

饿了么跨端技术的演进、实践与落地

饿了么前端工程负责人 / 刘宇

目录

- 跨端技术背景与演进历程
- 基于小程序跨端的行业现状和思考
- 基于小程序 DSL 的跨端解决方案
- 展望未来

跨端技术背景与 演进历程

跨端，跨的究竟是哪些端？



PC/Mobile/OTT/IoT



Android/iOS/HarmonyOS



微信/支付宝/手淘/抖音



Webview/ReactNative/Flutter

移动跨端技术演进历程



饿了么跨端投放现状 — 概览



饿了么跨端投放现状 — 问题、诉求和策略

渠道问题

渠道差异与限制

- 渠道的运行环境
- 渠道的流量特性
- 渠道的业务定位
- 渠道的管控规则



高度不确定性

- 渠道环境不确定性
 1. 商业能力与产品思路
 2. 能力成熟度与完整度
 3. 研发配套（语法、工具）
- 业务诉求不确定性
 1. 用户特性与诉求
 2. 渠道定位
 3. 业务运营策略



应用复杂度提升

- 业务关联关系
- 团队协作模式
- 功能对齐策略
- 能力感知范围

小程序（主）+ H5（辅）：N 个渠道、N 个业务、N 个应用、N个框架、N个团队

诉求

渠道业务灵活性？
研发体验和效率？



策略

围绕“研发体验一致性提升与复杂应用协作机制改进”双向保障业务高效迭代

饿了么跨端投放现状 — 残酷的现实

理想

推动框架统一
实现一码多端

降本增效



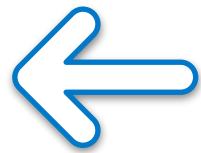
现实



思考

渠道流量
小程序 > H5

影响与成本



决策：以小程序为第一视角来实现多端

基于小程序跨端的行业 现状和思考

业界有哪些面向于小程序的研发框架？

框架对比		Taro	uni-app	Rax	Remax	Chamzleon	mpvue
LOGO							
公司		京东	DCloud	阿里集团	蚂蚁集团	滴滴	美团
DSL		React / Vue DSL	Vue DSL	React DSL	React DSL	类 Vue DSL	Vue DSL
支持平台		12个	15个	7个	7个	8个	1个
实现机制	运行时	不限技术栈，使用一套 Runtime 层，通过模拟浏览器环境，来兼容各种 DSL，运行时较重	小程序端通过提供 vue-runtime 来模拟，运行时	基于 khone 魔改，运行时	基于 react-reconciler，运行时	设计了一套统一框架标准协议，并针对各端分别实现和适配编译产物，运行时相对较重	基于 vue-runtime 魔改，以允许 Vue 运行在小程序端，运行时相对较重
	编译时	通过静态编译方式实现	vue2: uni-app 使用 Webpack 打包，vue3: uni-app 使用 Vite 实现，编译速度更快			基于 Webpack，主要用于编译 CML、CMSS 等框架 DSL，并输出各端编译产物	基于 Vue Compiler 实现和魔改

共同的特点：基于 React 或者 Vue DSL

为什么饿了么选择以小程序 DSL 为基础实现跨端？

上述开源框架能否解决我们面临的问题？—— 并不能



衡量因素



淘支微信小程序运营多年
存量业务多以支付宝或
微信小程序 DSL 编写



渠道业务是重要的流量阵地
重视程度与 App 无差
对性能和体验要求高



每个渠道都是一个小型饿了么 App
业务复杂度高/多业务域协同
应用集成困难/全链路功能衔接困难



以小程序 DSL 构建跨端解决方案

基于小程序 DSL 的 跨端解决方案

如何解决小程序多端编译?

目标

单端



多端

差异
分析

JSAPI 差异

配置差异

文件类型差异

模版指令差异

语法差异

生命周期差异

引用方式差异

功能支持差异

自定义组件差异

.....

