

快手商业化前端建设

蓝晓斌

前端工程师



自我介绍



蓝晓斌

2009 ~ 2012  百度

2012 ~ 2019  新浪网

2019 ~ 至今  快手

(商业化业务部)

“快手商业化前端” “建设”

3 种能力

- 业务支撑能力
- 赋能业务的能力
- 应对变化的能力

2 个阶段

- 团队组建
- 业务稳定

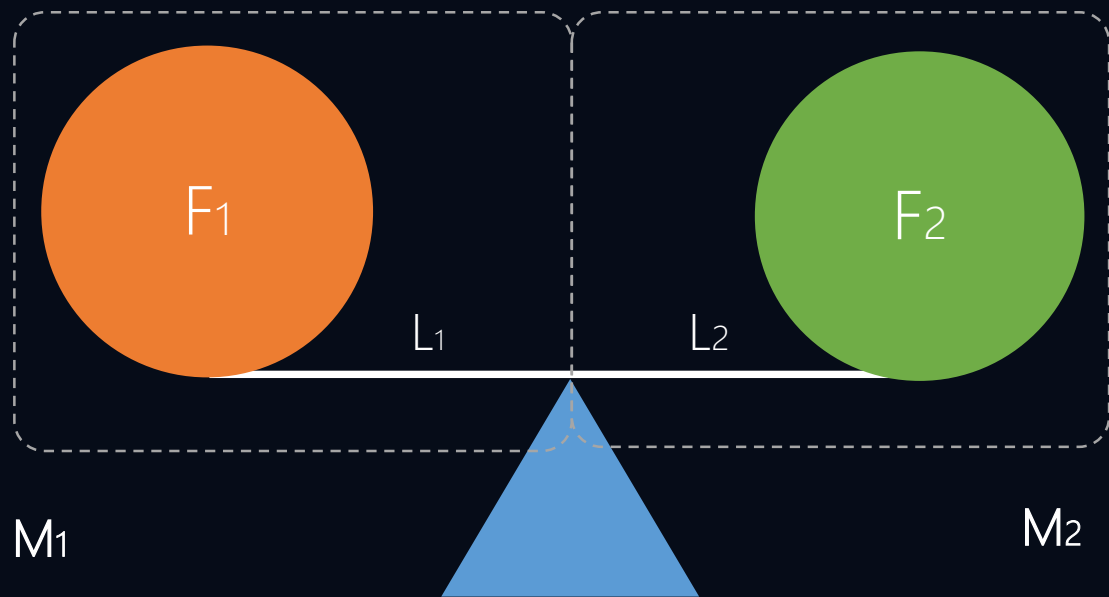
1 个总结

- 投入产出是广义的，能力建设是扩大投入产出的路径

35 分钟

力的公式

平衡即： $M_1 = M_2 \rightarrow L_1 \times F_1 = L_2 \times F_2$

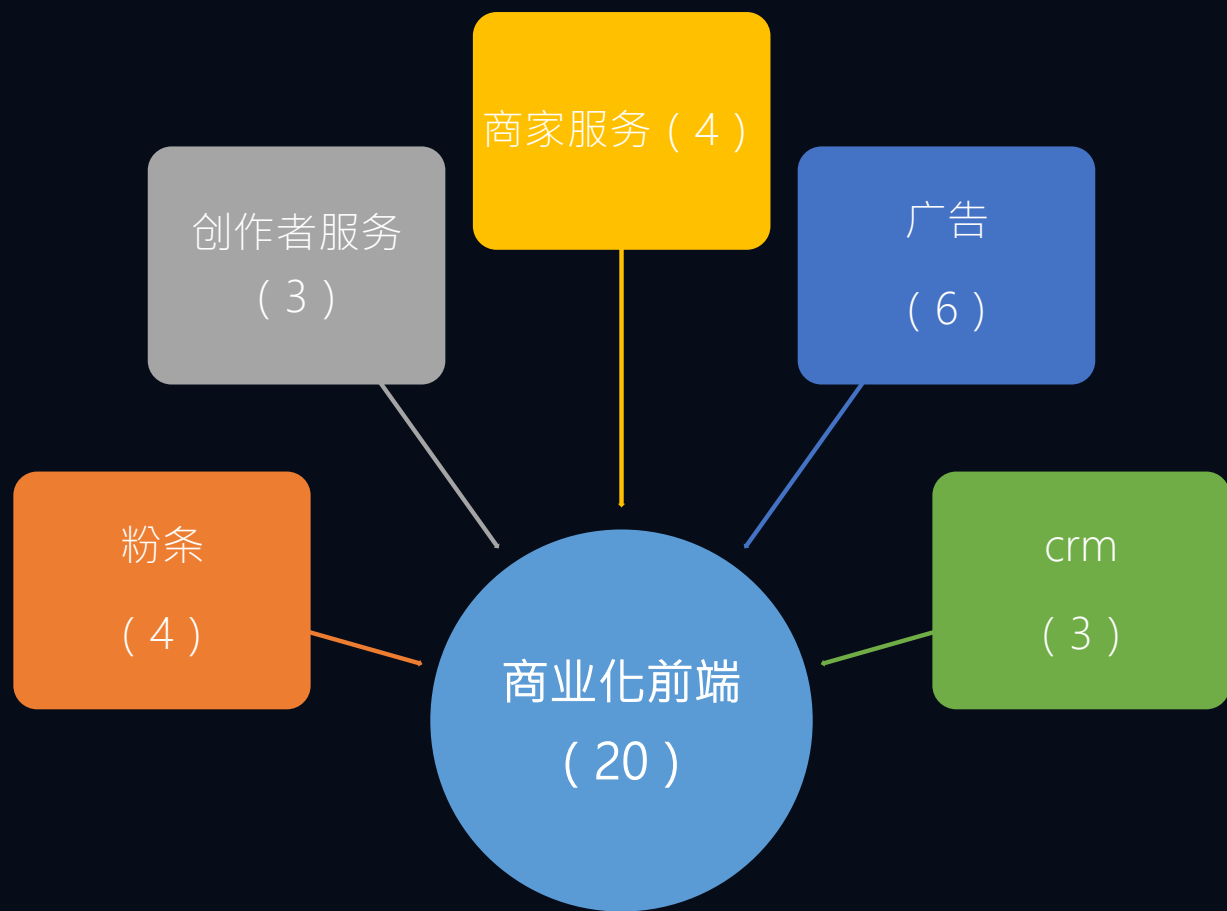


$$\begin{matrix} M & = & L & \times & F \\ \text{力矩} & & \text{力臂} & & \text{力} \end{matrix}$$

记住了么...

天平 公式

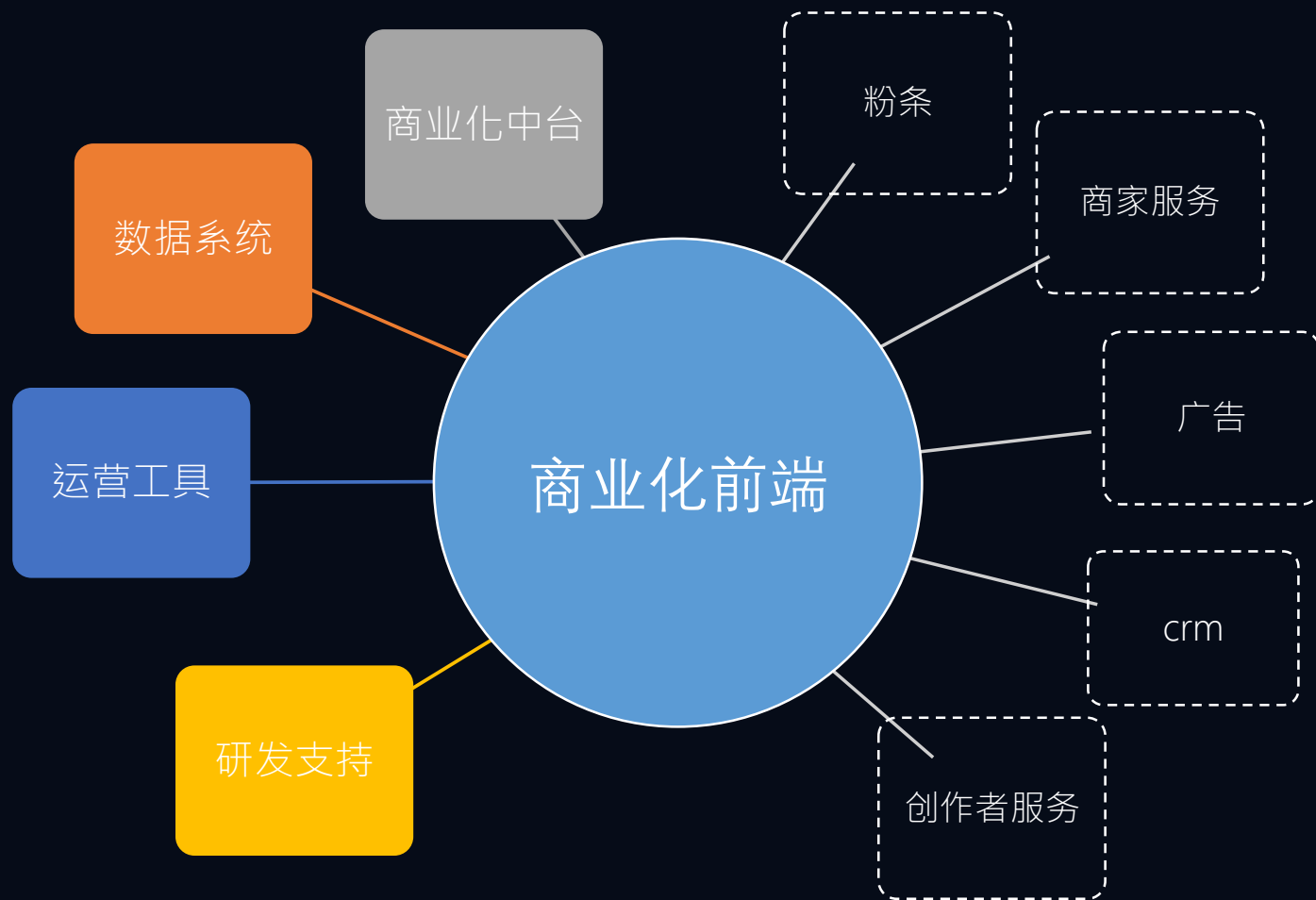
发展阶段1：团队初建



突然有了一个20人的前端团队



挑战：支撑力不足，疲于业务

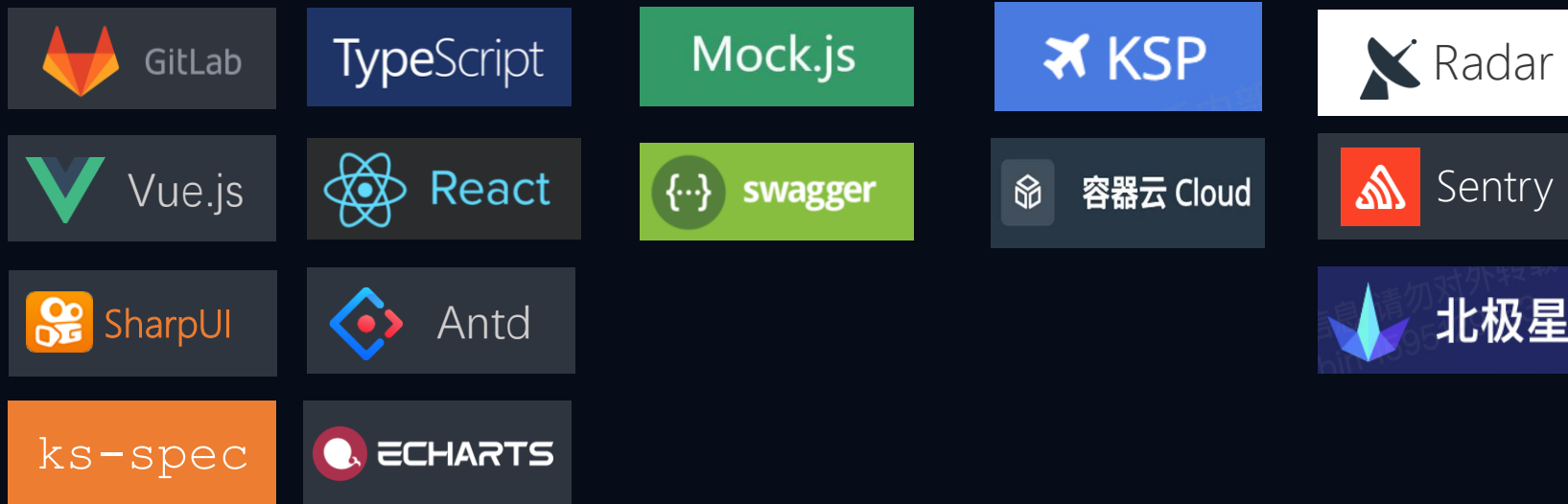


- 大团队要承担更多业务
- 短时人力无法快速补充

? 如何解决

命令行工具giao

项目初始化 → 开发 → 联调 → 部署/发布 → 监控



giao

giao project

giao swet

giao jinx

giao ...

