百度开源智能小程序runtime框架介绍和性能优化实践

付嘉兴

百度 技术经理

自我介绍

我是2010年加入百度，2011年开始深耕百度App客户端开发。历经了百度App的所有版本，见证了1个用户到亿级用户，负责过Android 搜索、Feed流。2018年，百度从战略高度启动百度智能小程序，负责组建团队开发小程序runtime。当前核心任务是提升小程序运行全流程的体验。

目录

百度智能小程序runtime框架

百度智能小程序的历史演进

轻应用

- 2013百度世界大会
- 特点：
  - Hybrid方案，H5+端能力

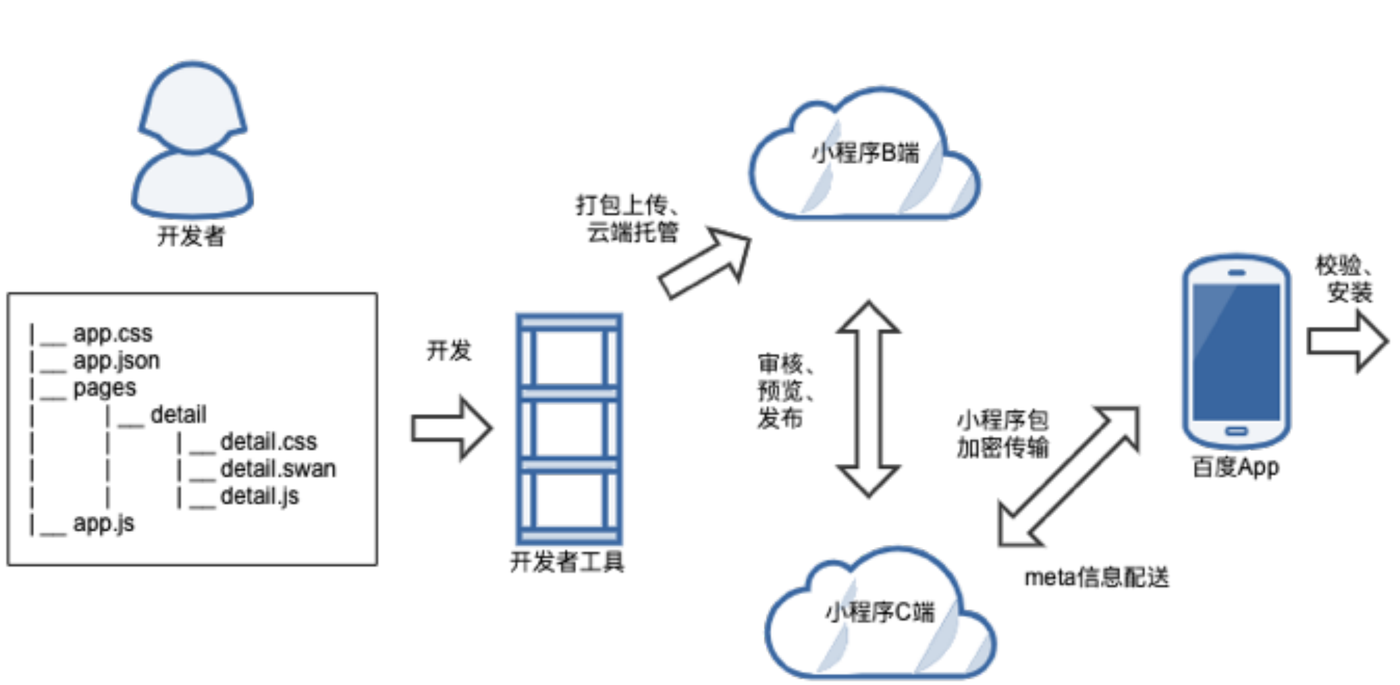
直达号

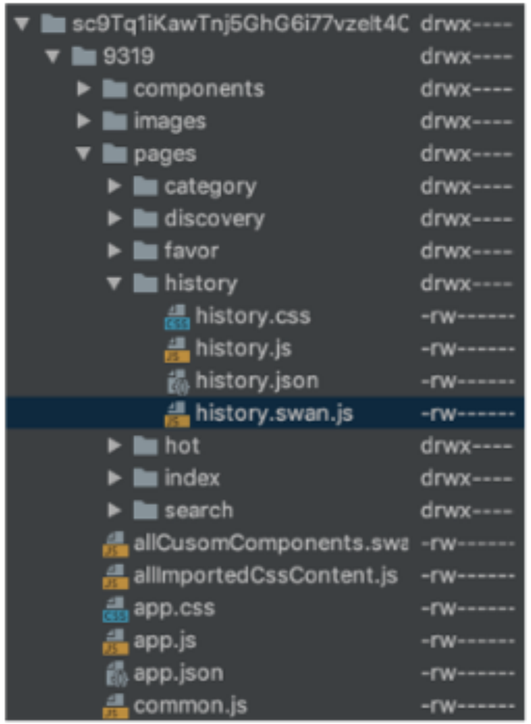
- 2014百度世界大会
- 特点：
  - 增加@直达

智能小程序

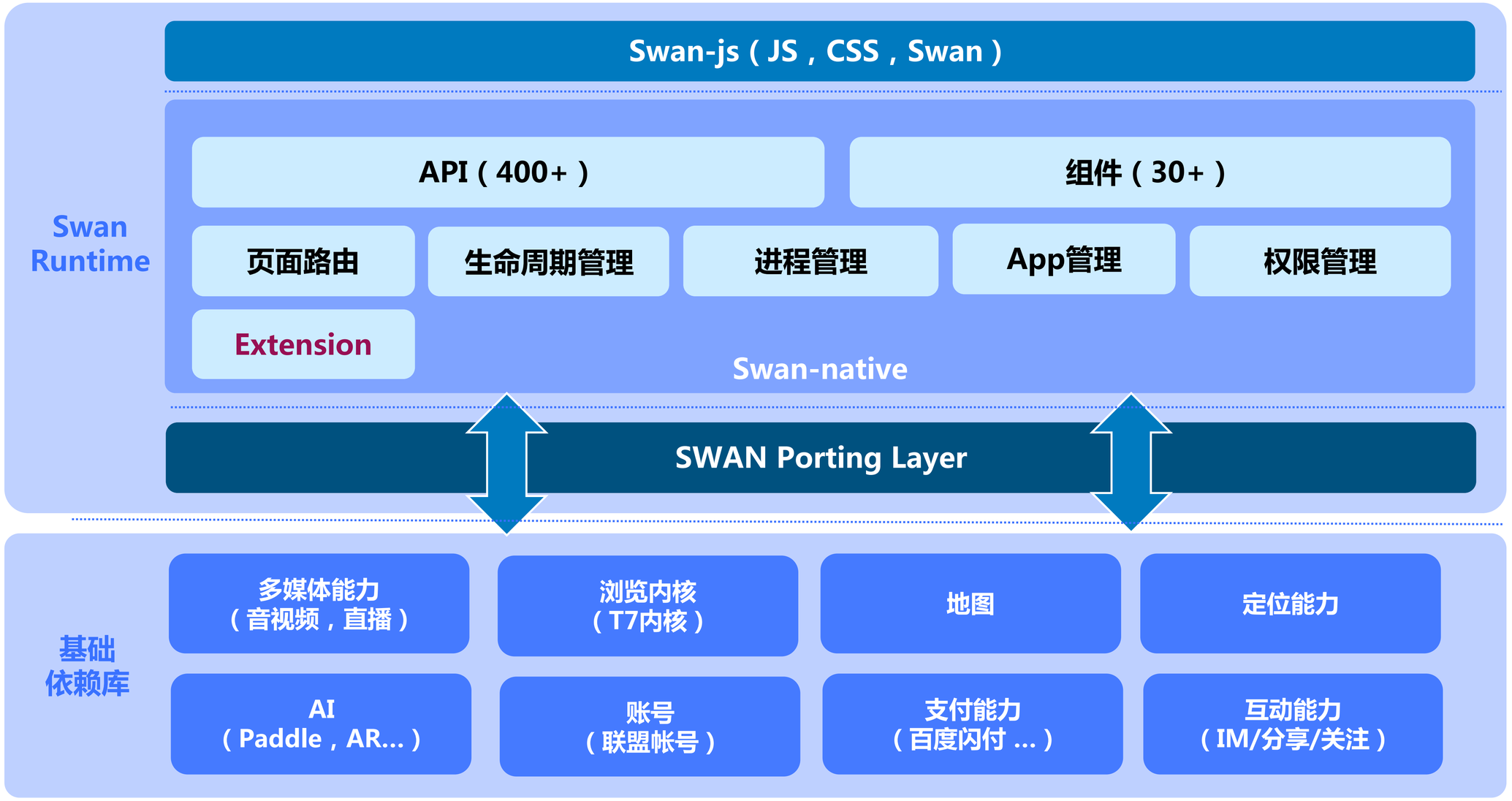
- 2018 AI开发者大会
- 特点：
  - 受限H5+端能力+UI组件

开发运行全流程概览

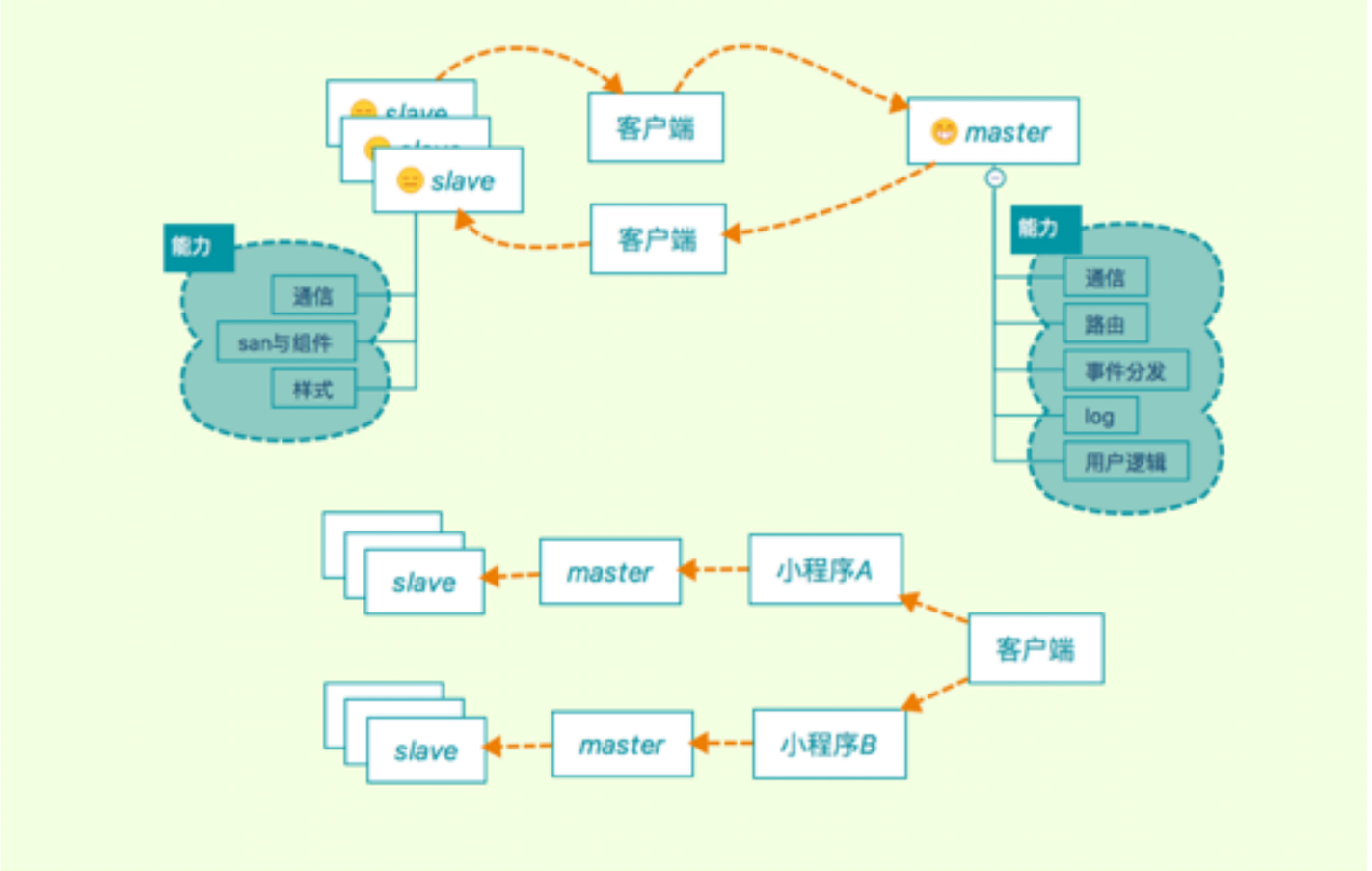

开发者
|__ app.css
|__ app.json
|__ pages
| |__ detail
| | |__ detail.css
| | |__ detail.swan
| | |__ detail.js
|__ app.js
开发
开发者工具
打包上传、
云端托管
小程序B端
审核、
预览、
发布
小程序包
加密传输
meta信息配送
小程序C端
百度App
校验、
安装


sc9Tq1iKawTnj5GhG6i77vzelt4C drwx----
9319 drwx----
components drwx----
images drwx----
pages drwx----
category drwx----
discovery drwx----
favor drwx----
history drwx----
history.css -rw------
history.js -rw------
history.json -rw------
history.swan.js -rw------
hot drwx----
index drwx----
search drwx----
allCusomComponents.swa -rw------
allImportedCssContent.js -rw------
app.css -rw------
app.js -rw------
app.json -rw------
common.js -rw------