衡量
因素

性能和体验

方案
选择

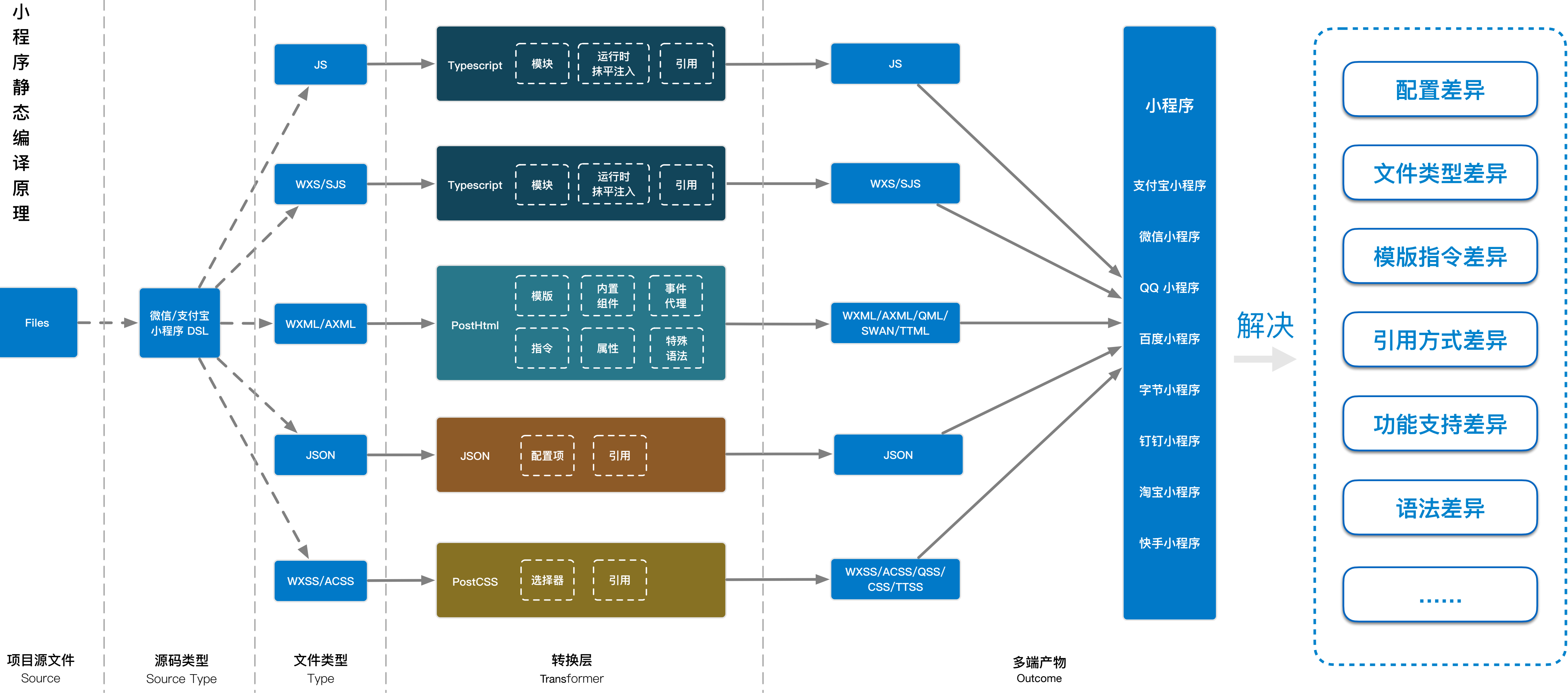
编译时（重）

+

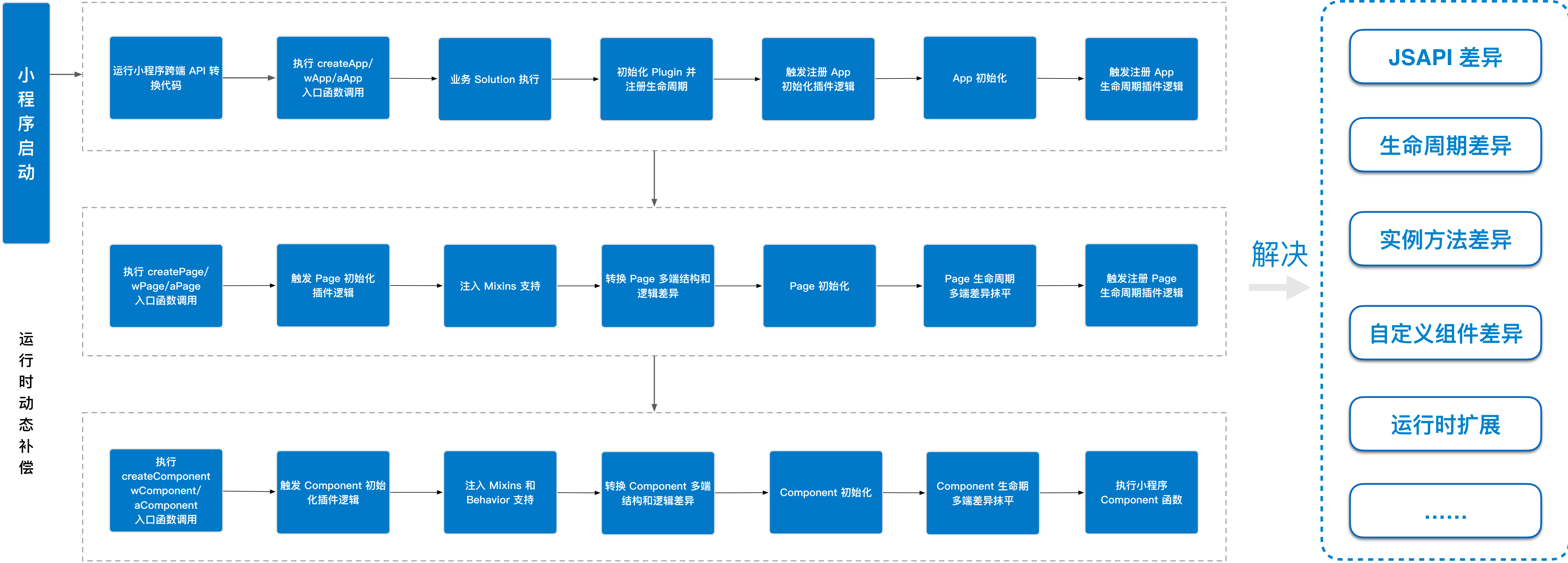
运行时（轻）

如何解决小程序多端编译 — 静态编译原理

小程序静态编译原理



如何解决小程序多端编译 — 运行时补偿原理



如何解决小程序转 Web?