giao project



giao project init

本地giao-project版本: 1.0.22

服务器最新版本: 1.0.22

? 请选择项目类型 (vue react kmp) kmp

下载路径: /Users/lanxiaobin/git/kmp-template

✓ 下载KMP模板

? 请输入项目名称 (package.json name字段) test

? 是否自动安装依赖 是

? CDN路径 //static.yximgs.com/static/test/

? router baseUrl (项目部署路径, 默认根路径"/") /

✓ 安装依赖

✓ 初始化Git

✓ 初始化Bit

模板安装完成, 运行命令

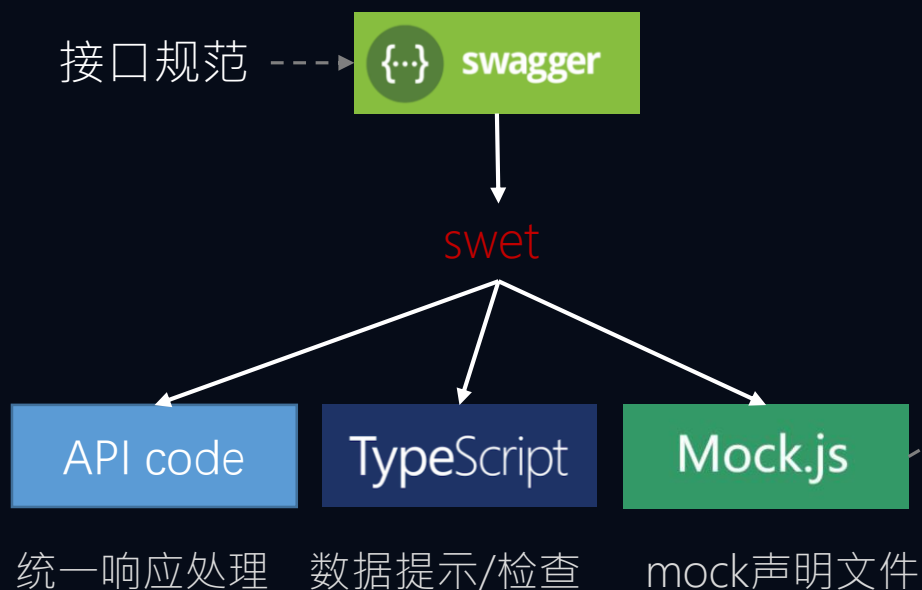
```
cd kmp-template
```

```
yarn serve
```

开始开发

giao swet

开发阶段自动化



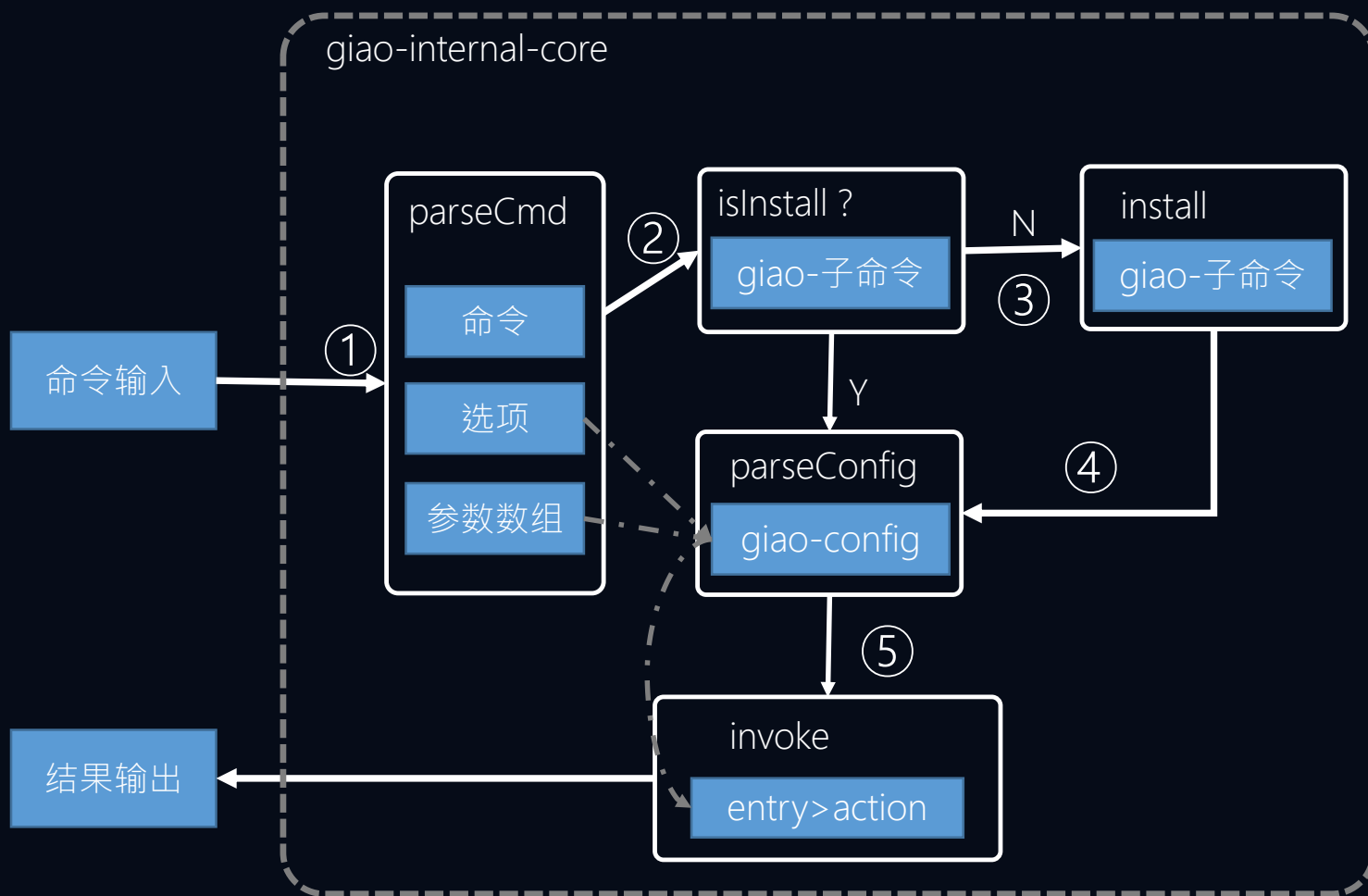
调试阶段自动化



一个swagger doc

- 前后端接口规范统一
- 类型提示和变更检查
- 自动处理数据异常
- 自动生成mock数据

giao 插件化



不同业务线多样性需求

- 插件化

giao everything !