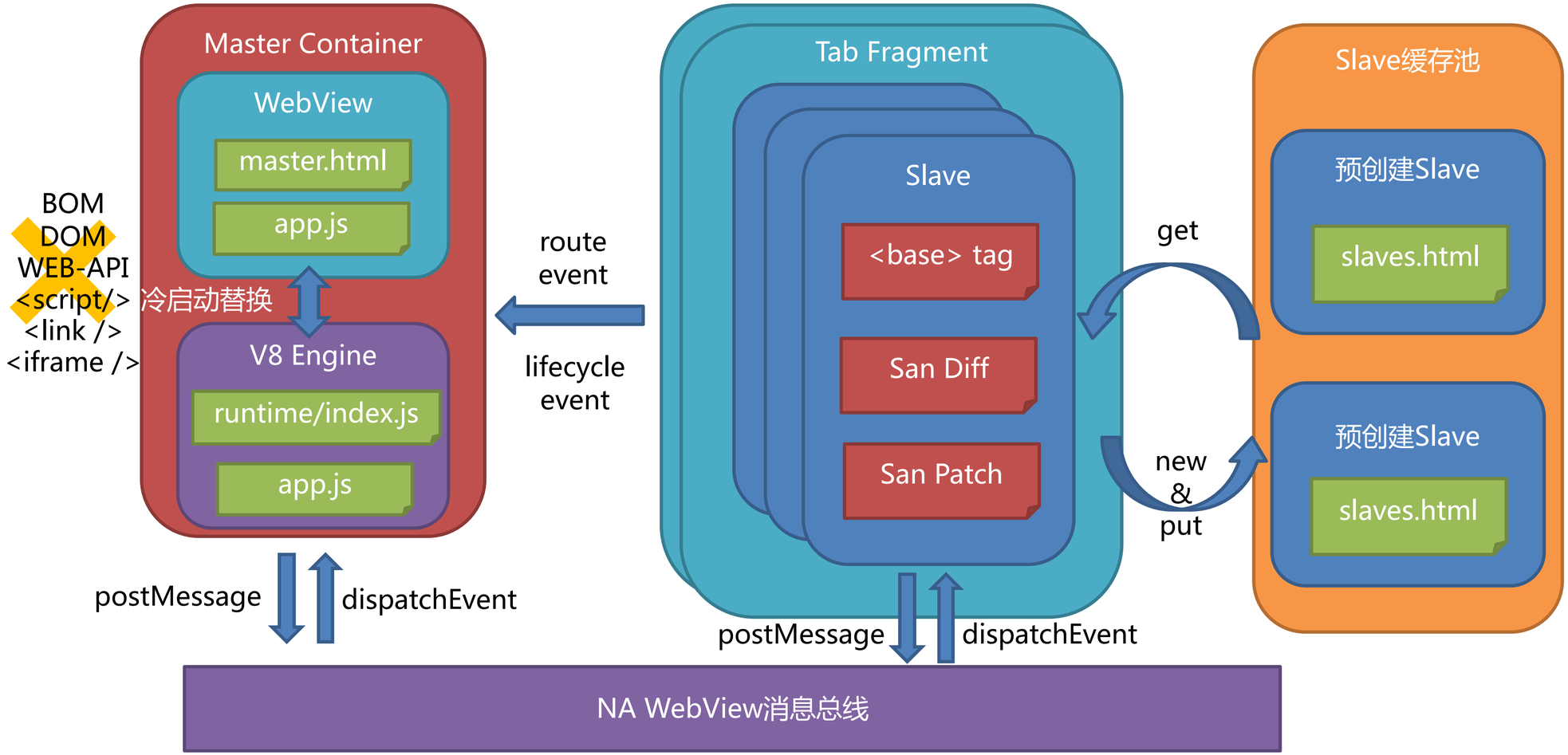
百度智能小程序的框架-SWAN



智能小程序架构-页面栈设计

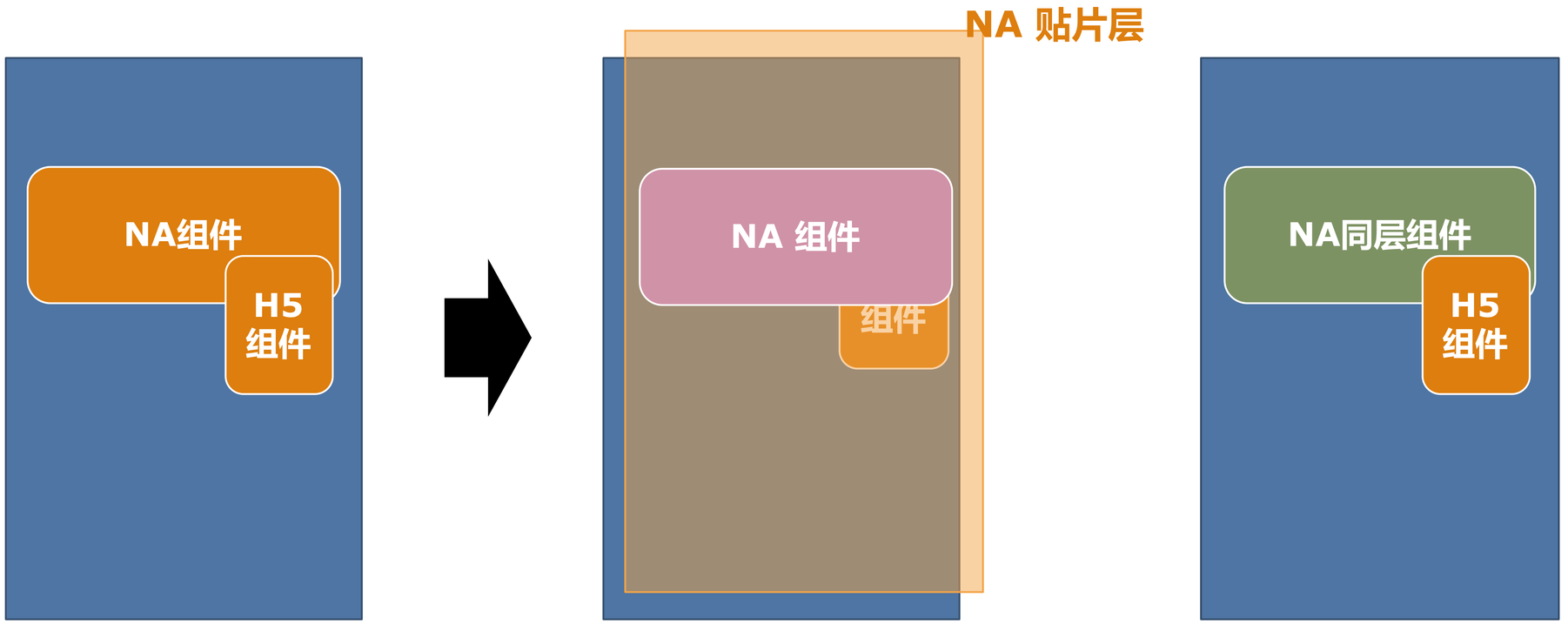


智能小程序架构-页面栈设计



小程序NA组件与界面关系

小程序组件的分类：按实现分类：Native原生组件、H5前端组件

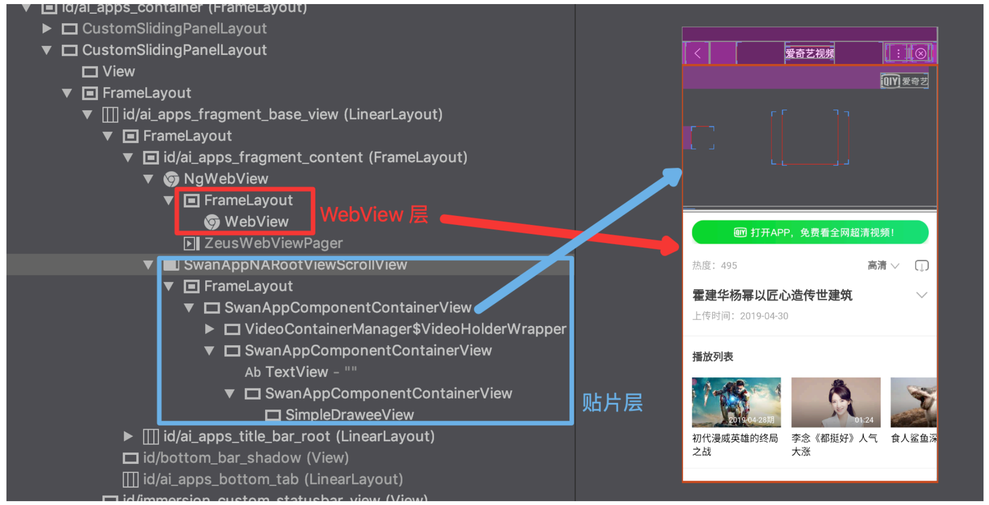


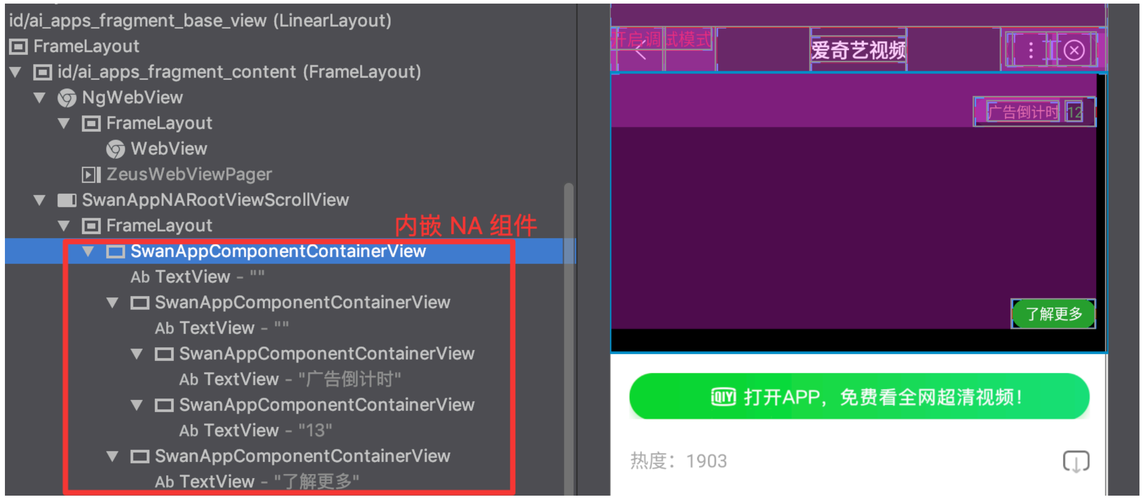
