D2C & 低代码在腾讯音乐人业务的探索实践

腾讯音乐 / TME研发部 / 朱明鹏

About me

现任腾讯音乐人平台前端组长

从业11年，专注于效能工具的设计研发

曾出版企业低码实践书籍《活动中台》

大纲

1、优化前端开发模式&流程

2、D2C 转码技术的关键实现

3、低码与D2C合力完成“端到端”

4、总结与展望

1. 优化前端开发模式&流程

1.1 现状前端模式

1.2 发现的问题

1.3 探索方案

1.4 流程链路

现状前端开发模式

工程化开发

类型：存量项目、体系的项目。

特点：逻辑复杂、复用性低、低配置性。

低代码开发

类型：B端管理系统、C端运营页

特点：组件、模版可复用、逻辑可配置、高配置性。

发现的问题

“研发还原度低，**很明显的问题**没有发现。”
“设计走查**太浪费时间**，细节问题多，按时上线都有风险。”

设计师要“效果”好

研发要“效率”高

“哪种开发模式都绕不开切图还原设计稿，**过程成就感低**。”
“一些显而易见的业务和组件，能不能**自动生成**？”
“还要用走查工具去细节比较，我想**专注更有价值的工作**。”

探索方案

Desgin to </>

流程链路

2. D2C 转码技术的关键实现

2.1 设计稿清洗

2.2 图标合并

