻辑代码到小程序 DSL 代码的映射。

Taro 就是通过上诉方式实现了 Taro 运行时代码转义工作。

票票 Taro 化小程序

使用 Taro 进行多端统一之后目前 百度、头条、微信、抖音 等小程序已经在一

套代码逻辑下进行维护了。



小程序自动化埋点方案

小程序爆发痛点

2018 年小程序如雨后春笋般涌现，目前现有的小程序有微信小程序、百度小程序、头条小程序等，头条下面还分头条小程序、抖音小程序、皮皮虾小程序。

淘票票自 2018 年至今，相继尝试和接入了几乎所有的小程序，在小程序埋点实施中，我们主要遇到以下痛点：

1) 各端 DSL API 不一致：各小程序平台的 DSL API 实现不一致，适配成本高；

2) 维护和扩展成本高：比如埋点要增加参数，需要修改所有平台的小程序；

3) 工作量大：在不借助框架的前提下要完成各端小程序的埋点，工作量是很大的；

为了解决这些问题我们决定实现一套小程序自动化埋点框架。

埋点方案探索

对于一套埋点方案来说有三个关键性的环节：采集上报、数据落库、数据分析，而本方案主要涉及到的是数据的采集和上报环节，其整体的实现思路主要包含以下三步：

1）在不同小程序端获取要采集的数据；

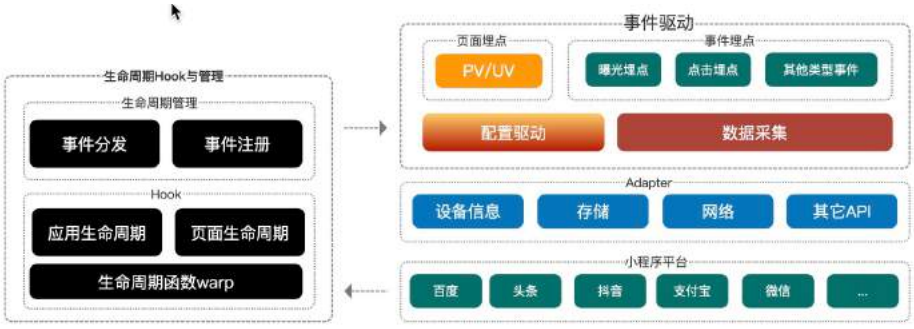
2）根据请求规范完成数据的上报；

3）日志后端会将该数据识别为相应的小程序的日志，然后进行落库。

这是我们埋点方案的最初版，其中第 1、2 步是该埋点方案所需要解决的问题，第 3 步可以借助阿里现有的能力来实现。但该方案还无法实现自动化埋点，小程序自动化埋点的最核心的就是小程序声明周期的 hook，为了实现生命周期的 hook，我们借助 JavaScript 自执行函数的特点实现了生命周期管理与 Hoook 层（下文会介绍），然后配合为不同小程序设计的 Adapter，实现了自动化的埋点框架 Slog。

1. 小程序埋点框架的架构剖析

在整体架构上埋点框架共分为事件驱动、生命周期 Hook 与管理、适配器三部分：



1）事件驱动层：对外的入口也是埋点框架的门面，提供 VV 页面级埋点，以及曝光、点击、等其他类型的事件的埋点相关的功能，同时它支持配置驱动模

式，根据配置进行自动化埋点，另外，它也负责埋点信息的采集和上报；

2）生命周期 Hook 与管理层：负责小程序生命周期的 hook，负责生命周期的管理和分发，同时会根据配置来配合事件驱动层进行自动化埋点；

3）Adapter：对上（事件驱动层）提供统一的 API 接口，对下（各小程序容器）提供不同小程序平台底层 API 的适配。

2. 小结

目前小程序自动化埋点方案不仅支持对传统 DSL 小程序的埋点，也支持基于 Taro 框架开发的各种小程序。

小程序账号登录方案总结

小程序的登录是小程序业务流程中重要的一环，淘票票小程序页面在容器特性上可以划分为两块，DSL 页面与 H5 页面，由于页面特性不同。那么在进行登录流程设计时就需要针对这两块进行区别处理。要完成 DSL 页面登录和 H5 页面登录才能算是小程序的登录完成。

先科普一下小程序这个生态的官方登录逻辑 , 小程序由于内嵌于第三方 APP 中，当小程序需要登录时，其宿主 APP 需要是登录状态（抖音 APP 未登录无法使用小程序）。小程序一般都提供了登录的 API xxx.login() 通过调用小程序的 login 方法可以访问小程序的登录凭证，通过自己的业务接口将小程序的登录凭证换取用户标识来标记用户（其中换取登录标识的逻辑主要在服务端可以通过查阅小程序的官方文档接入），从而完成小程序的登录。