页面设计

NA组件与界面关系1

NA组件与界面关系2

小程序NA组件与界面关系



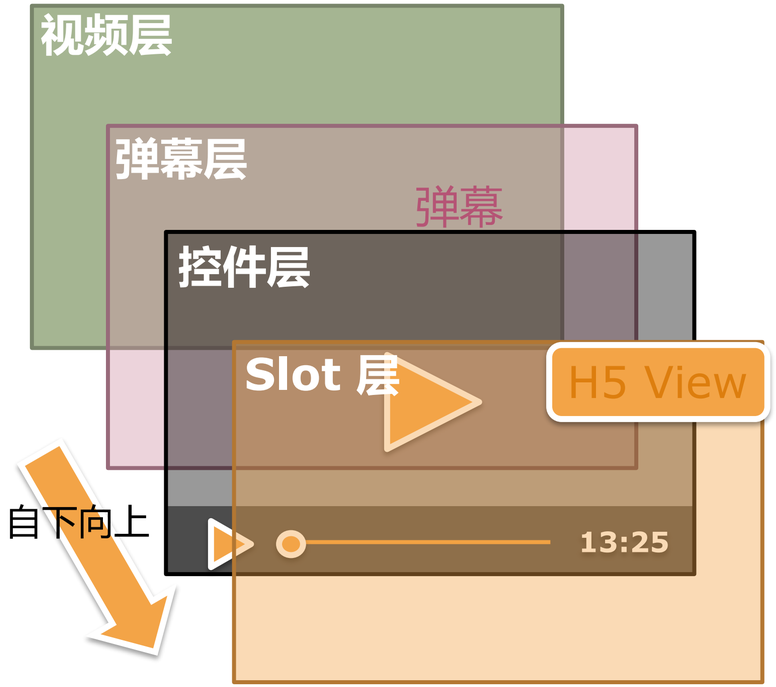


前端 占位，Native悬浮贴片（WebView兄弟节点）
前端属性支持不好如css
前端z-index不生效，无法同层

彻底解决与H5原生元素的 z-index 覆盖问题。
解决Native组件对css属性支持问题。
原生组件无法在scroll-view，swiper等前端组件内嵌套使用的问题。