- 快速开发新插件
- 快速整合已有插件

我们做了什么？

- 规范统一化
- 重复流程自动化
- 步骤简单化
- 减少非必要消耗

giao命令行工具

giao插件化

project, swet, jinx

standard, ssh等工具

业务模板

kast等

浏览器扩展

模拟登录

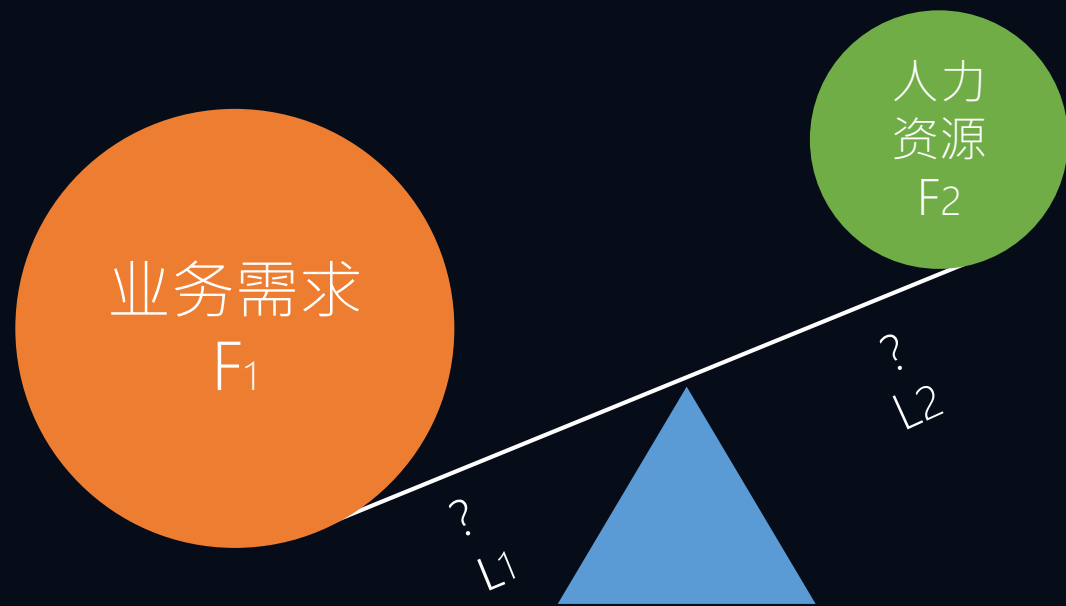
weblogger

等

小结

M1: 团队产出

M2: 团队投入

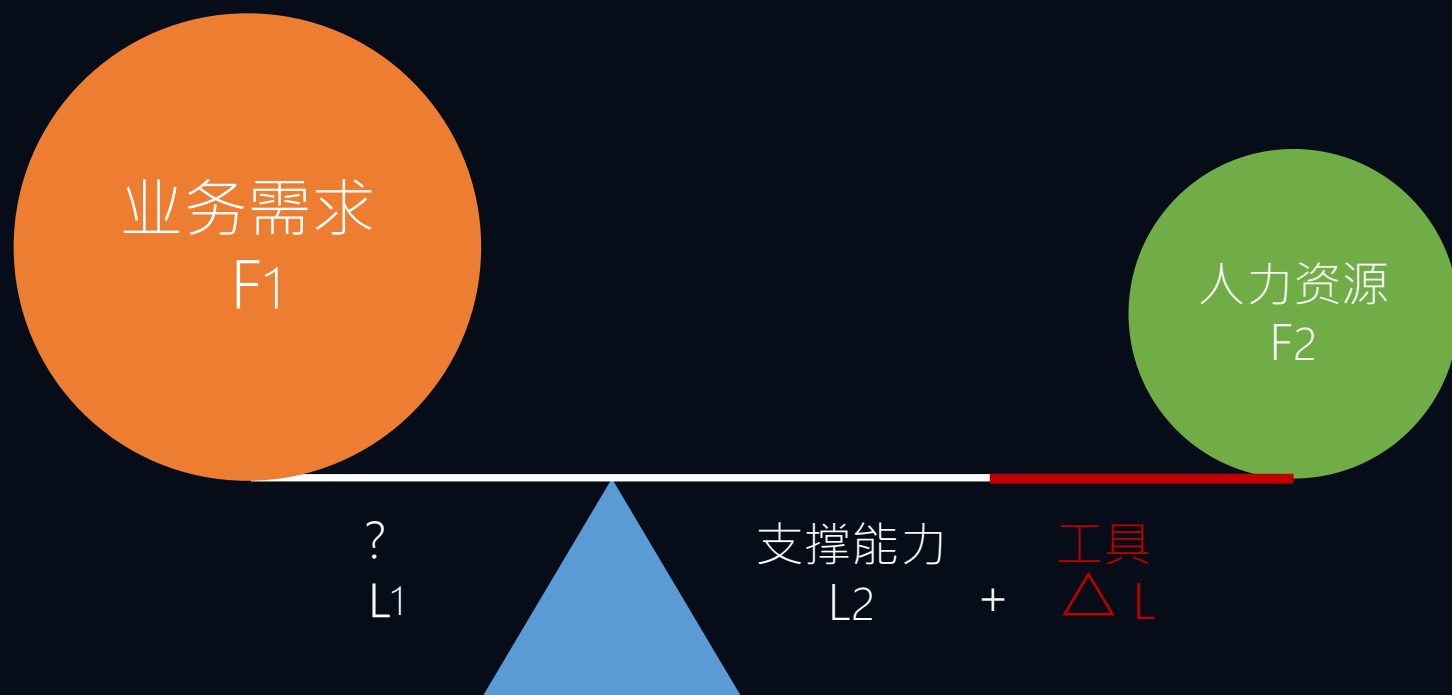


$$M(\text{力矩}) = L(\text{力臂}) \times F(\text{力})$$

- 初期，需求多，业务产出压力大
- 资源有限

疲于业务 -> 累

小结：业务支撑能力

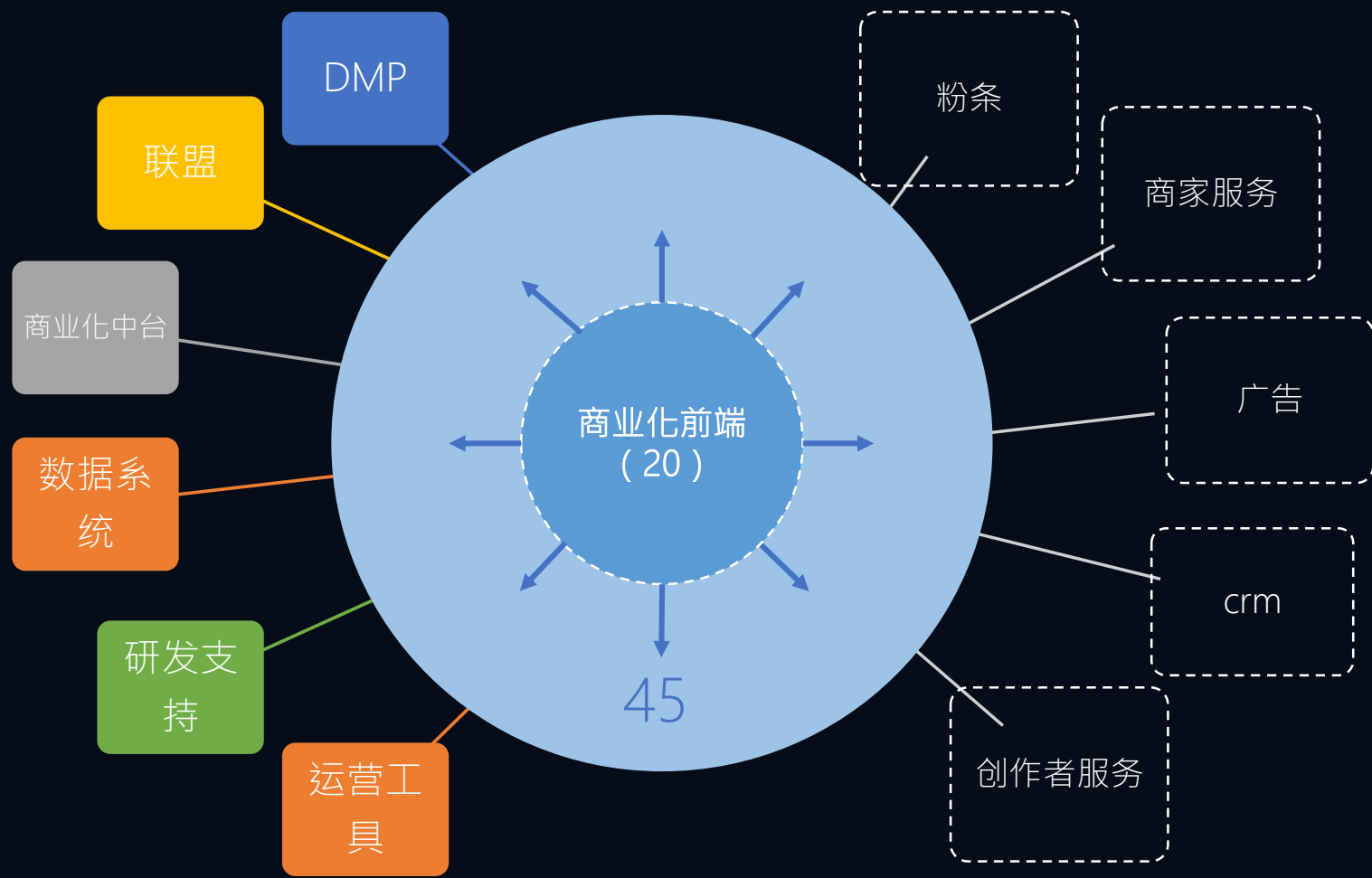


固定资源下业务支撑力变强

$$M_2 \uparrow = F_2 \times L_2 \uparrow$$

广义的投入 = 人力资源 × 业务支撑能力

发展阶段2：业务扩张，团队扩张

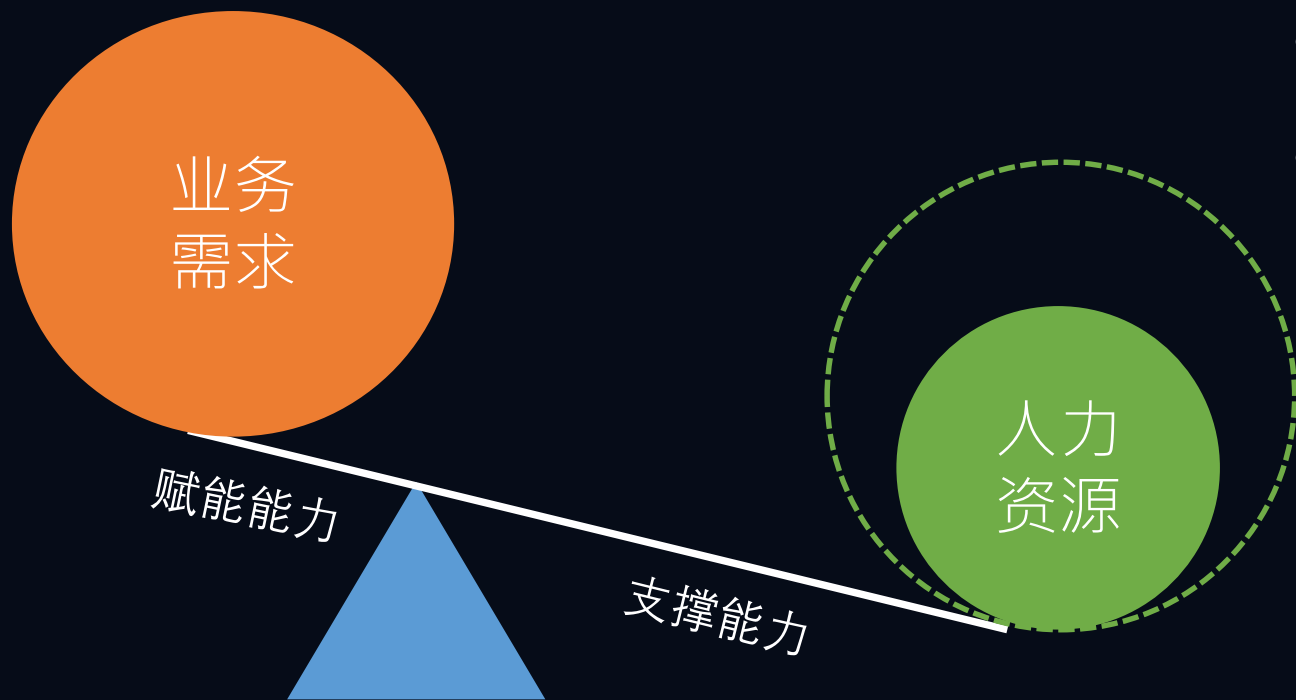


- 有一定的基建基础
- 多了一些新的业务方向
- 原有的业务变的更复杂
- 增加了一些通用中台

问题：

- 业务类型相似，有一定的基建，业务完成较为轻松
- 长期产生枯燥感
- 业务参与度不高，小资源团队 -> 大资源团队

广义上的产出并没有跟上投入
成就感降低



从需求外找输出，给业务赋能

- 广义的团队投入

- 人力 x 业务支撑能力

- 广义的团队产出：

- 需求 x 帮助需求做得更快，更好，

更稳定的一系列手段

(赋能业务的能力)



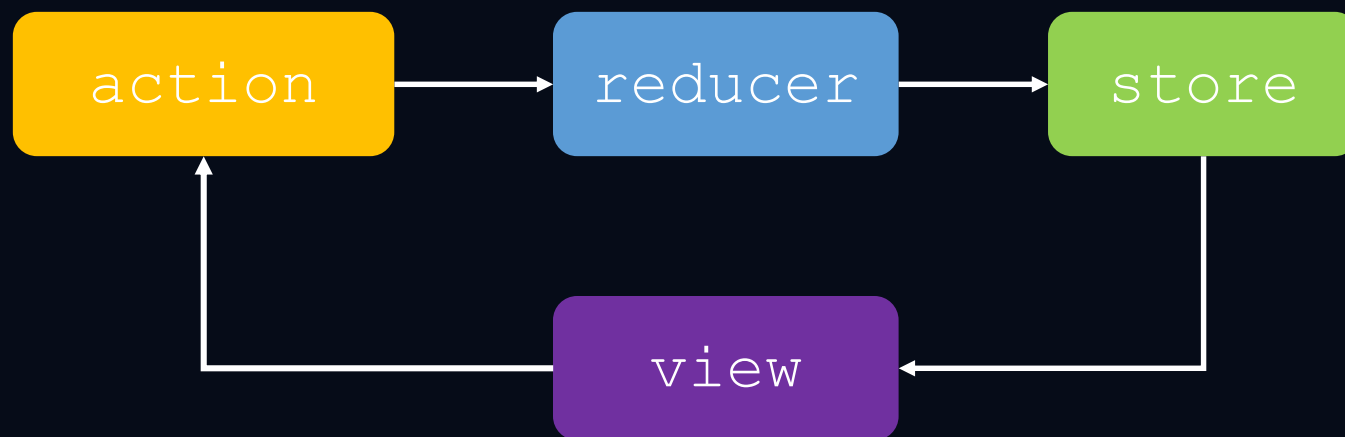
dobux：

- 在同质化业务中找问题

广告卡片：

- 打破常规，实现业务的新思路

自研dobux：redux为什么不香了？



redux的模板代码

// 1. Actions

```
const INCREASE = 'INCREASE';  
const DECREASE = 'DECREASE';
```