业务
投放

细分渠道或非小程序环境渠道

银行金融渠道、客户端极小包、广告投放等以 H5 来承接

衡量
因素

研发成本和维护成本

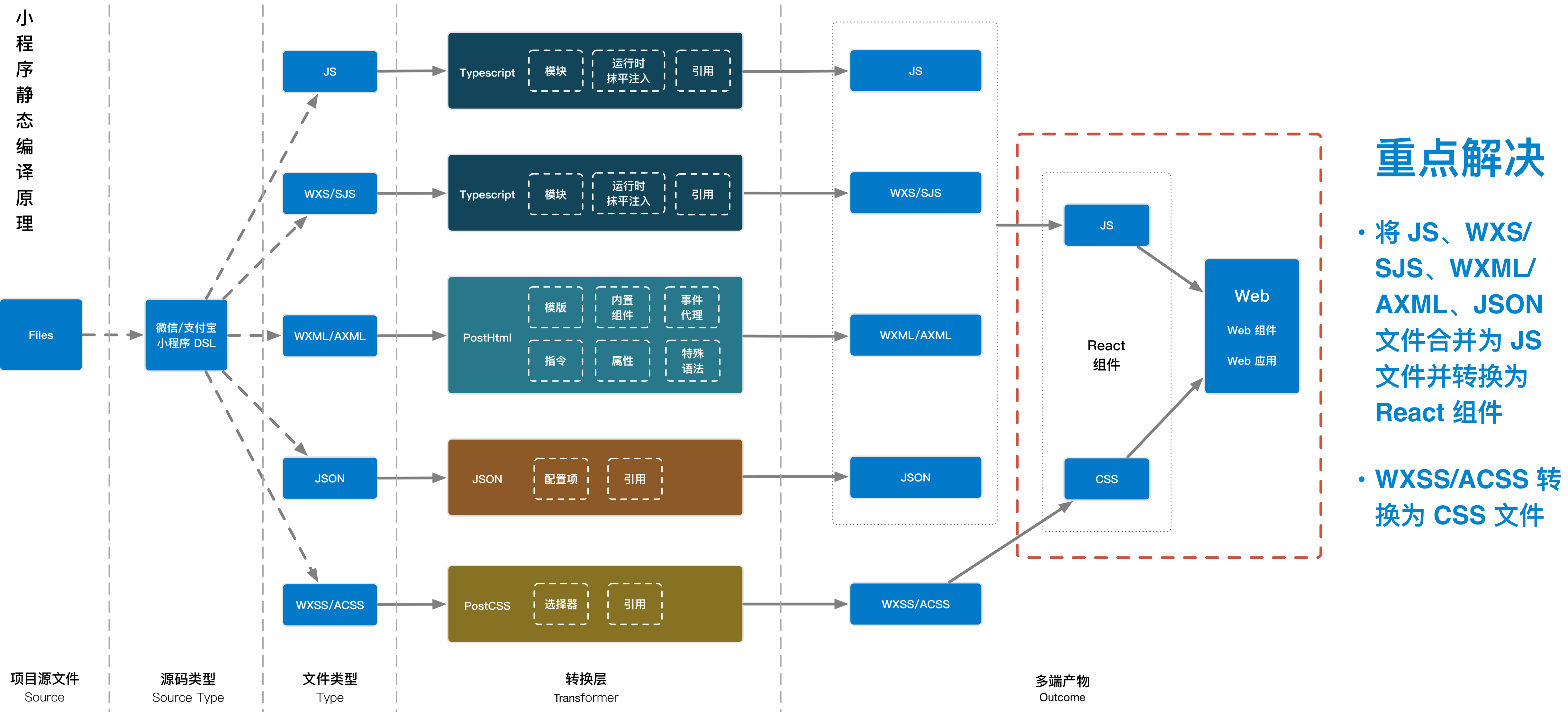
独立维护一套 H5 划不来

方案

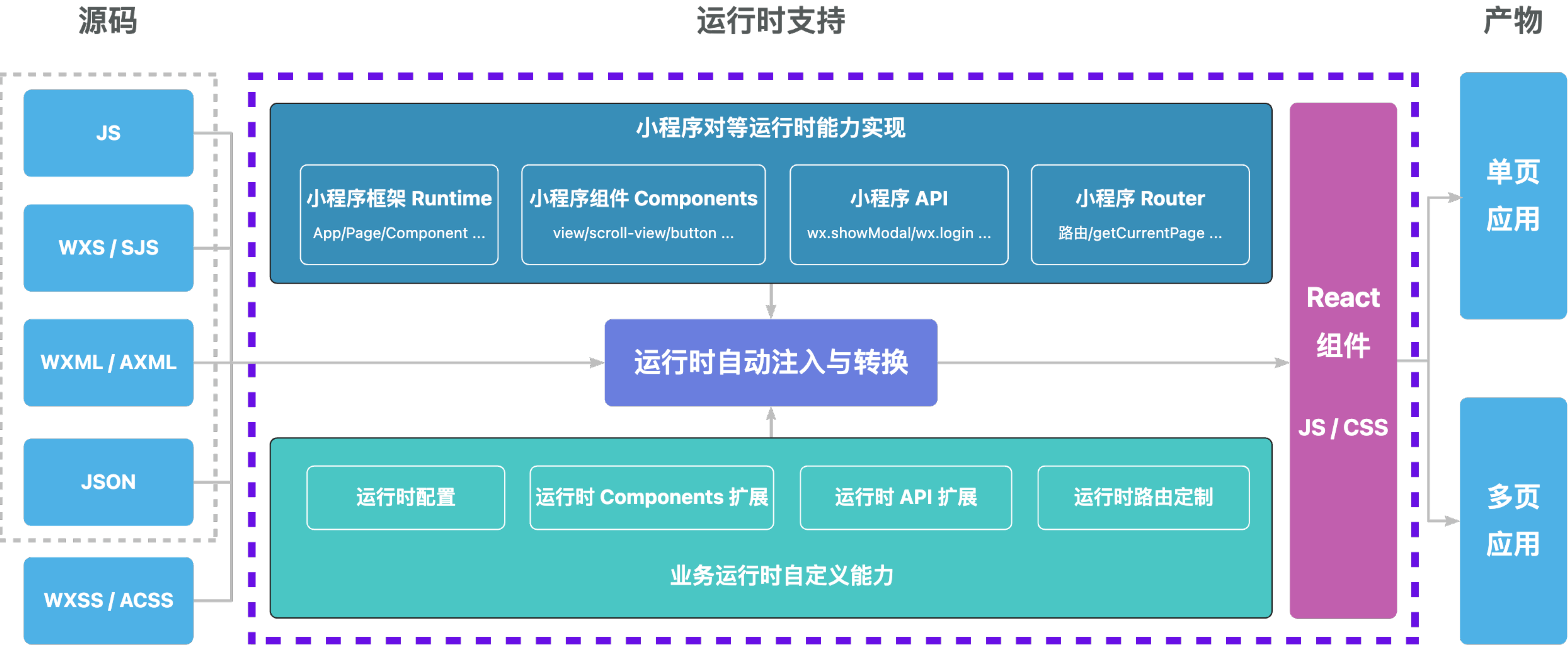
小程序转 Web

编译时 + 运行时

如何解决小程序转 Web — 编译原理



如何解决小程序转 Web — 运行时原理



如何解决多端多形态的问题？

痛点：由于形态差异，各端分别维护一套代码

方案：消除形态差异（业务难以适配的部分）



入淘：支付宝小程序接入淘宝作为小程序插件



入微：支付宝小程序接入微信作为小程序分包



入支：微信分包接入支付宝作为小程序插件



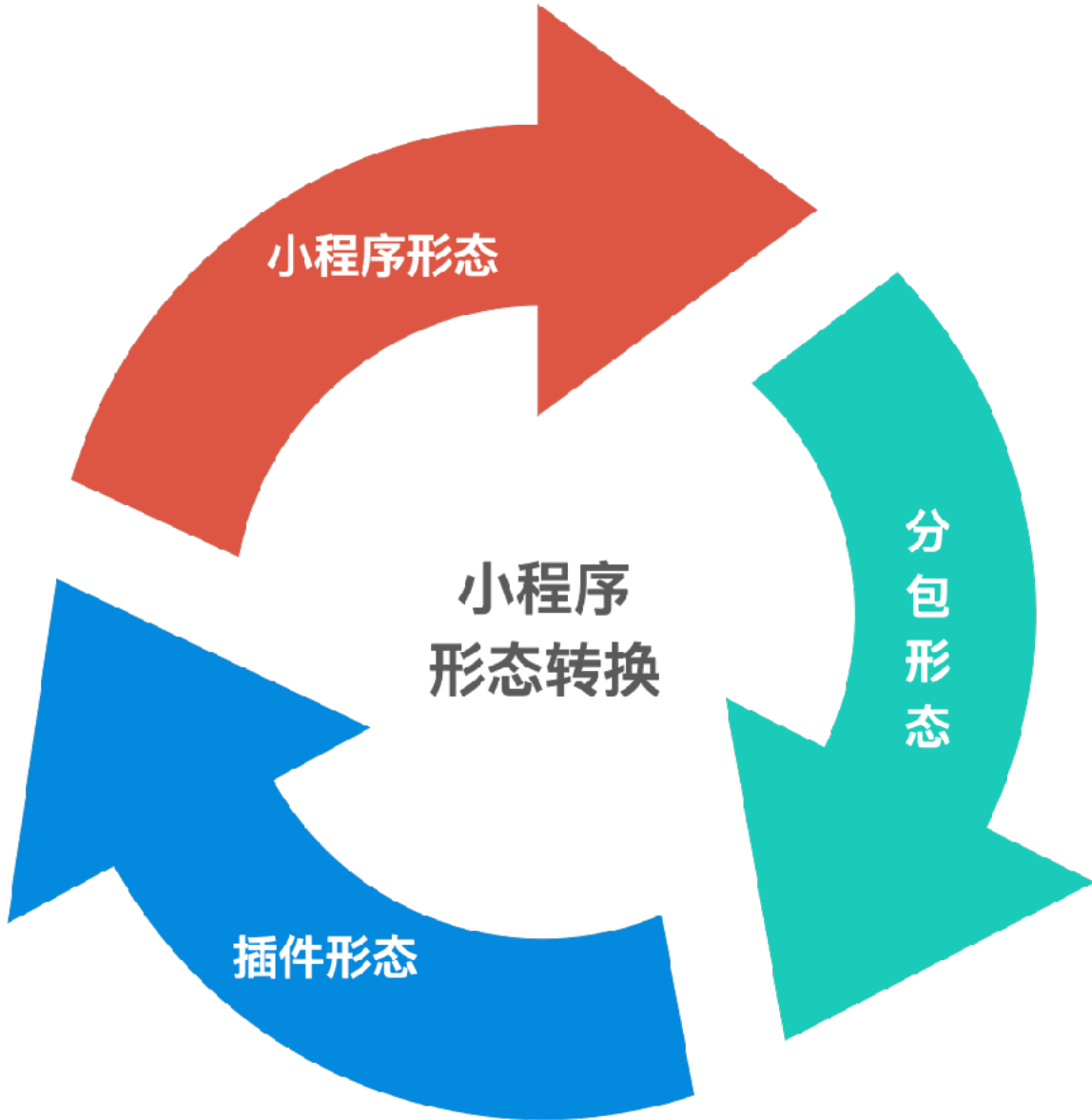
入饿：支付宝小程序插件接入饿了么作为小程序

.....



- getApp 差异
- App 应用生命周期差异
- 全局样式差异
- NPM 使用限制
- 接口调用限制
- 路由差异

.....



形态差异解决办法举例 — 生命周期与全局函数

插件或分包无应用（APP）生命周期怎么办？
getApp 方法调用缺失以及转换后的 **getApp** 实例冲突怎么办？

```
App({
  onLaunch(options) {
    // some logic
  }
})
```

转换前的 app.js 文件

```
var G = require('./subpackage.global.js');

G.init({
  onLaunch(options) {
    // some logic
  }
})
```

转换后的 app.js 文件

模拟 App 初始化

```
注入 subpackage.global.js 文件

var $app;

module.exports.getApp = function() {
  if ($app !== null) return $app;
  if (typeof getApp === 'function') return getApp();
};

module.exports.init = function(app) {
  if ($app !== null) return;