NA同层组件-视频

同一父 Dom 中



分层视图

视频层

<t7-video-player-type
= `**inline**’>

视频层标记

宿主能力一致性保障

CTS兼容性测试

自动化测试

- 账号体系
- 授权系统
- 地理位置
- 视频播放
- 支付系统
- 网络
- 分享能力
- 下载能力
- 图片查看

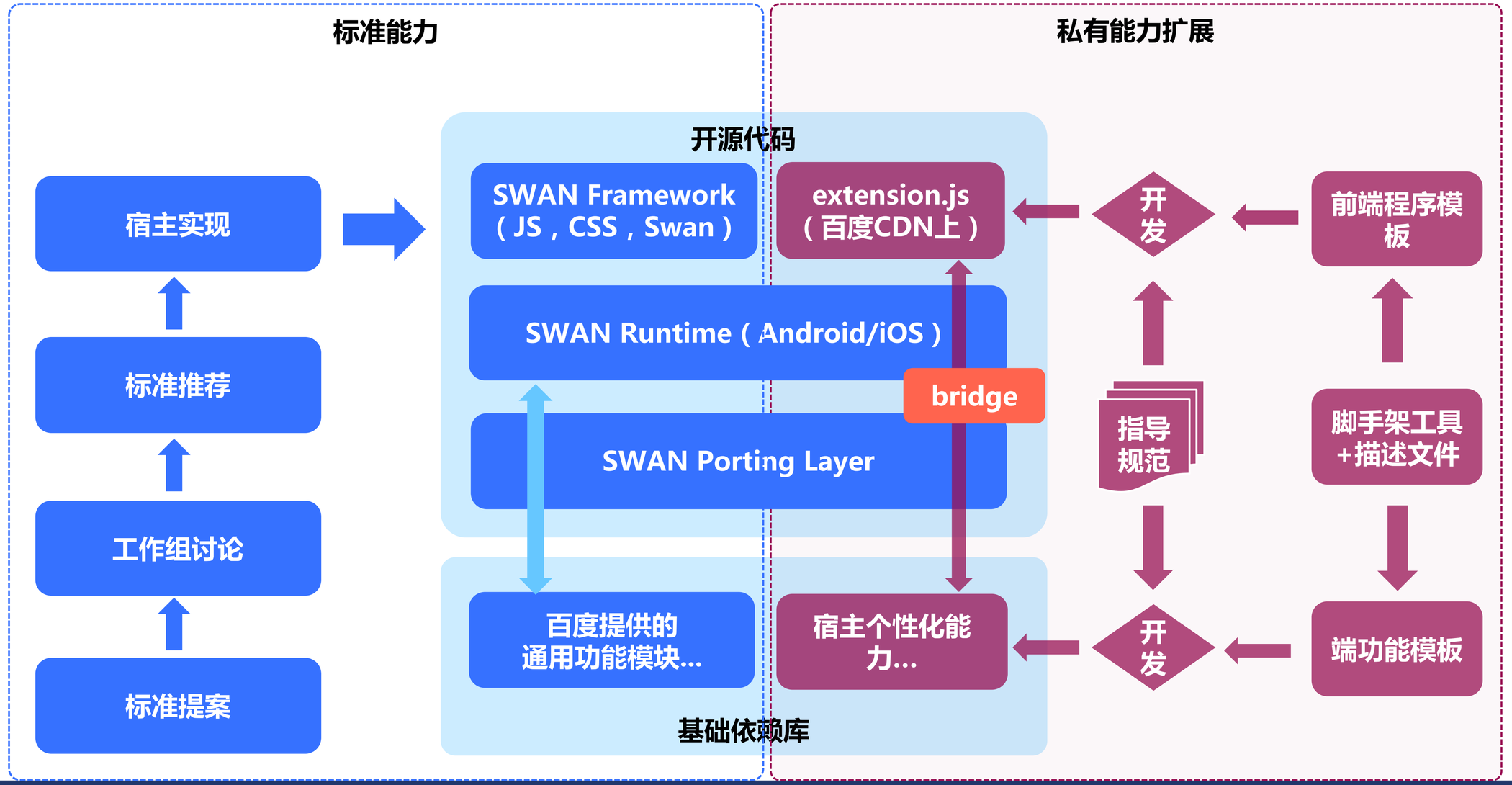
必选能力

手动

- 地图模块
- 直播组件
- 发票地址
- 夜间模式
- AI能力
- 弹幕
- Push

可选能力

Extension机制



章节小结

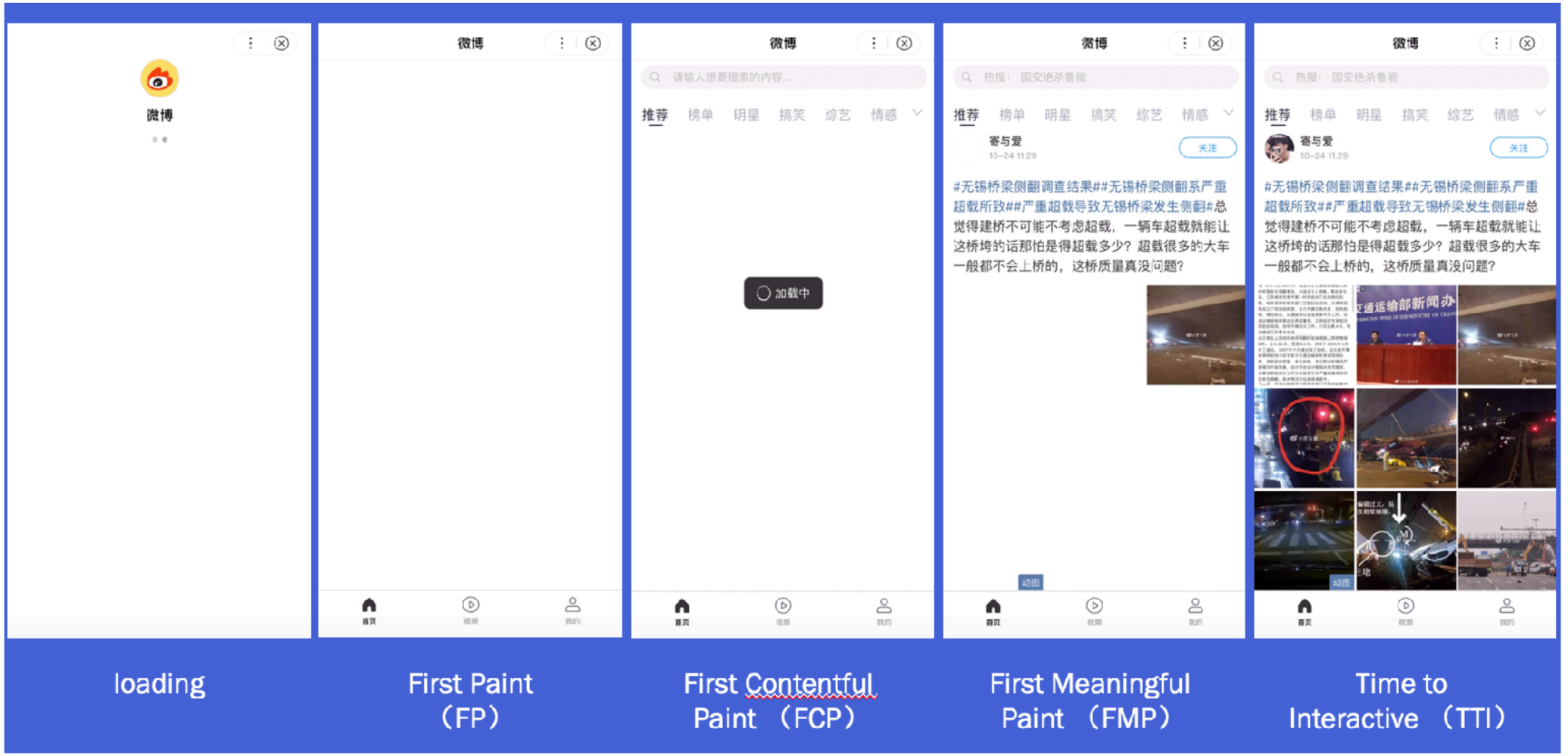
1. 双栈模型：slave(webview)渲染层，负责界面显示；master 逻辑层，执行业务逻辑；