Action type

```
function increase() {  
  return {  
    type: INCREASE  
  };  
}
```

Actions

```
function decrease() {  
  return {  
    type: DECREASE  
  };  
}
```

```
function increaseAsync() {  
  return async (dispatch) => {  
    await delay(2000);  
    dispatch({ type: INCREASE });  
  };  
}
```

// 2. Reducers

```
function counter(state = { count: 0 }, action) {  
  switch (action.type) {  
    case INCREASE:  
      return {  
        count: state.count + 1  
      };  
    case DECREASE:  
      return {  
        count: state.count - 1  
      };  
    default:  
      return state;  
  }  
}
```

reducers

store

// 3. Store

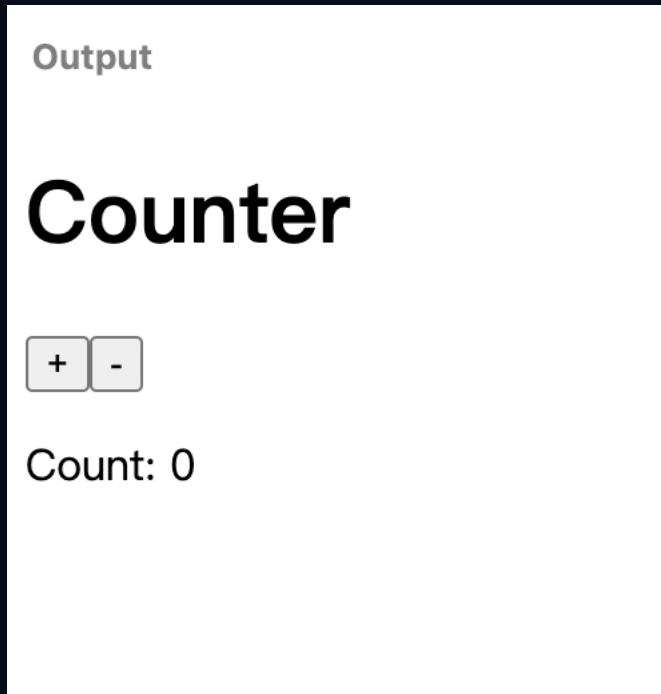
```
const store = createStore(counter, applyMiddleware(thunk));
```

// 4. View

```
const Counter = connect<any, any>(state => state)((props) => {  
  return (  
    <div>  
      <h2>{props.count}</h2>  
      <button key="increase" onClick={() => { props.dispatch(increase)}>  
      <button key="decrease" onClick={() => { props.dispatch(decrease)}>  
      <button key="increaseAsync" onClick={() => { props.dispatch(increaseAsync)}>
```

dispatch

redux带来的问题



```
✓ campaign
TS action.ts
TS action-types.ts
TS reducer.ts
TS reducers.ts
TS store.ts
```

- dispatch不支持ts类型提示
- reducer的state需要手动
immutable

自研状态管理工具dobux



Dobux

轻量级的响应式状态管理方案

快速开始 →

简单易用

仅有 3 个核心 API，无需额外的学习成本，只需要了解 React Hooks

不可变数据源

通过简单地修改数据与视图进行交互，生成不可变数据源，保证依赖的正确性

TypeScript 支持

提供完整的 TypeScript 类型定义，在编辑器中能获得完整的类型检查和类型推断

轻量级

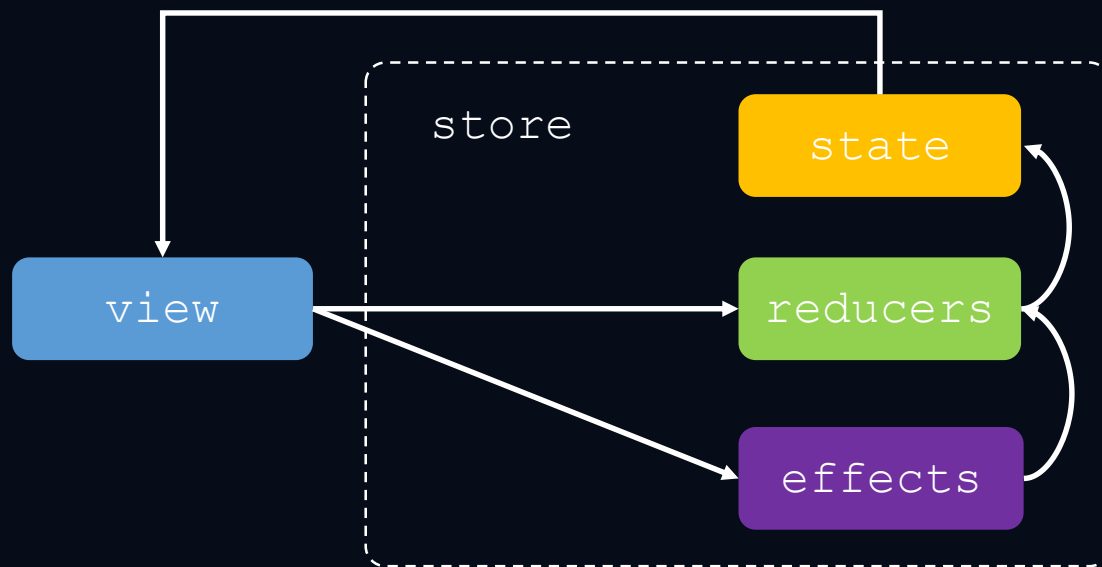
渐进式

状态管理方案

dobux – 状态操作集中管理，数据流向清晰

```
// 1. Store
const store = new Store({
  counter: {
    state: {
      count: 0,
    },
    reducers: {
      increase(state) {
        state.count++
      },
      decrease(count) {
        state.count--
      },
    },
    effects: (store, rootStore) => ({
      async increaseAsync() {
        await wait(2000)
        store.reducers.increase()
      },
    }),
  },
})
```

状态操作集中管理



```
const Counter = () => {
  const { state, reducers, effects } = store.useStore('counter')
```

dobux – 自动的状态不可变包装

```
private produceState(state: C['state'], reducer: ConfigReducer, payload: any = []): C['state'] {  
  let newState  
  
  if (typeof state === 'object') {  
    newState = produce(state, draft => {  
      reducer(draft, ...payload)  
    })  
  } else {  
    newState = reducer(state, ...payload)  
  }  
  
  return newState  
}
```

```
reducers: {  
  increase(state) {  
    state.count++  
  },  
  decrease(count) {  
    state.count--  
  },  
},
```