  $app = app || {};

  if (typeof getApp === 'function') {
    $app.__proto__ = getApp();
  }

  if (typeof $app.onLaunch === 'function') {
    var fn = wx['getLaunchOptionsSync'];
    $app.onLaunch(fn ? fn() : {});
  }
};
```

模拟调用 onLaunch 生命周期方法

```
Page({
  onLoad(options) {
    // some logic
  }
})
```

转换前小程序页面

```
自动引用 app.js, 确保 app.js 的自动初始化
require('../app.js')
var G = require('./subpackage.global.js');

(function (getApp) {
  Page({
    onLoad(options) {
      // some logic
    }
  })
})(G.getApp)
```

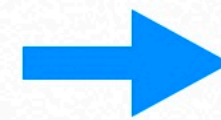
转换后的小程序页面

注入 getApp 方法支持

形态差异解决办法举例 — 全局样式

小程序转分包或插件之后全局样式失效怎么办？

```
.banner-wrap {  
  position: relative;  
}  
  
.banner {  
  height: 176rpx;  
  margin-top: 10rpx;  
}  
  
.banner-item-img {  
  height: 176rpx;  
  width: 702rpx;  
}
```

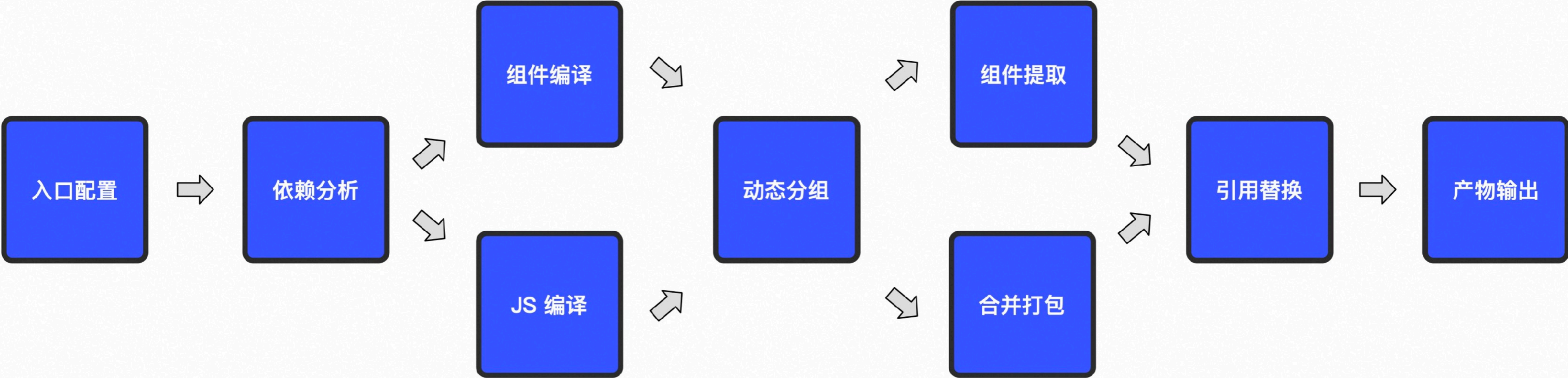


```
@import "../app.wxss";  
  
.banner-wrap {  
  position: relative;  
}  
  
.banner {  
  height: 176rpx;  
  margin-top: 10rpx;  
}  
  
.banner-item-img {  
  height: 176rpx;  
  width: 702rpx;  
}
```