2. 组件：分H5组件、NA组件；NA组件和界面有两种关系，贴片、同层

3. 分发：百度智能小程序可在所有开源联盟宿主内分发

百度小程序框架性能优化实践

百度智能小程序加载分阶段过程

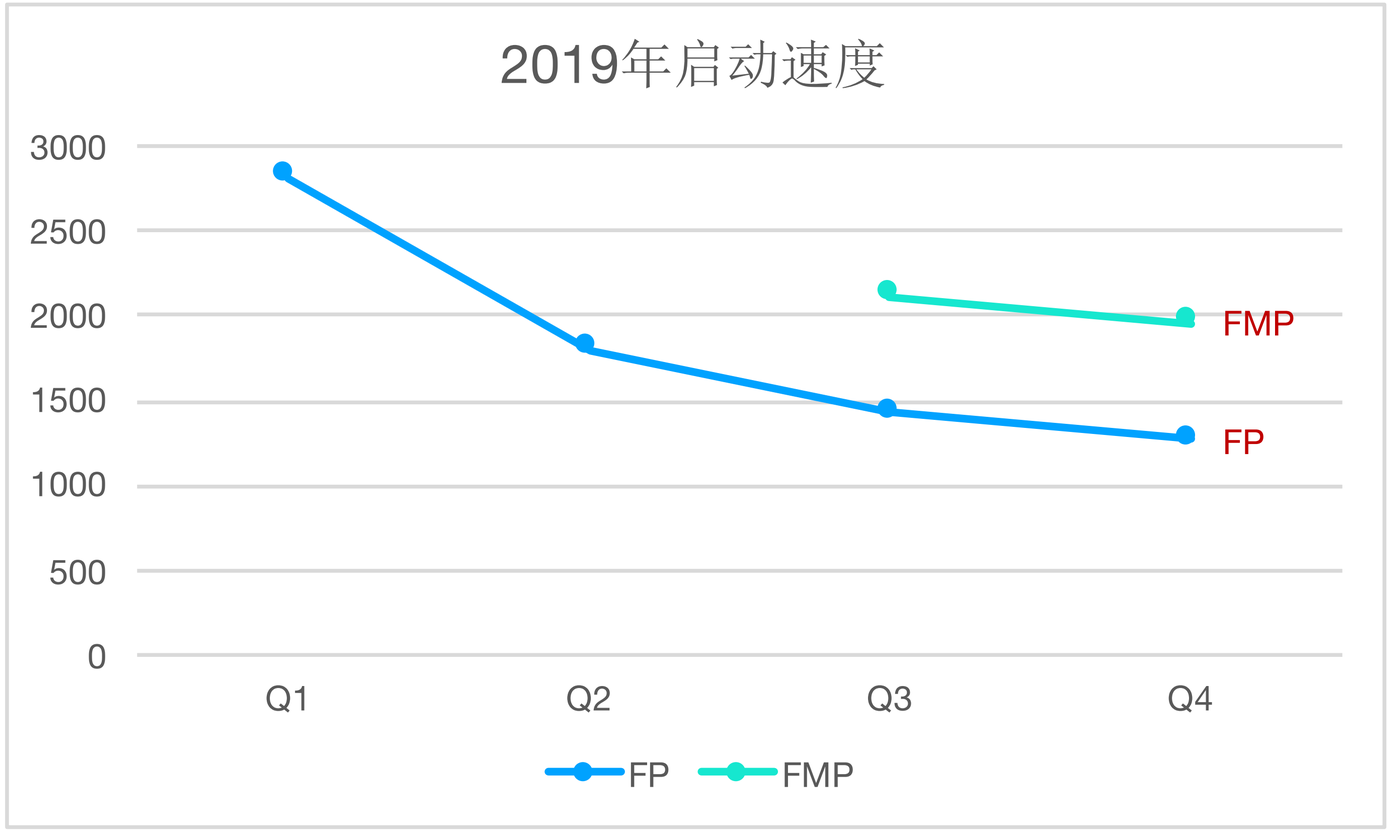


loading | First Paint（FP） | First Contentful Paint （FCP） | First Meaningful Paint （FMP） | Time to Interactive （TTI）

百度智能小程序-性能基线

FMP 1.9s

百度智能小程序-性能历史曲线



小程序启动目标：**无限接近NA体验**

挑战：与目标还很远，而小程序框架层优化越来越接近极限。

小程序自身性能还有巨大优化空间。

百度智能小程序-启动流程

- 调起协议校验

- 小程序包查询
- 小程序包下载
- 小程序包解压

- 小程序NA框架构建
- 小程序前端框架加载
- 业务JS加载
- 生命周期事件派发

- 初始化数据 initData
- 业务数据 setData
- 生成dom节点操作

- WebView渲染 & NA组件渲染

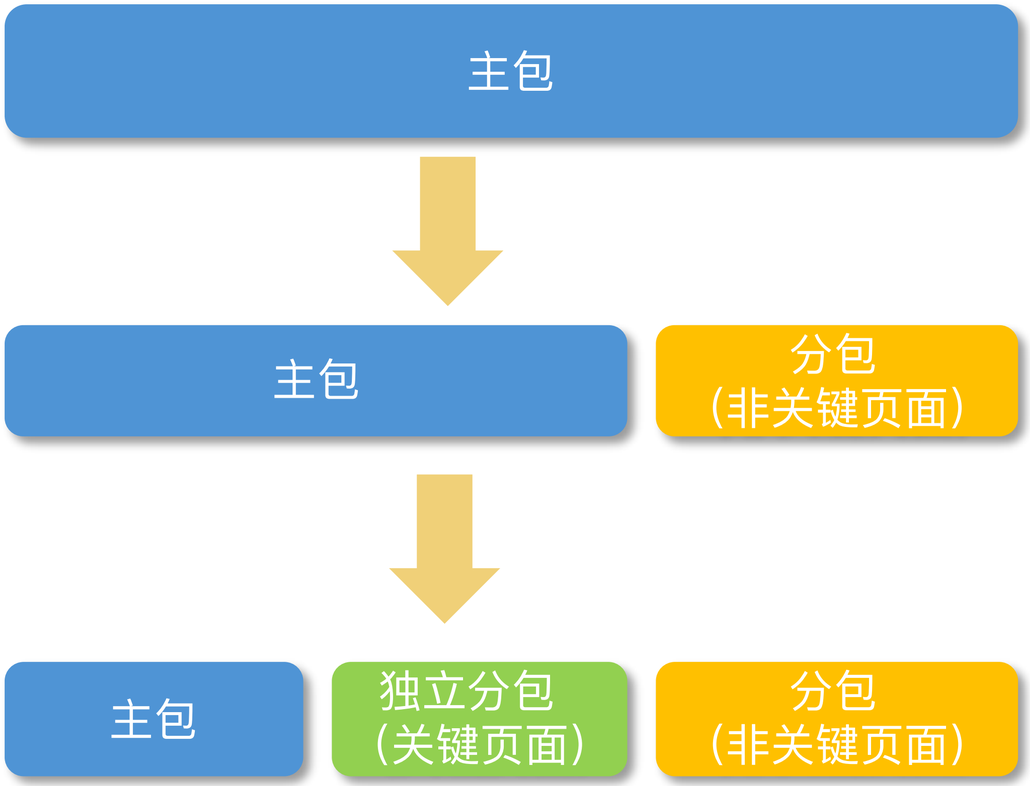
性能-包体优化

建议：保持包体积在1M内

下载小程序包时间占到当次打开60%时间

包体优化- 分包技术 & 独立分包技术

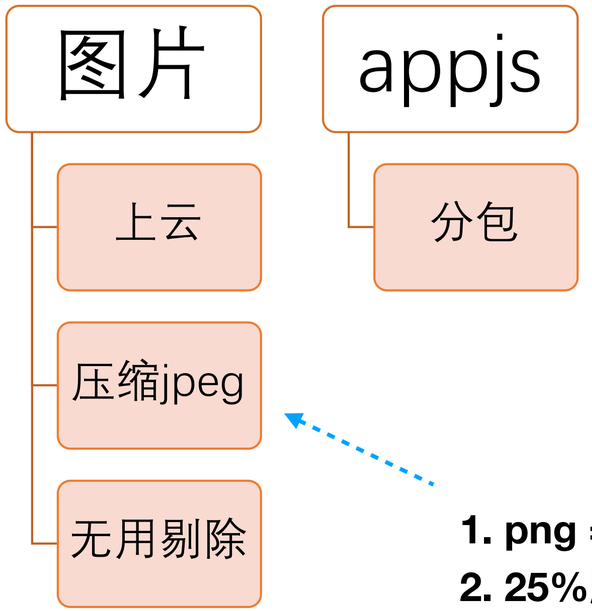
分包技术：包体分主包和分包，分包不可独立运行。
独立分包：每个分包都可独立运行。