dobux - 友好的ts类型推断

```
5  const { useStore } = store
6
7  const Count: FC = () => {
8    |   You, a few seconds ago • Uncommitted changes
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```


dobux的成果

- 解决模板代码问题
- 完美支持dispatch的类型提示
- 自动处理reducer的state不可变性



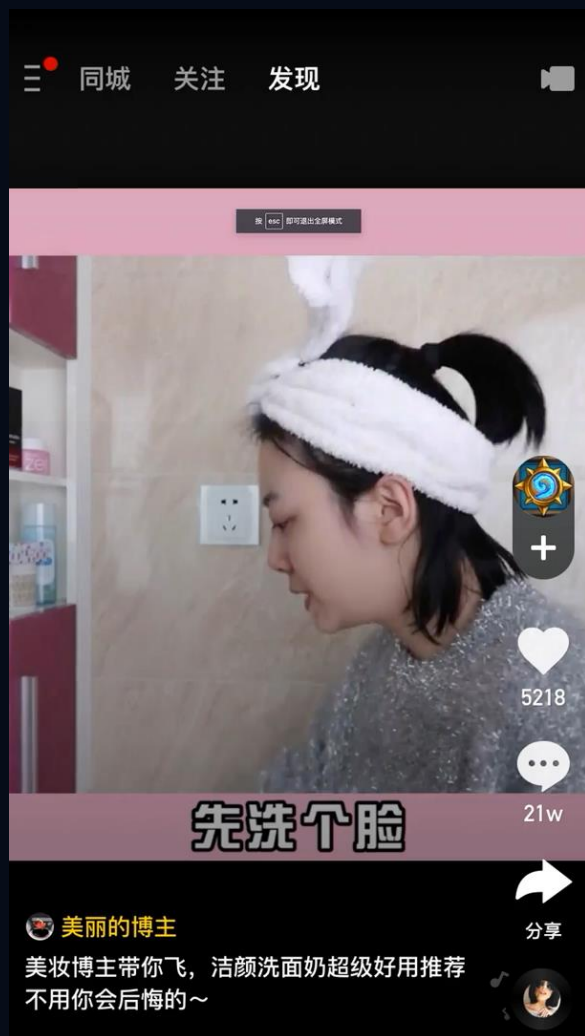
- 3 个核心 API，简单易用
- 非单一store，更新精准
- 支持time-travel



解决了项目的维护问题

- 基建中台业务全面引入
- 覆盖10+个产品
- 开发体验上升
- 新人接手快，代码稳定性提高

案例2：广告卡片



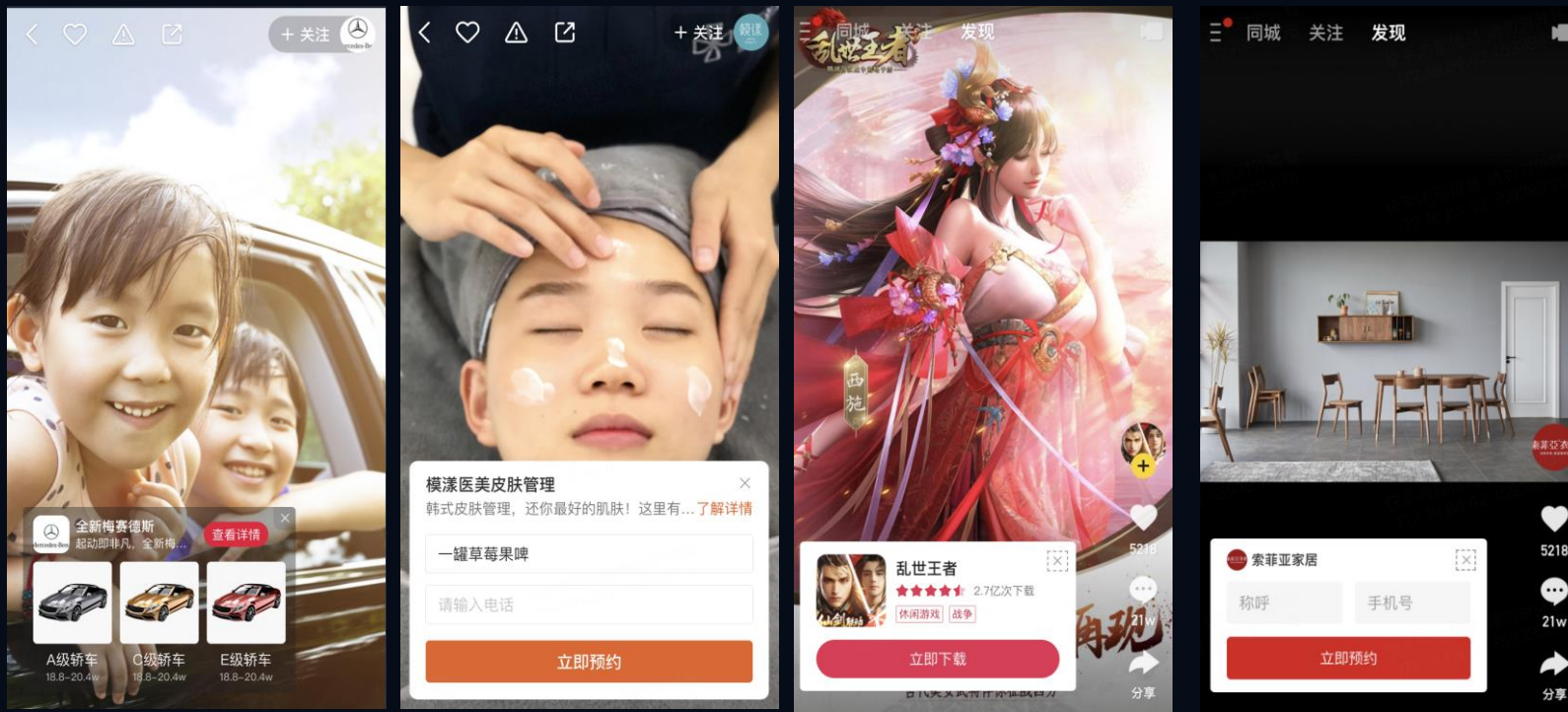
app原生开发 + 配置项

案例2：广告卡片

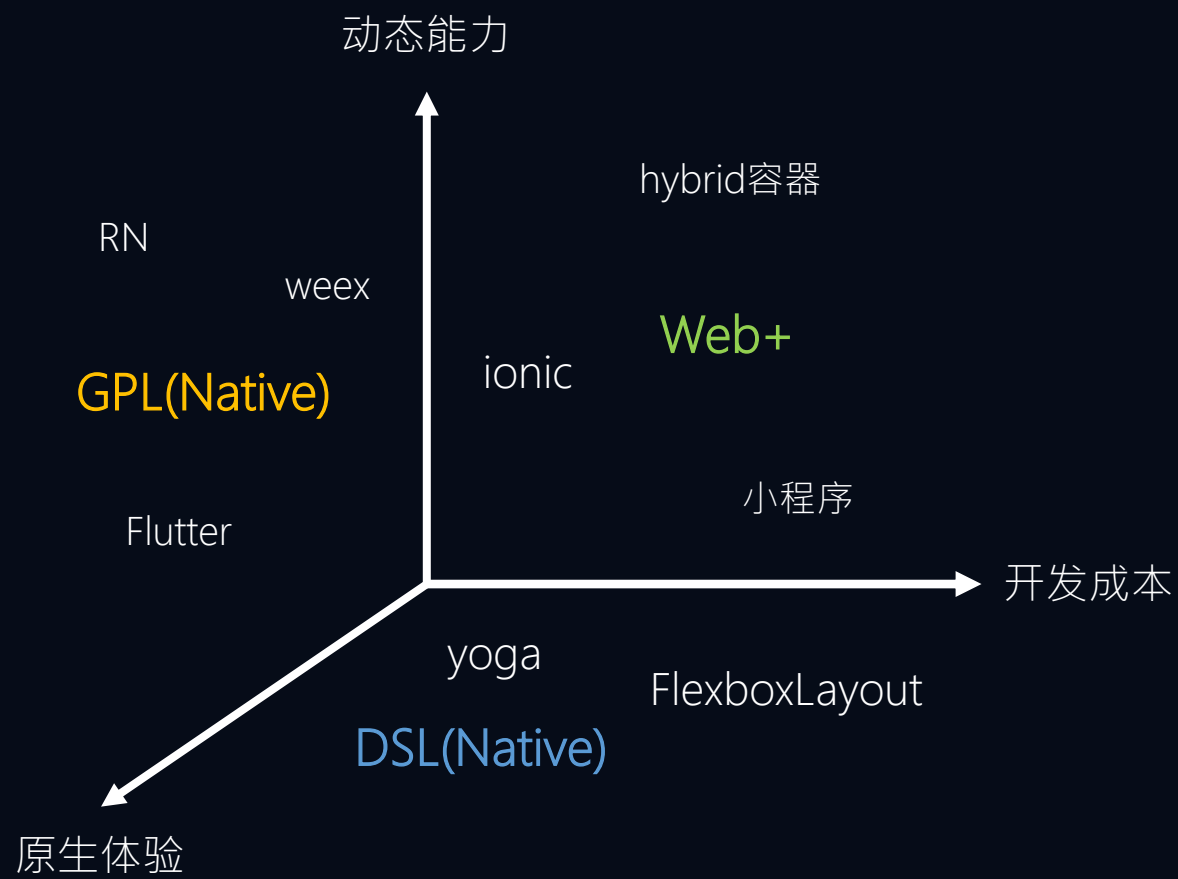
样式变多，表单交互需求

- 灵活度
- 包大小
- 开发周期

快速实验 > 广告卡片动态化



动态方案差异

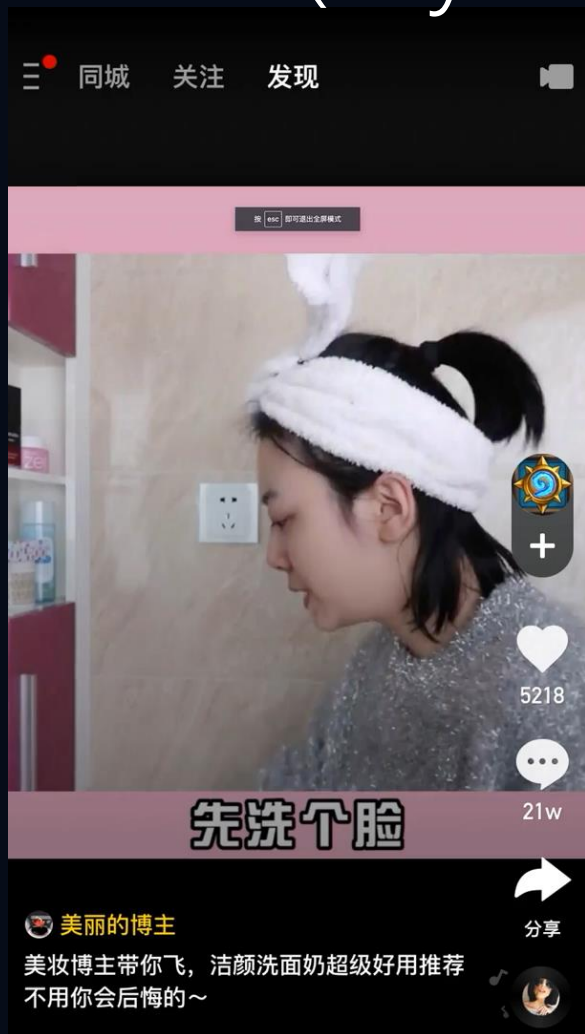


	原生体验 (性能/能力)	动态能力	开发成本	包体积
GPL(Native)	高	高	高	高
Web+(hybrid)	低	高	低	低
DSL(Native)	高	中	低	低

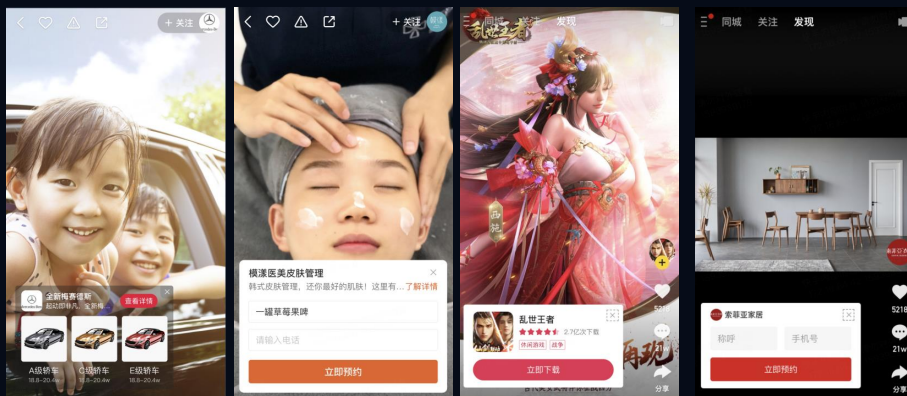
综合业务形态来看，快速落地实验选择了web+方案

<https://zhuanlan.zhihu.com/p/64968076> (@于天航)

web+ (hybrid方案) vs DSL



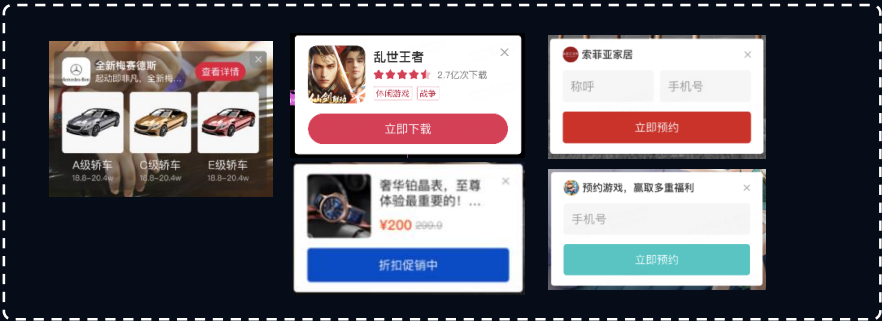
综合业务场景来看，web+能够最快达到目标



前端动态方案成果

2019.12 → 2020.6

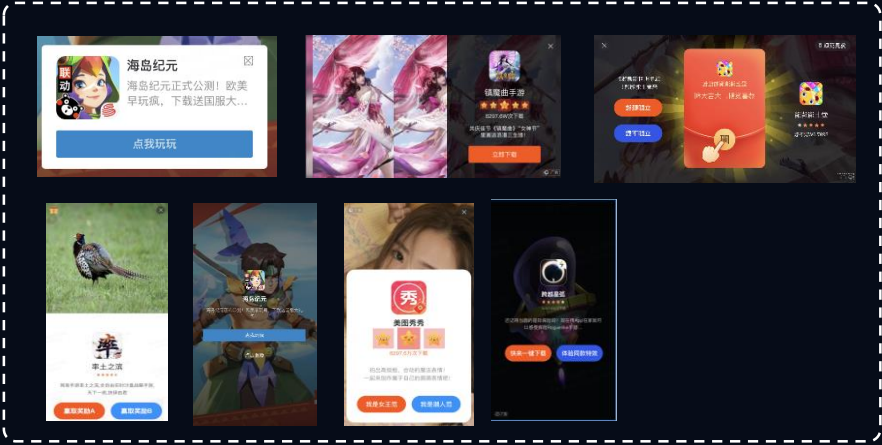