页面和组件样式中自动注入 app.wxss 引用

形态差异解决办法举例 — NPM 构建支持

不同小程序平台对 **npm** 构建支持不同且不友好怎么解？



如何治理“复杂小程序”？

解决应用架构与研发协作问题

特征

- 跨业务领域
- 长周期
- 多团队协同
- 主链路+多分支

表现

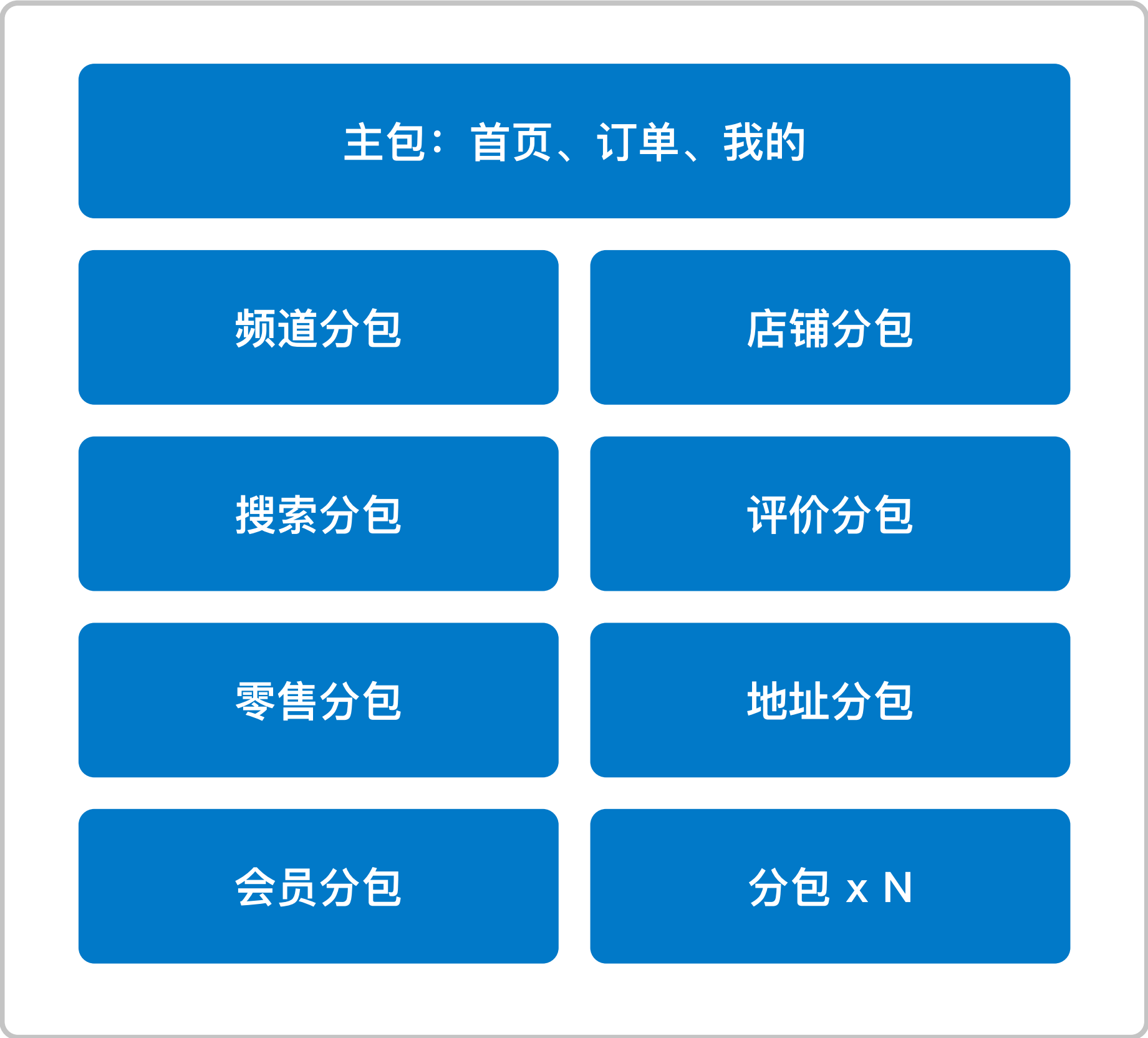
- 应用形态多样
- 业务诉求多样
- 关联业务面广
-

“复杂小程序”的定义

如何治理“复杂小程序” — 业务示例、问题剖析与解法