- 将非关键页面拆分为分包
- 有效减小主包体积，提升小程序主包的打开速度
- 小程序包的下载耗时，占用整个启动时长的60%左右

- 将关键页面拆分为独立分包
- 独立分包可以脱离主包独立运行
- 独立分包体积小，下载速度快
- 非常适用于feed和搜索等场景

包体优化- 资源压缩



1. png => jpeg (无需透明情况)
2. 25%压缩体积减少90%

单个包控制在1M以内
文件个数控制在200个以内

性能-数据拉取

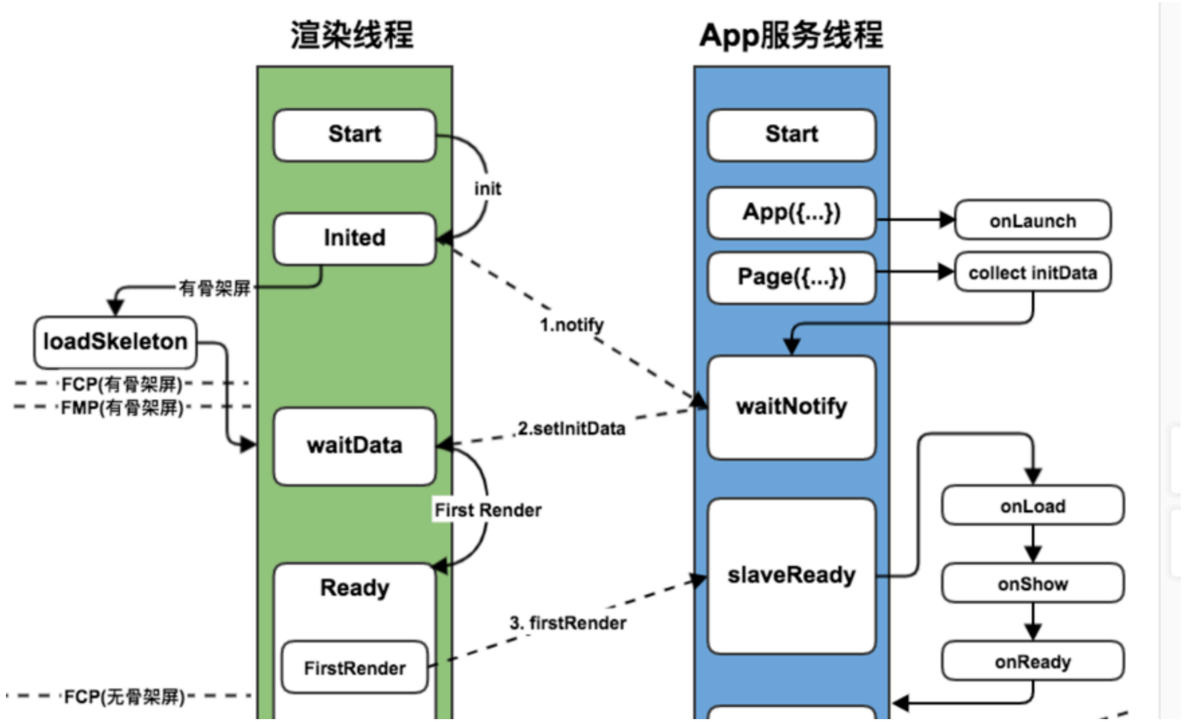
建议：快速让页面有内容，减少用户白屏等待时间

业务骨架屏 pk 框架骨架屏

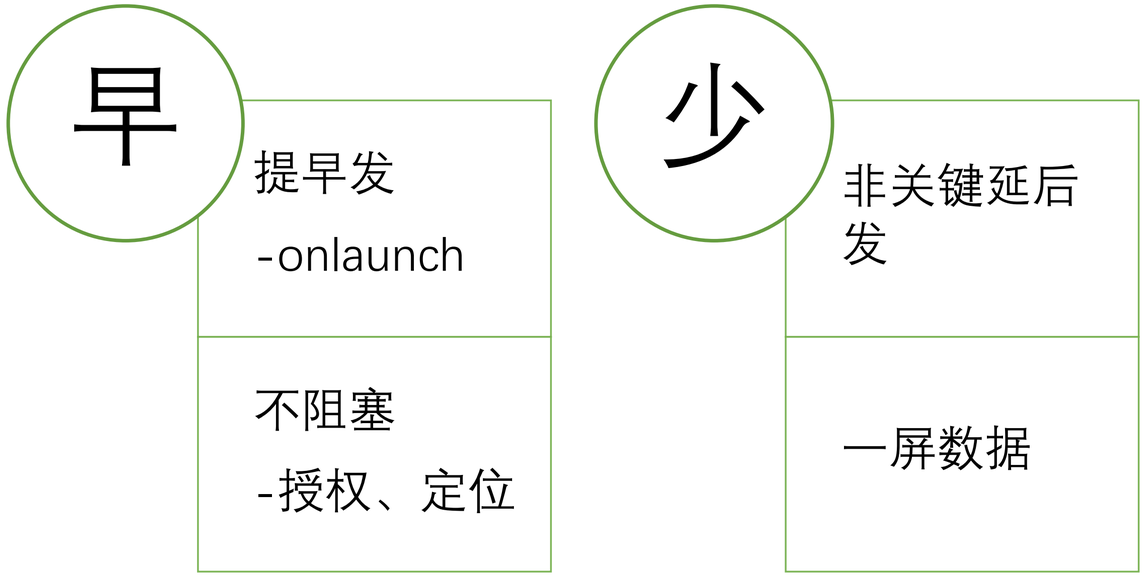
背景：小程序开发者会借鉴H5做法在内容显示前先显示一个骨架屏，
对真实内容显示速度会产生较大影响。如何解决此矛盾?