信息流



快享



联盟



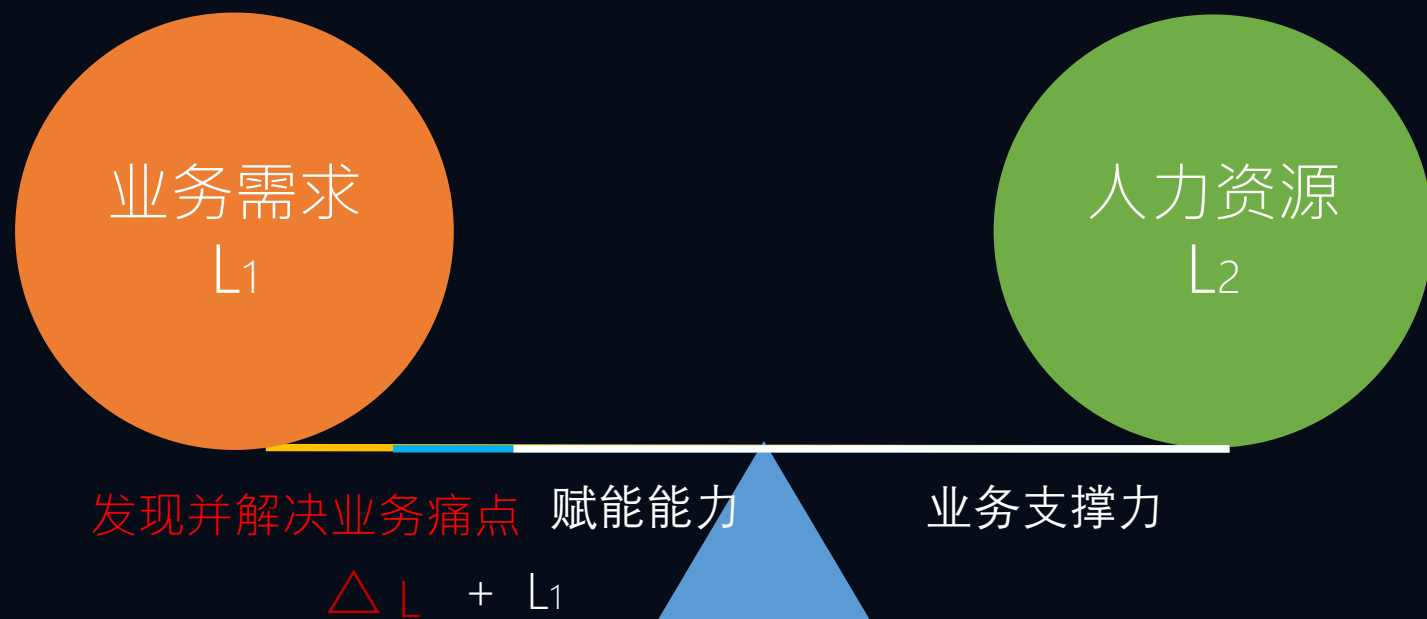
3 业务线 10+ 卡片 100+ 迭代 N 实验



性能问题 (轮播 , 更复杂的动画等)
模板url形式的劫持问题
多端联调问题

更高性能的动态方案
(DSL下发 , 原生渲染)

小结：业务赋能能力



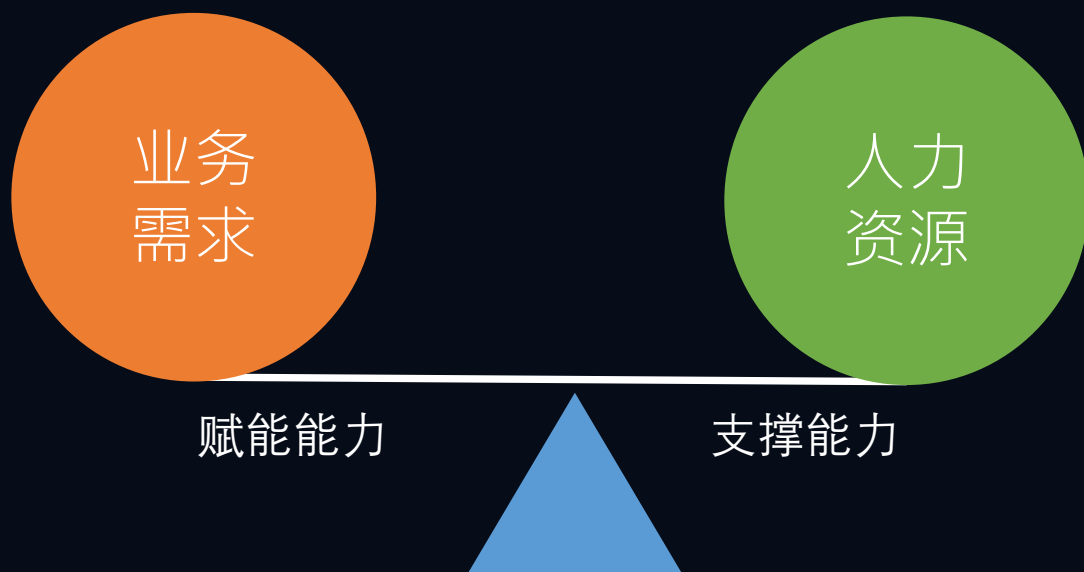
同质化业务中找到更多的赋能点

$$M_1 \uparrow = F_1 \times L_1 \uparrow$$

广义的产出 = 业务需求 \times 业务赋能能力

第三个阶段：跳出业务，规划未来

团队建设：抵抗变化的能力



- 微前端架构应用
- 低代码中后台开发平台

微前端在商业化的主要场景



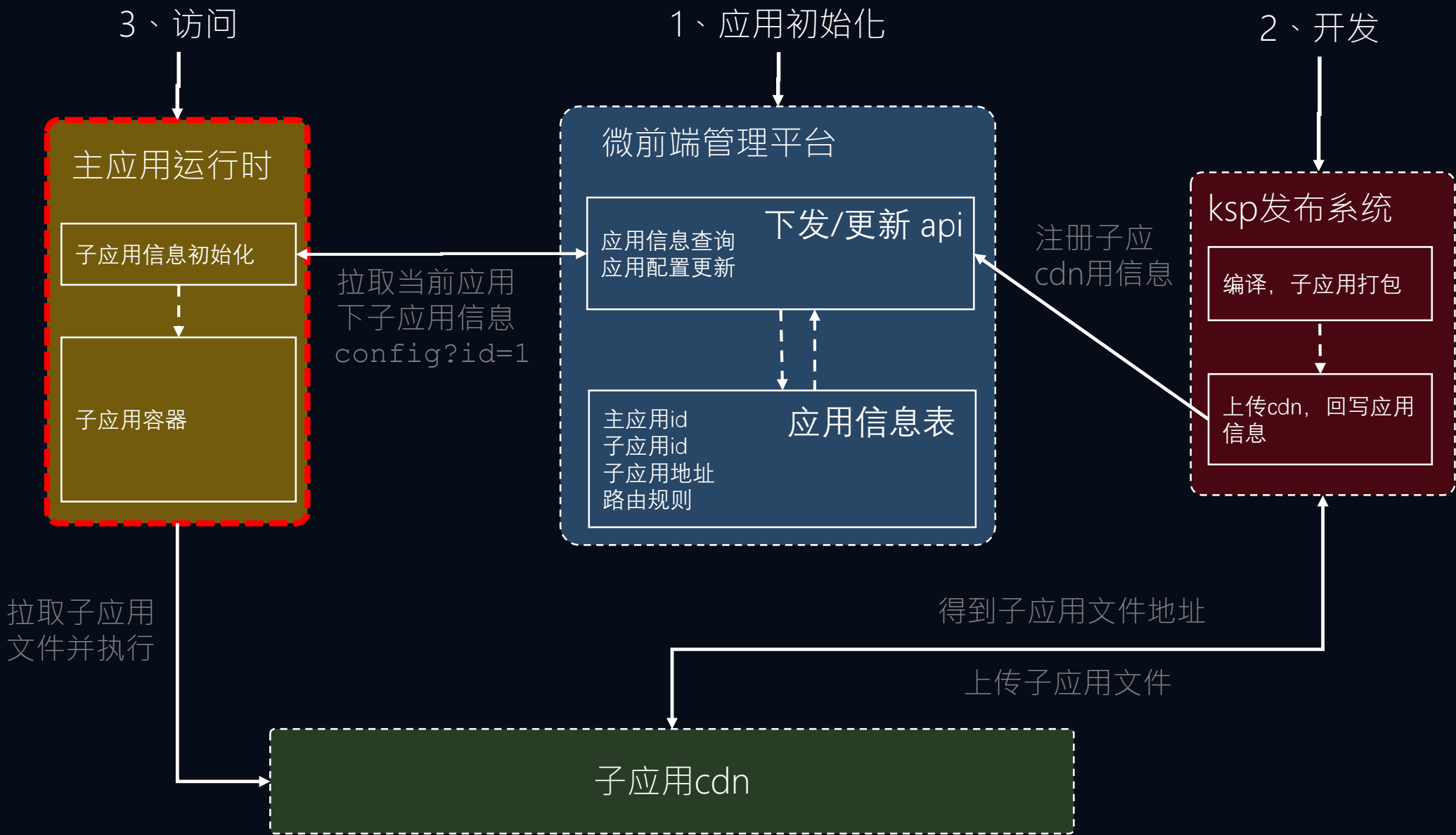
- 不同时期项目
- 不同业务线团队维护

技术栈无关
接入方便
应用容器

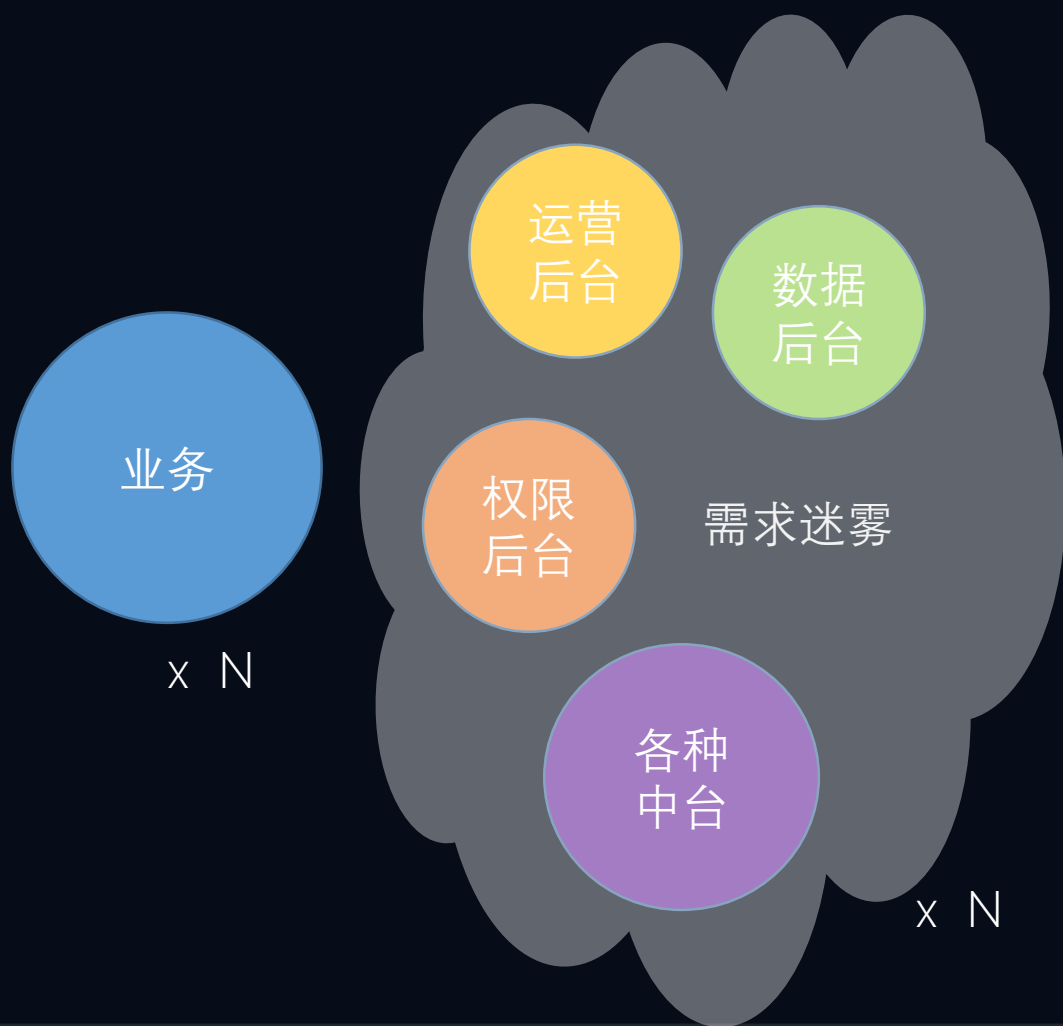
} 微前端

微前端运行时-dolly





中后台需求爆发：低代码开发平台

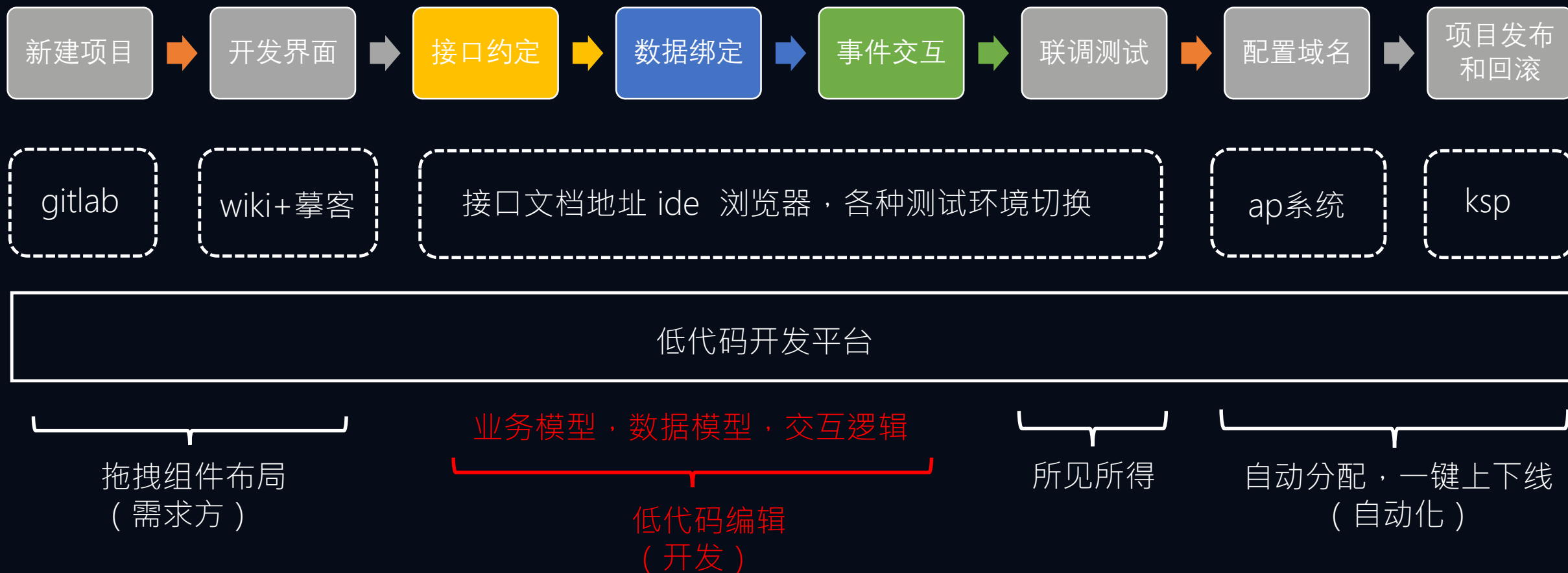


迷雾需求

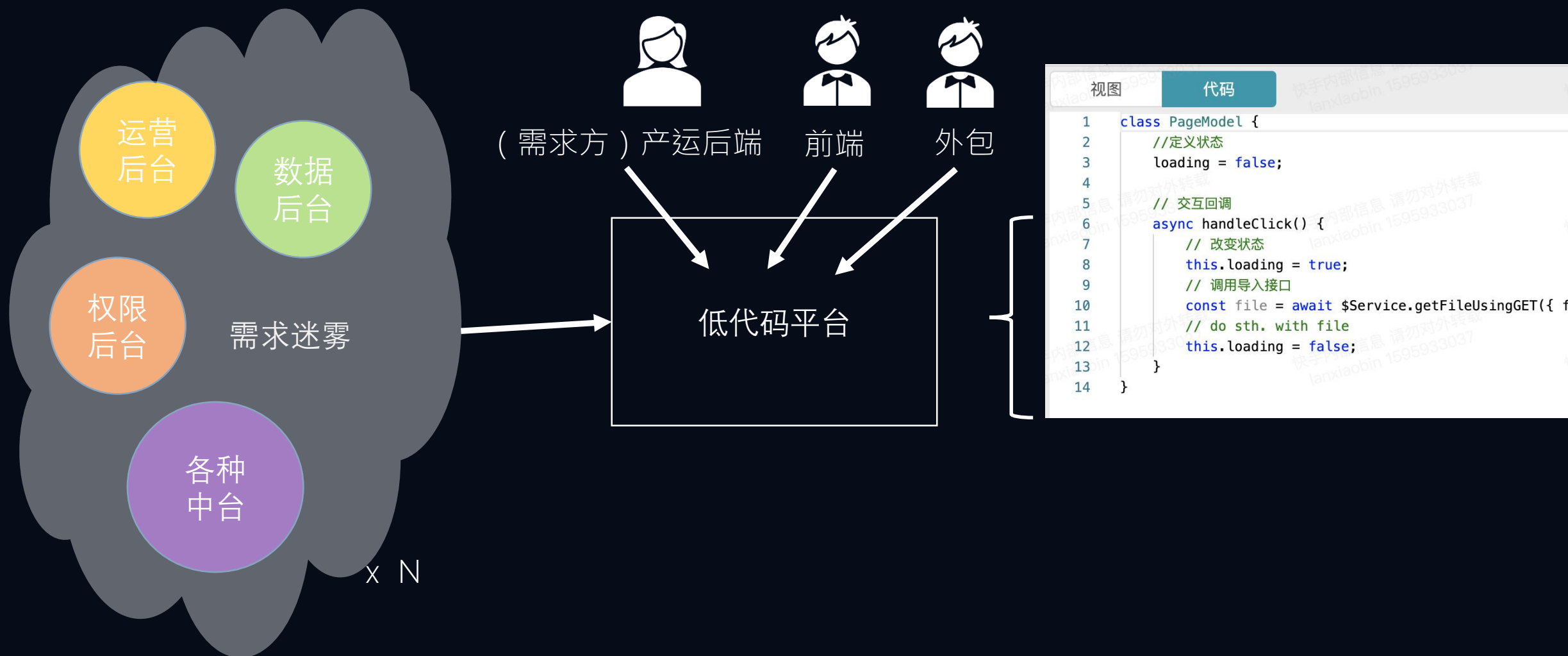
- 需求大体一致，但业务领域模型各异（很难0代码解决）
- 完成后迭代频率低，重拾成本高（一次性需求）
- 平台分散，缺少统一管理，到处存在被遗忘的平台（难以管理）

对业务发展有益，不得不做 -> 低代码平台

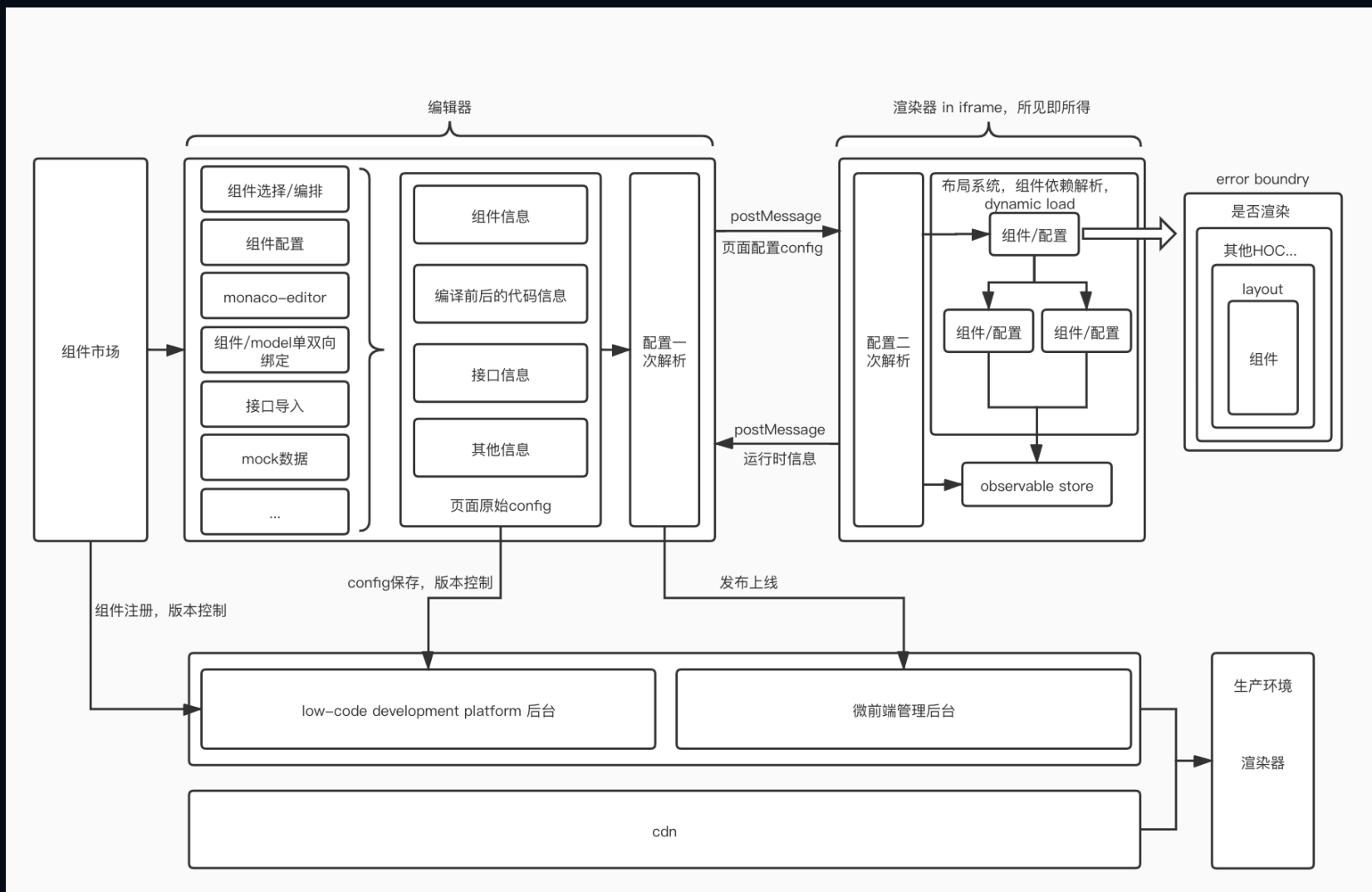
低代码平台的优势 - 一站式开发



低代码平台的优势 – 降低研发门槛，转移生产力



低代码平台主要实现



编辑器 (布局 , 组件管理 , 代码编辑器)

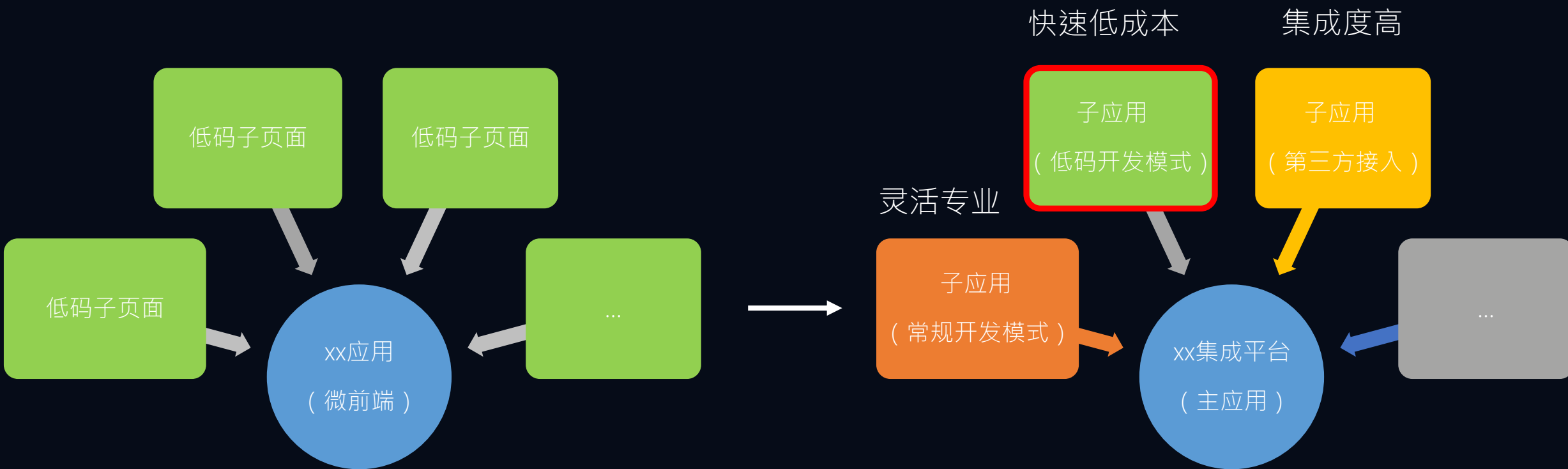
实时渲染器

绑定解释器

observable store

运行时渲染引擎等

低代码平台与微前端



- 低代码产物: 子页面 (子应用)
- 微前端: 项目容器, 路由, 版本控制

品