饿了么微信小程序 业务形态



由 主包+ N个分包组合而成

核心痛点



N个业务相互耦合，无法独立部署，研发效率低



研发流程不规范，业务稳定性较差，发版质量难以保障



所有分包编译完才能预览，单一分包失败导致整体失败

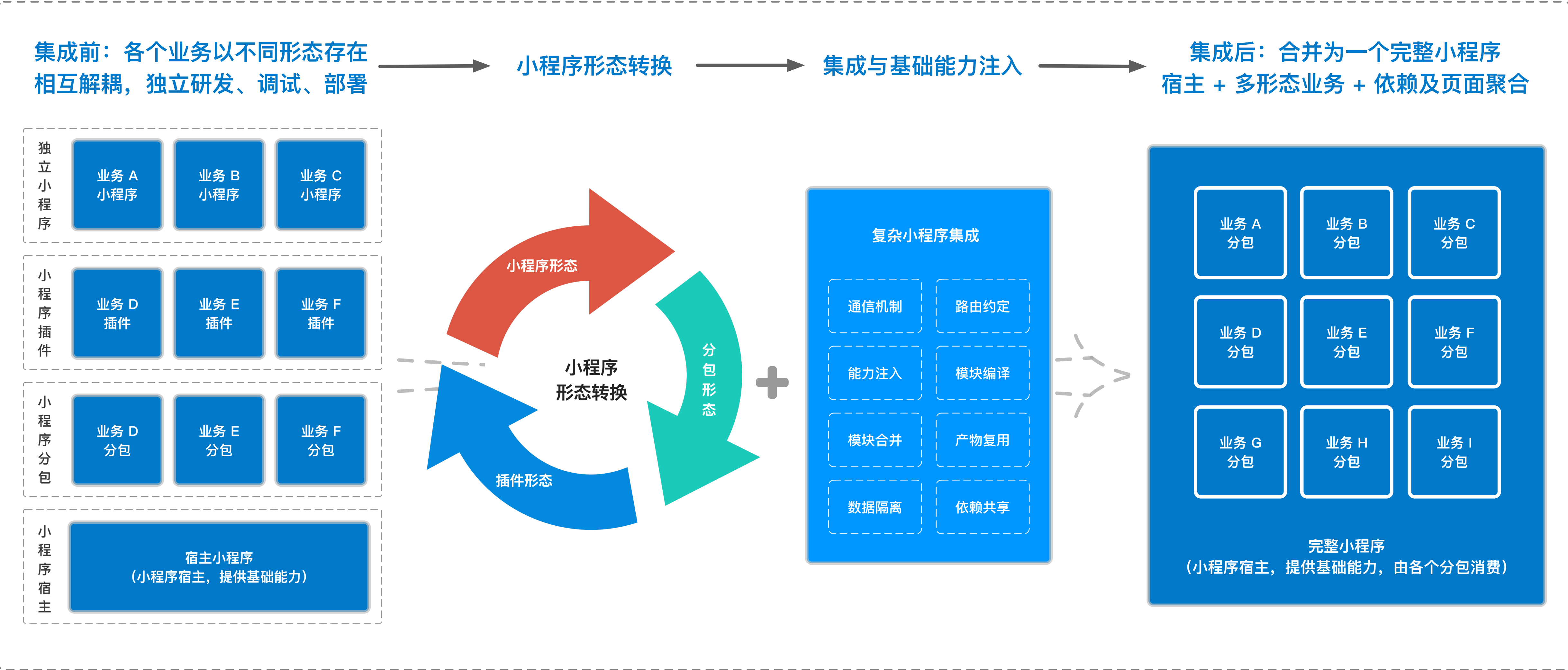


整包大小接近大小上限，存在大量重复依赖

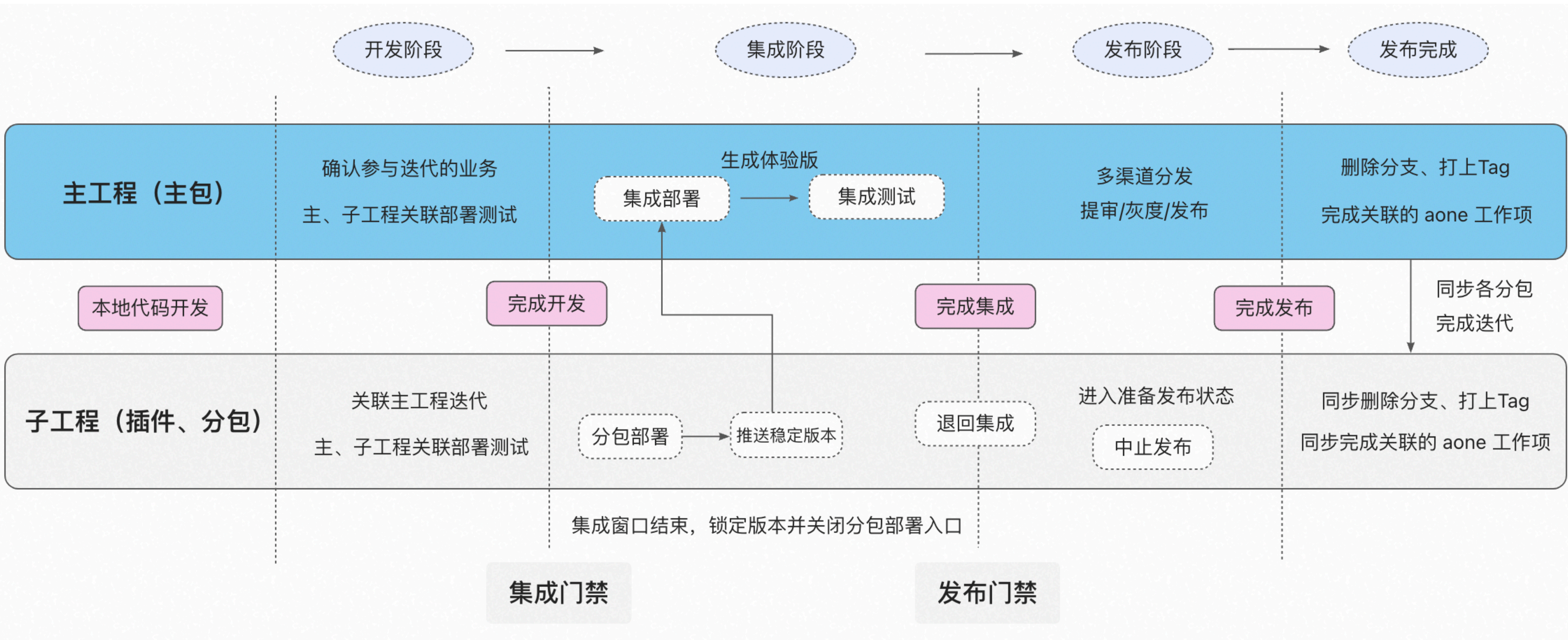
解决方案

线下线上结合的集成研发能力

如何治理“复杂小程序” — 线下集成研发