以上就是小程序的一个标准登录流程，在了解了一个大体的登录流程之后，再来看看淘票票小程序的登录与市面其他小程序的登录有哪些区别和优化。

淘票票小程序登录流程介绍

通过上面介绍的域外小程序官方流程，相信大家应该对整个小程序的登录体系有

了大致的了解。

在淘票票接入登录业务的时候由于自身与其他小程序实现方式不同，票票小程序目前采用 DSL 与 H5 混合方式进行开发，登录态需在 DSL 和 H5 两个模块同时生效。所以淘票票小程序登录态就需要分为 DSL 和 H5 两个模块去管理。

那么一次完整的登录，就可以看成是 DSL 登录 +H5 登录来实现的：

先来介绍 DSL 登录：

淘票票小程序 DSL 登录的大致流程，与市面其他小程序的登录流程类似，主要分为 4 个阶段：如图（2-1）

图（2-1）

1）请求小程序平台 API 授权登录；

通过 `xxx.Login` 方法进行小程序平台的授权登录操作。

```
xxx.login({
success: res=> {console.log('login success', res) },
fail: err=> {console.log('login fail', err);}
        });
```

2）获取小程序平台用户信息；

完成小程序登录授权之后再通过 `getUserInfo` 方法获取小程序平台的用户信息，在这个步骤中，小程序框架会弹出用户信息的授权框提示用户授权用户信息，进行登录操作。

```
xxx.authorize({
               scope: 'scope.userInfo',
               success: function(res) {xxx.getUserInfo();}
              });
```

3）拿到用户登录凭证后，访问登录接口，进行登录请求；

```
xxx.request({
              url: xxx/login,
              data: requestData,
              header: {'content-type': 'APPlication/xxx'},
              dataType: 'json',
              method: 'POST',
              });
```

4）调用登录接口返回登录凭证后，持久化到小程序 DSL 存储中，完成 DSL 页面登录态存储。

完成了 DSL 登录之后，再来看一下 H5 容器的登录流程：

由于域外小程序的 H5 容器页需要通过 SetCookie 的方式在 H5 页面上种入登录信息，来完成 H5 登录态的持久化。所以 H5 登录流程一般是在 DSL 登录完成之后进行可以分为两步。

a）服务端需要设计一套提供给 H5 页面注入登录 Cookie 的中间页面，同时页面的链接通过之前 DSL 登录接口下发，当小程序 DSL 登录完成之后，立即访问接口下发的 setCookieUrl 地址，进行登录 Cookie 的注入；

b）在注入 Cookie 完成之后，H5 页面需要调用小程序的 postMessage 方法提醒 H5 容器页，当前 Cookie 注入完成。并提示用户登录完成。

如何进行登录优化

上述内容介绍了，小程序的整个登录流程。在完成登录流程的前提下，淘票票小程序也做了一些针对登录模块的优化：

1）H5 容器页 Cookie 保活优化。

由于小程序的 H5 容器页，对于 Cookie 的存储，只能停留在一次小程序生命周期的时间段内。如果关闭小程序，再次打开 H5 容器页的 Cookie 就会丢失，导致 H5 登录失效。这样的交互明显无法满足正常的业务需求，针对这种场景淘票票小程序制作了一套“静默登录”的补充逻辑。

a）淘票票小程序会在第一次打开小程序时检测当前小程序 DSL 模块是否登录，如果登录则在打上标记。

b）当页面跳转路径是 DSL 模块向 H5 模块跳转时，则拦截本次跳转，并通过检查本地登录时效判断是否发起请求接口登录，当需要更新登录态时，则发起登录请求。登录接口入参添加被拦截的 H5 页的目标 url，请求返回后访问返回的 Cookie 注入页 URL，完成 H5 容器注入 Cookie 那么整体的 H5 的登录态更新就完成了。

c）最后当登录态更新完成之后，注入 Cookie 页会将当前页面 302 到登录接口传入的 H5 目标页 URL，完成整个静默登录。整个静默登录流程用户完全无感知，也解决了 H5 容器页 Cookie 丢失的问题。

2）H5 容器注入 Cookie 流程简化。

由于 H5 登录时，每次都需要打开一个 H5 容器页去注入登录 Cookie，用户体验较差目前，目前淘票票小程序将 login 页面改造为带有容器页面的 DSL 页。解决多次跳转的问题。

淘票票小程序在登录的 DSL 中，内嵌了一个专门用于 Cookie 注入的 Web-

View 组件。并设计了 isShowWeb 的组件开关，当登录接口完成请求之后，需要访问 Cookie 注入页时，页面会将当前登录页的 isShowWeb 开关打开，当前登录页就会渲染 web 容器，在登录页完成 Cookie 的种入并完成 H5 登录。使用登录页内嵌 web 容器方案之后，可以减少一次小程序登录时的页面跳转，优化用户体验。

阿里云 开发者社区

加入交流群

阿里巴巴文娱技术
钉钉交流群

关注我们

阿里巴巴文娱技术公众号

更多独家电子书

阿里云开发者“藏经阁”
海量免费电子书下载