组件

数据

版本

新增接口

导入接口

alarm-controller(Alarm Controller)

createUsingPOST

/api/dqc/alar

create

dubhe-report-sla-controller

(Dubhe Report SLA Controller)

get

getReportInfoUsingGET

/api/da/report/report

离线数据质量|巡检页|获取数据质量信息

get

getReportReadyInfoUsingGET

/api/da/report/report_summary

离线数据质量|巡检页|获取数据质量信息(总览信息)

dubhe-table-execute-info-controller(Dubhe Table Execute Info Controller)

get

getBusinessChainUsingGET

/api/da/table/business

离线数据质量|报表信息页|获取业务和子业务联动信息

get

getTableExecuteReadyInfoUsingGET

/api/da/table/report_summary

离线数据质量|报表信息页|获取数据质量信息(总览信息)

get

getTableExecuteInfoUsingGET

/api/da/table/tables

离线数据质量|报表信息页|获取过滤条件下的table执行信息

接口管理

视图

代码

日期

2020-07-04

业务

请选择

子业务

请选择

数据总体概况

P0: 0

未配置报警信息个数: 0

完成: 0

按时完成: 0

未完成: 0

P1: 0

未配置报警信息个数: 0

完成: 0

按时完成: 0

未完成: 0

P2: 0

未配置报警信息个数: 0

完成: 0

按时完成: 0

未完成: 0

就绪监控报表

表运行例行时间

数据表名称

数据表类型

业务

子业务

No Data

保存发布

容器_86 属性配置

布局

AD DATA ADMIN

监控 / 就绪监控

日期

2020-07-04

业务

请选择

子业务

请选择

查询

数据总体概况

P0: 8