建议：如果要使用骨架屏，使用框架骨架屏。
业务骨架屏影响真实内容展示速度300ms

业务骨架屏：把骨架屏作为自身业务串行加载
框架骨架屏：按照框架提供的骨架屏机制来实现，提供与业务并行加载能力



request优化



* onLaunch比onLoad早大概200ms
* 减少请求前置条件，如定位（需要高度的定位耗时几秒）

* 网络请求底层线程池管理，大量请求会排队处理
* 减少首屏数据量，80分位 1M数据1~2s传输

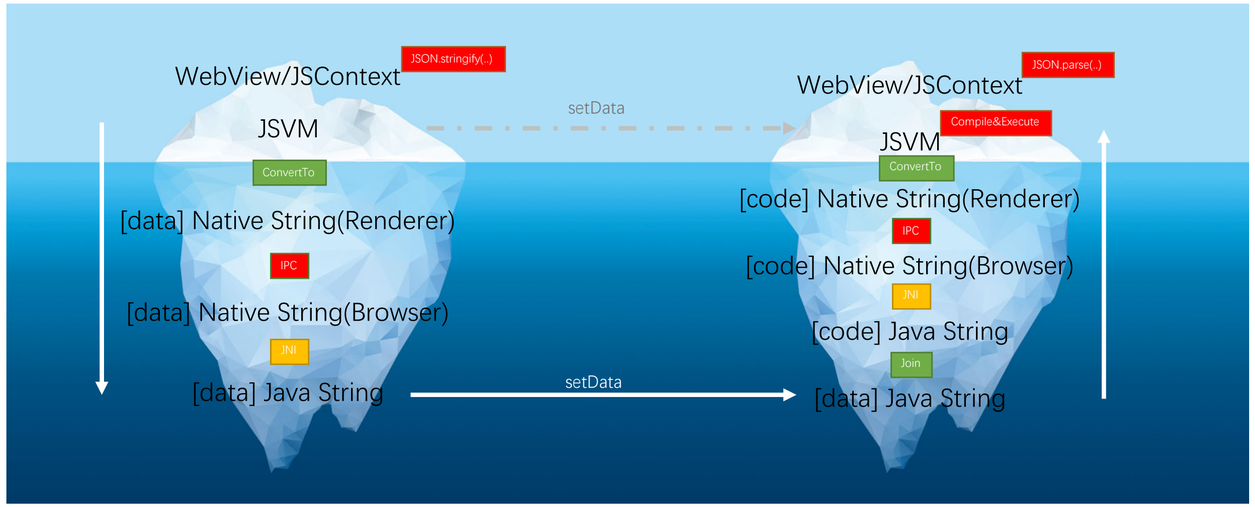
性能-渲染

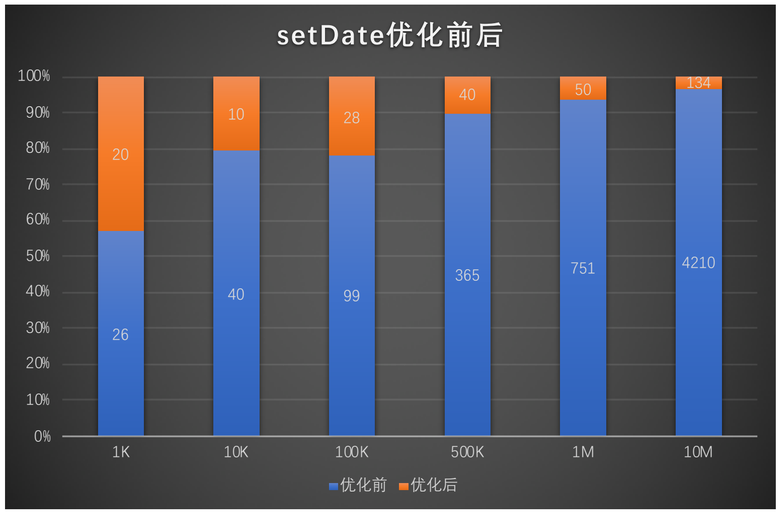
建议：setData操作是较为昂贵的，减少数据量和次数。

setData代价昂贵

小程序运行性能的主要瓶颈之一
- 核心API
- 频繁调用
- 资源开销大，慢（框架层虽然做了优化但依然很耗时）

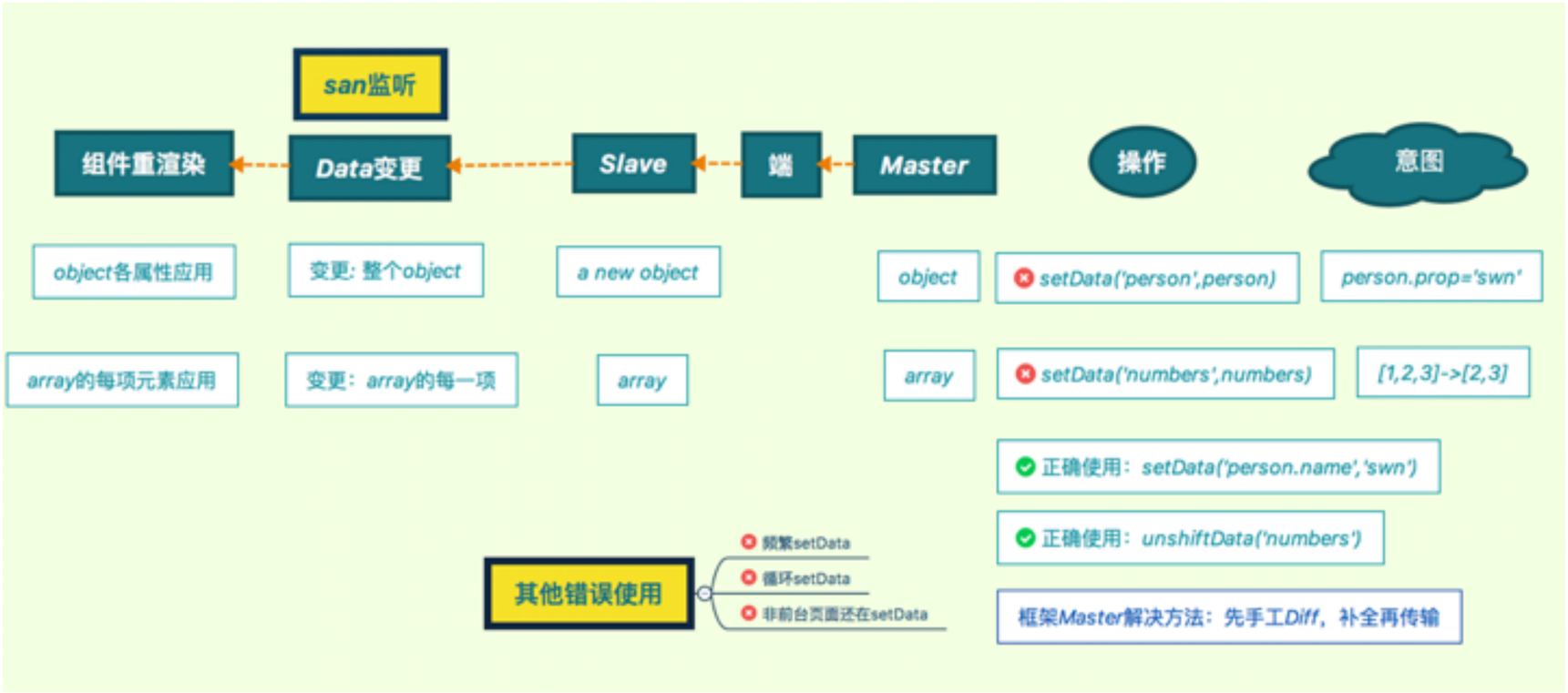
对用户体验影响非常大
- 延迟页面渲染
- 造成操作卡顿



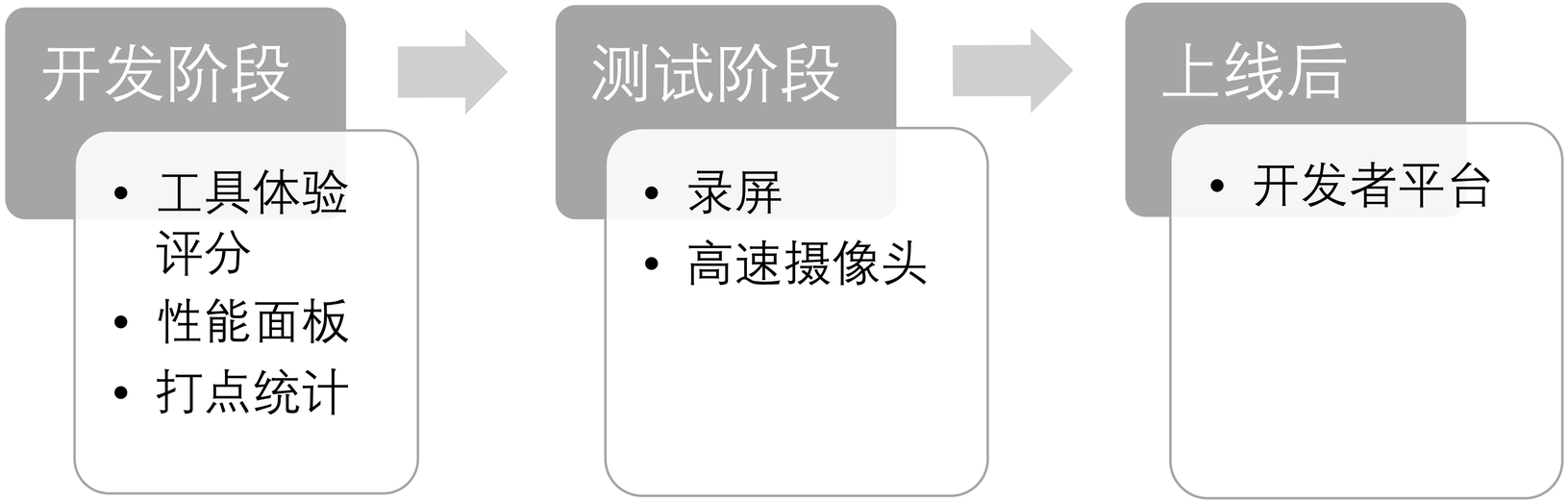


setData使用优化

1. 减少调用setdata次数。goodcase：将多次setData合并成一次setData调用。
2. 减少setdata数据量。 badcase：新一页数据添加上之前页面数据后再调用setData
3. 变量变化只更新变量不更新对象。



性能自查



获取技术支持的官方途径

开发者文档 

社区 

QQ群 

微信群 

章节小结

1. 开发者可从 包体积、数据请求、渲染 3方面去优化性能

2. 包体：1M内。分包技术、压缩图片、无用资源剔除

3. 骨架屏：如果要使用，建议采用框架骨架屏

4. setData：减少频度、减少数据量

官网优化建议：

总结

双栈结构　　NA组件 贴片、同层　　联盟分发

分包技术　　框架骨架屏　　优化setdata使用

THANKS

GMTC
全球大前端技术大会