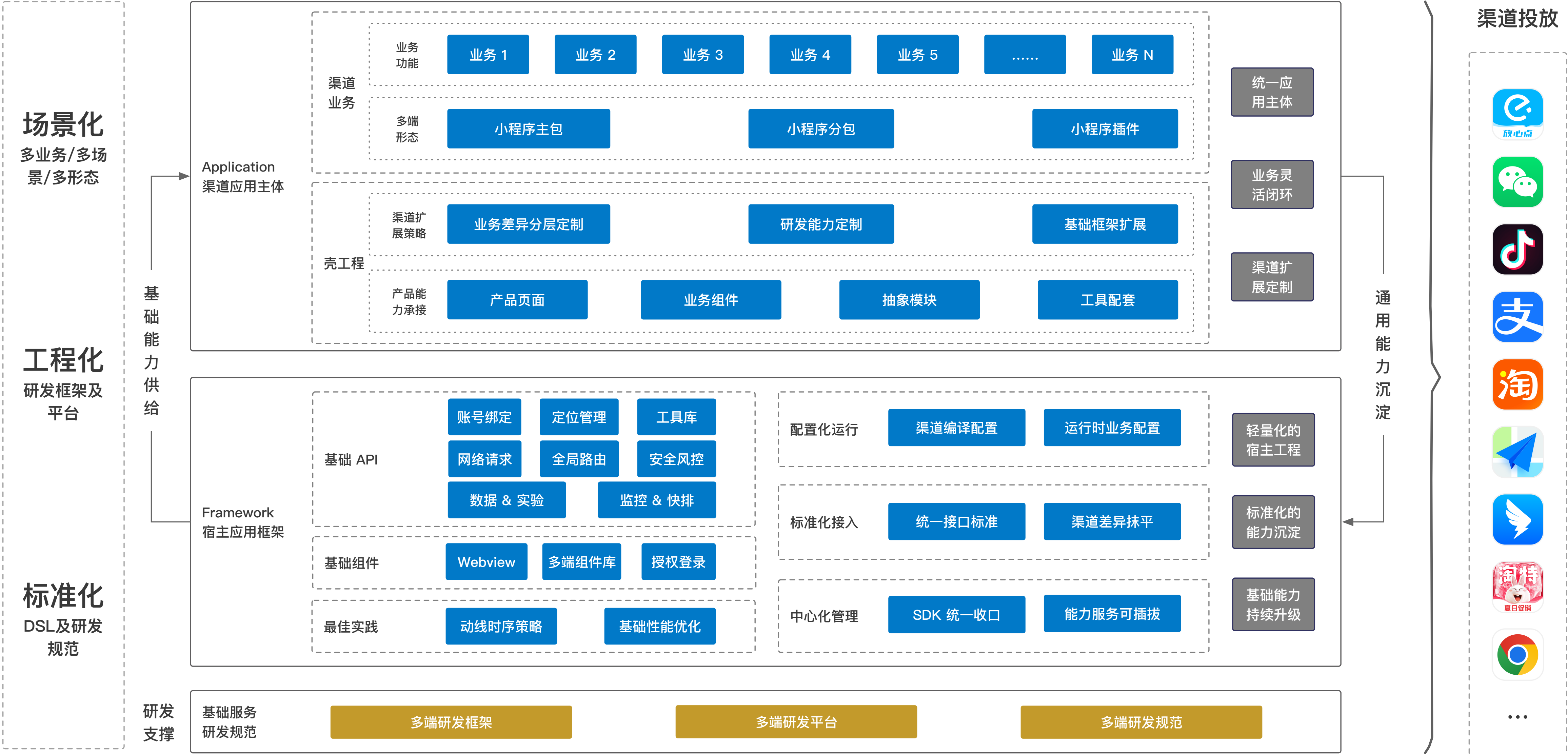


复杂小程序 — 线上研发协作



再进一步，多端业务最佳实践

多端应用分层架构



跨端成果 — 业务效果 & 研发提效



饿了么支付宝小程序



饿了么微信小程序



饿了么淘宝小程序



饿了么淘特小程序



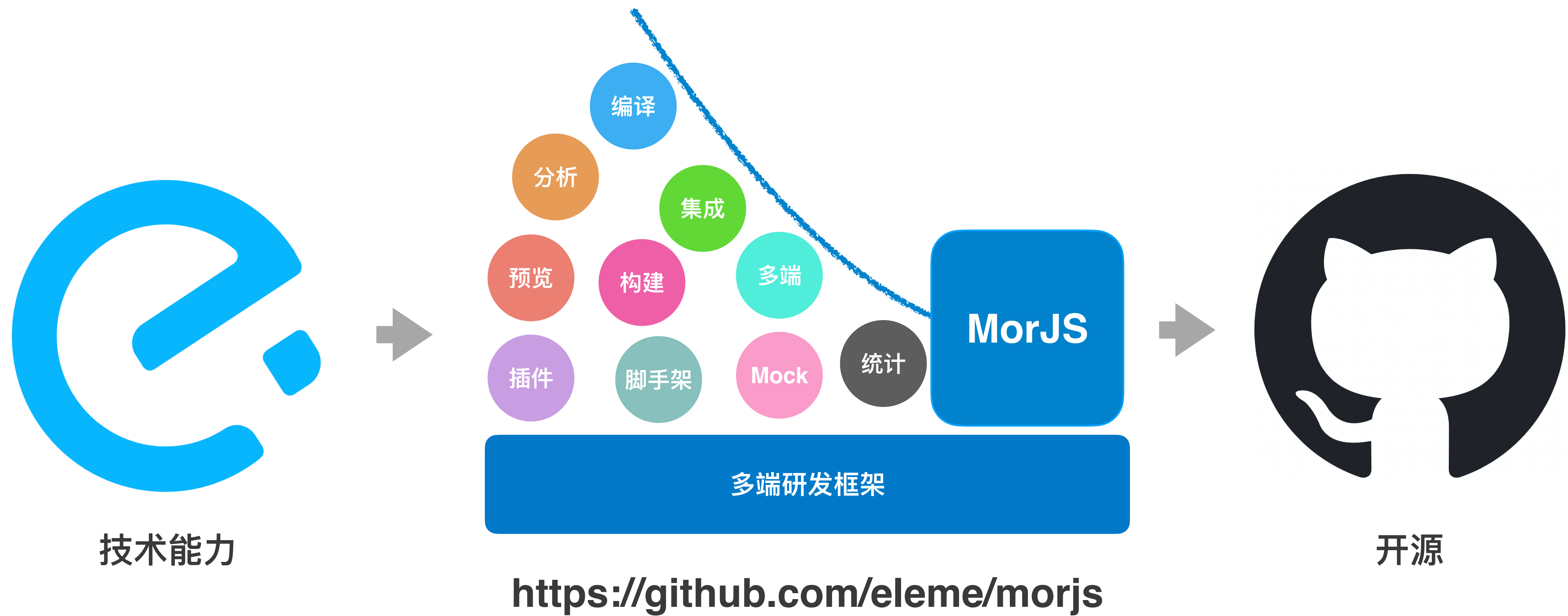
饿了么抖音小程序



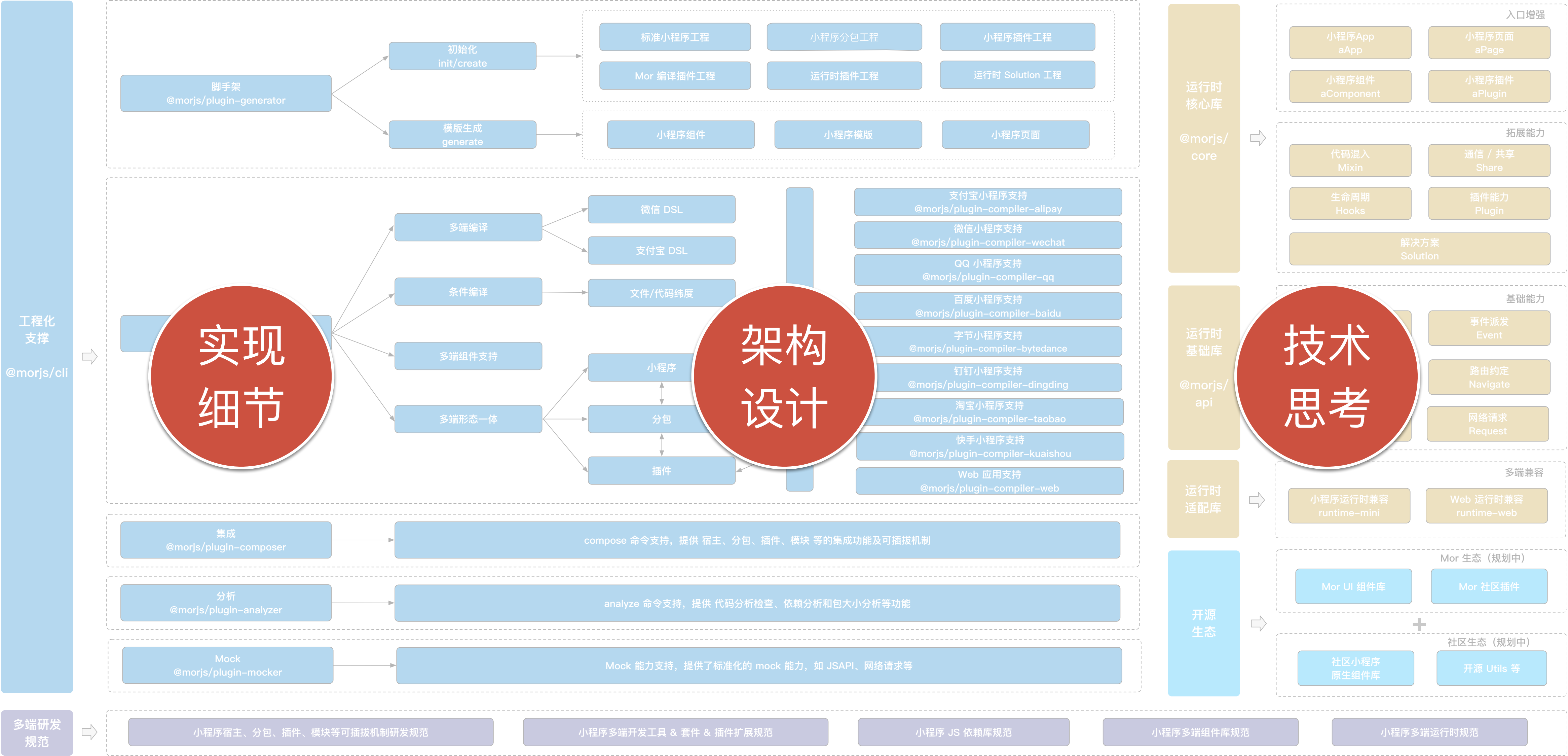
饿了么M站 (Web)