未配置报警信息个数: 17

完成: 0

按时完成: 0

未完成: 0

P1: 64

未配置报警信息个数: 10

完成: 11

按时完成: 1

未完成: 17

P2: 12

未配置报警信息个数: 0

完成: 1

按时完成: 0

未完成: 0

其他: 1

未配置报警信息个数: 0

完成: 1

按时完成: 0

未完成: 0

就绪监控报表

表运行例行时间

数据表名称

数据表类型

业务

子业务

优先级

表运行队列优先级

数据表元信息链接

表运行报警配置

kd_ad_dw.creative_audit_stage_df

整体数据

业务中台

审核

P2

查询

查询

查询

ks_ad_dev_ad_dw_service_line_cost_base_di

【新样式】商业化概览

管理视角

商业化总览

P0

L_1

he

查询

查询

查询

ks_ad_dw.account_audit_stage_df

账户视角

业务中台

审核

P2

L_0

chen

查询

查询

查询

ks_ad_dw.ad_account_hit_line_monit

创建任务

数据工具

数据超市

P2

L_0

lin

查询

查询

查询

定位

上

0px

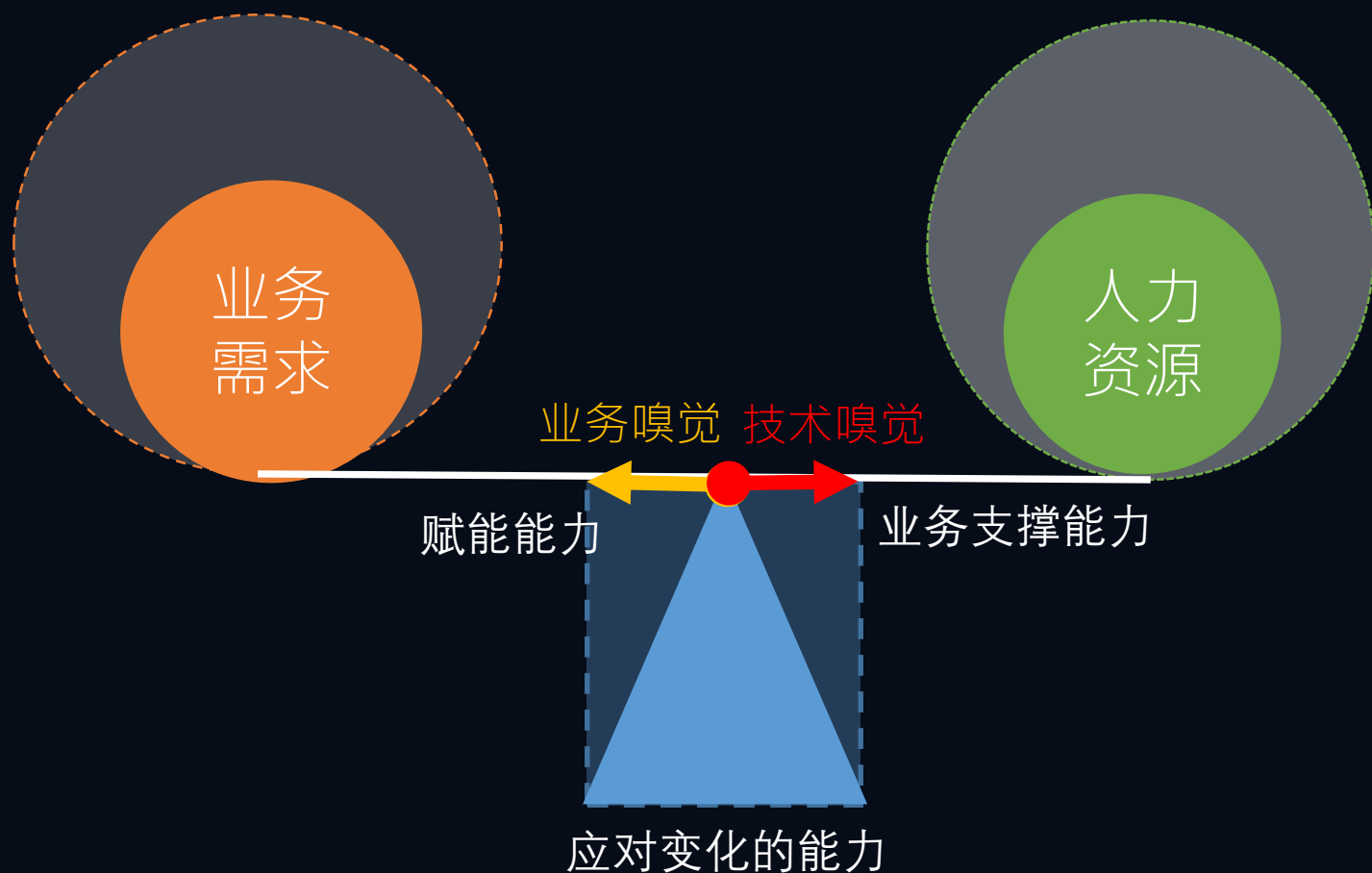
左

0px

右

0px

小结：应对变化能力



- 业务嗅觉
多了解业务，保持业务走向的判断
- 技术嗅觉
多了解新技术和业内流行的技术方案，思考是否在业务中能够有所应用

支点 -> 面 稳定性

回顾一下...

3种能力

- 业务支撑能力
- 赋能业务的能力
- 应对变化的能力

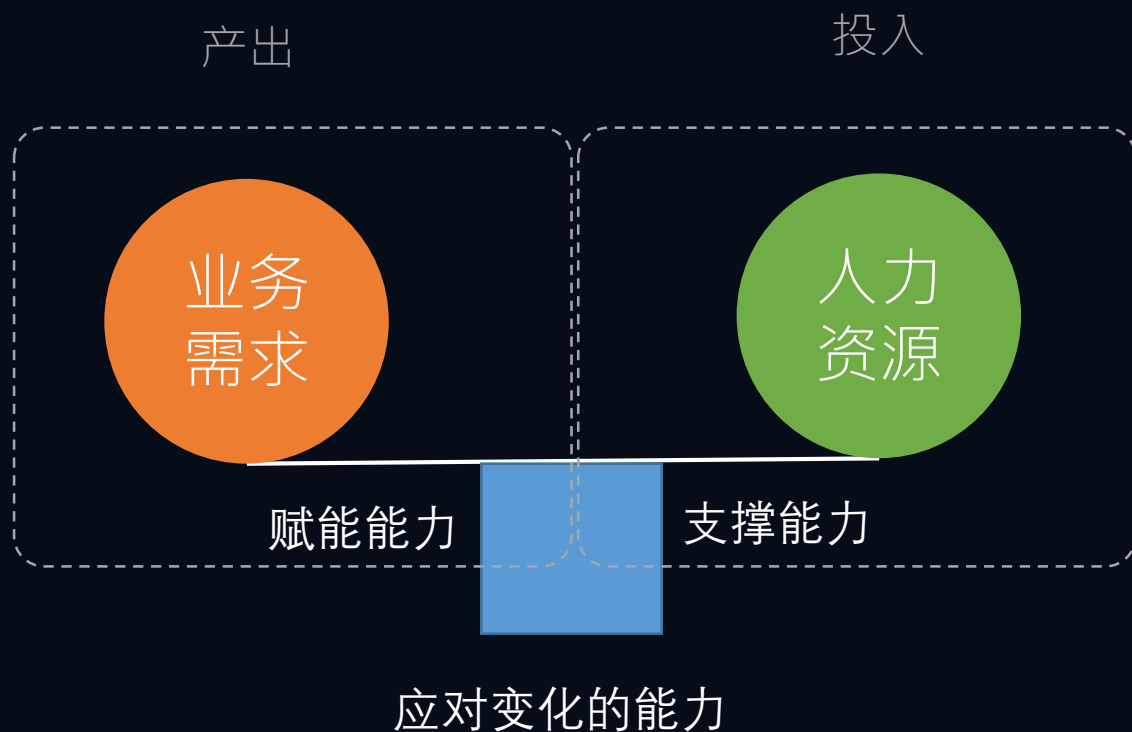
2个阶段

- 团队组建
- 业务稳定

5个案例

- giao
- dobux
- 动态卡片
- 微前端
- 低代码平台

总结：投入&产出动态平衡



团队建设的目标：稳固的天平模型

$$M(\text{力矩}) = L(\text{力臂}) \times F(\text{力})$$

快手大前端
技术交流会 2020

THANKS



——感谢可爱且认真的你们