- 研发提效：采用一码多端和集成研发模式的业务平均提效 70%，同构的端越多提效越多
- 多端占比：饿了么内部 85%+ 的多端业务基于该方案实现多渠道业务研发和投放
- 业务覆盖：涵盖了饿了么全域等各个业务板块

能力沉淀 — 饿了么自研 MorJS 多端研发框架



能力沉淀 — 饿了么自研 MorJS 多端研发框架 — 架构呈现



能力沉淀 — 饿了么自研 MorJS 多端研发框架 — 支持情况



微信小程序 DSL
支付宝小程序 DSL



小程序 / 小程序插件
小程序分包 / 小程序多端组件



微信/支付宝/百度/字节
快手/钉钉/手淘/QQ/Web

能力沉淀 — 饿了么自研 MorJS 多端研发框架 — 特性介绍



可扩展

MorJS 实现了完备的生命周期，并使其插件化，MorJS 内部功能也全由插件完成。此外还支持插件和插件集，以满足功能和垂直域的分层需求。



开箱即用

MorJS 内置了脚手架、构建、分析、多端编译等，仅需一个依赖即可上手开发。



企业级

经饿了么内部 100+ 公司项目的验证，值得信赖。



大量自研

多端组件打包、文档工具、请求库、数据流、复杂小程序集成、小程序形态转换等，满足日常项目的周边需求。



多端支持

支持一键转换为各类小程序平台及 Web 应用，节省双倍人力。



面向未来

在满足需求的同时，我们也不会停止对新技术的探索。比如 多端扩充、高性能渲染方案 等。

能力沉淀 — 饿了么自研 MorJS 多端研发框架 — 使用案例

微信小程序



饿了么



饿了么智能服务



蜂鸟跑腿



饿了么超市



口碑



组件库示例



饿了么拼团



阿里本地生活联盟

支付宝小程序



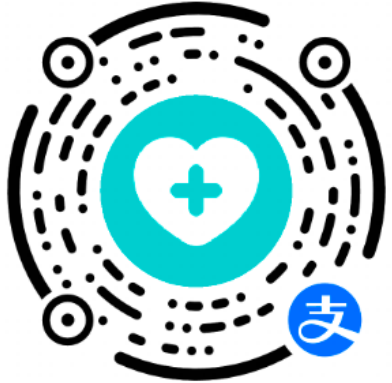
饿了么



饿了么超市



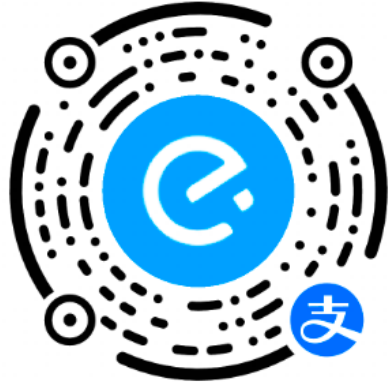
饿了么果园



医疗口腔



蜂鸟跑腿



饿了么外卖下单

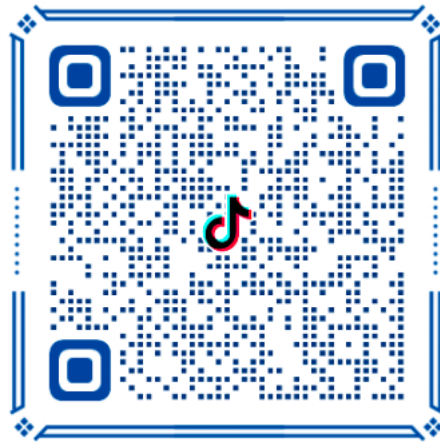


饿了么智能服务



饿了么拼团

抖音小程序



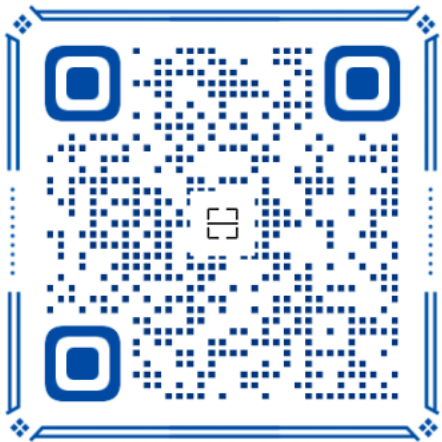
饿了么

淘宝小程序



饿了么

H5



饿了么M站

能力沉淀 — 饿了么自研 MorJS 多端研发框架 — 用户原声

当前市面最接地气的小程序框架 👍

效果还不错!

MorJS 考虑支持下快应用么?

赞! 帮我解决了微信小程序转 H5 的问题

是否 KPI 项目, 会长期维护么?

支持社区组件库么?

对比 Taro 和 UniApp 的优势是什么?

有状态管理 (Store) 支持么?

该项目是否有明确计划长期迭代? #80

是否KPI项目? 无恶意, 纯粹担心没几天就不维护了, 如果不维护, 是否愿意交给社区志愿者之类的?

✓ Answered by lyfeyaj

先说结论：MorJS 不是一个 KPI 项目，是饿了么内部面向多端研发的首选及主要框架，被大量的项目所使用，会长期且持续的迭代和升级。

非 KPI 项目

大量项目使用

长期持续迭代

历经数
年打磨

我们并不热衷于追求 DSL 的优雅，也不热衷于追求 DSL 的通用性，我们更倾向于在小程序 DSL 中追求渐进增强和无感升级，这也是我们选择小程序 DSL 来实现多端的原因。

能力沉淀 — 饿了么自研 MorJS 多端研发框架 — 社区服务



欢迎使用钉钉
扫码加入

群号：29445021084

展望未来

未来与愿景

MorJS 框架

面向未来

愿景



提升多端可用度



完善社区组件库兼容



扩展编译目标平台

成为小程序多端
研发的基础设施

想一想，我该如何把这些
技术应用在工作实践中？

THANKS