2.3 还原空间关系

2.4 分行分列

2.5 提取循环节点

2.6 转化样式

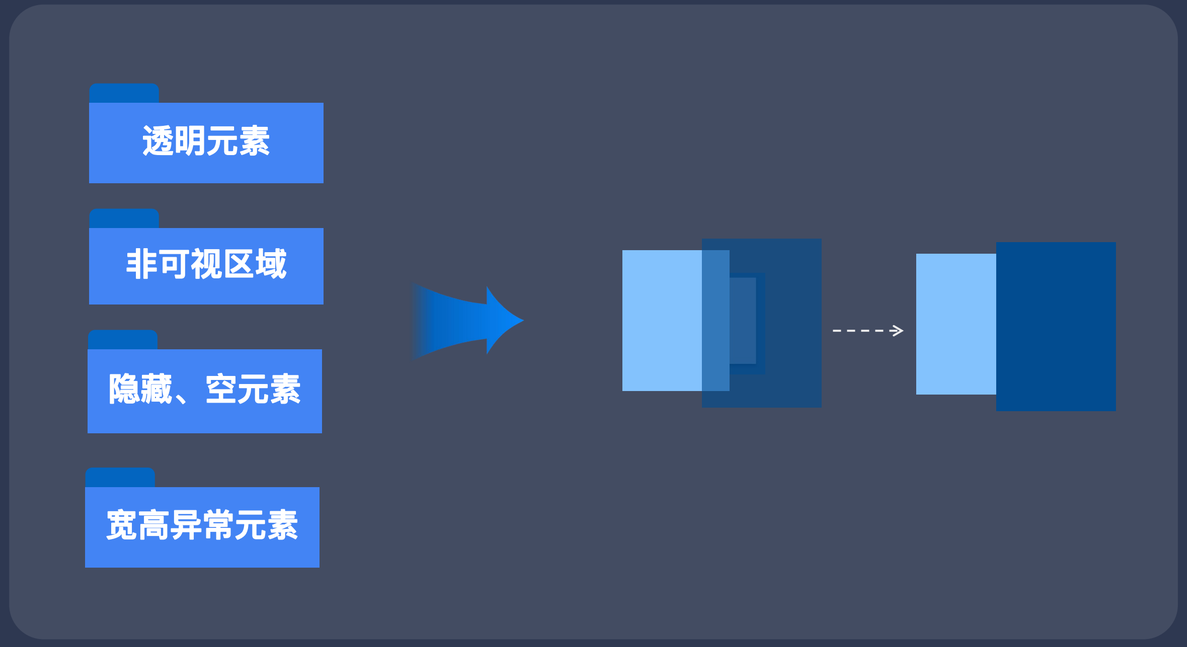
2.7 构建骨架

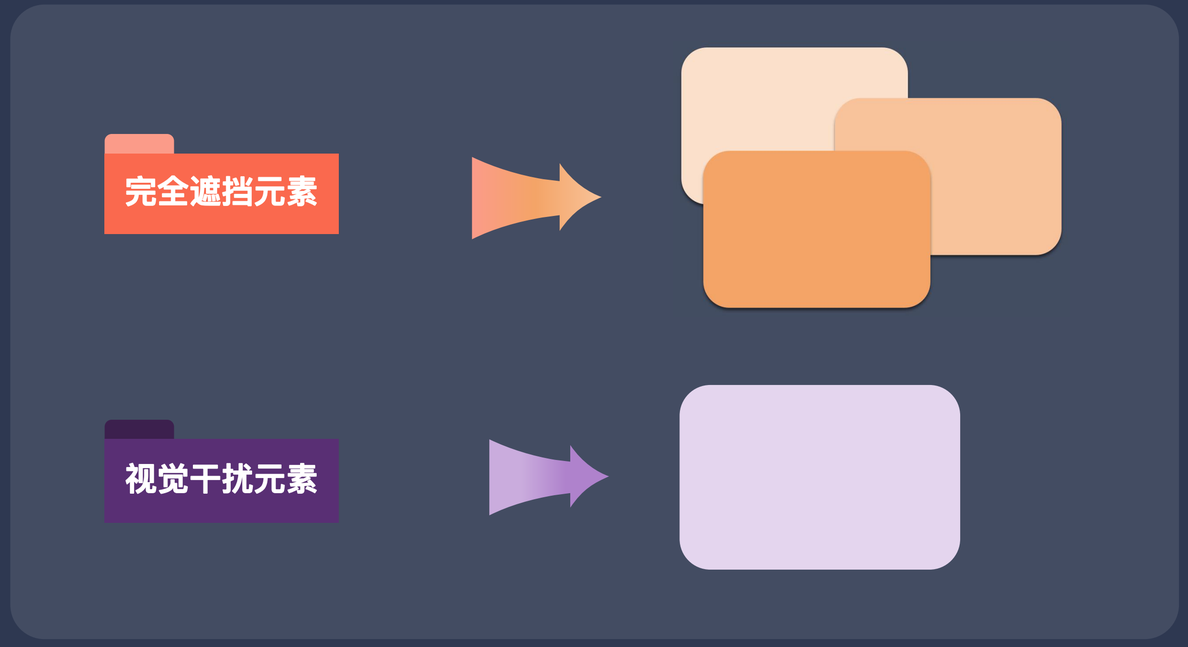
2.8 多平台生成

D2C 转码关键实现 / 1.设计稿清洗 / 去除干扰元素

1.去除干扰元素

识别无效元素，确保生成的结构不包含冗余代码。





D2C 转码关键实现 / 1.设计稿清洗 / 合并&转换节点

2.合并&转换节点

将设计稿中的图层，统一封装成标准的四种元素。

1. 原始节点合并

2. 节点增加类型

3. 标记切图

- #ICON
- #IMG

图标　图片　容器　文本

D2C 转码关键实现 / 2.图标合并

图标二次合并

多元素组成图标的情况

合并策略

尺寸阈值 && 类型 + （相切、相交、包含）

现实问题

实际合并后，层级发生不正确覆盖

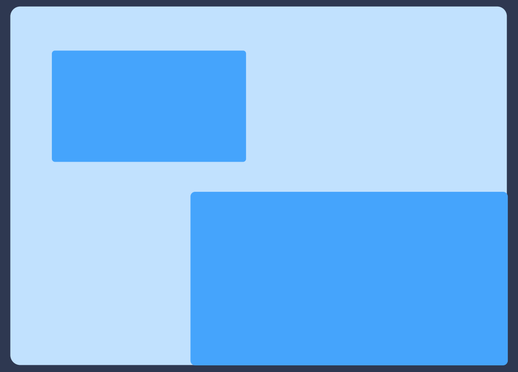
解决方案

检测图标组成元素的层级，二次裁剪后合并

新势力计划 2023 邀请
新势力计划 2023 邀请
新势力计划 2023 邀请

D2C 转码关键实现 / 3.还原空间关系

包含 – 父子关系

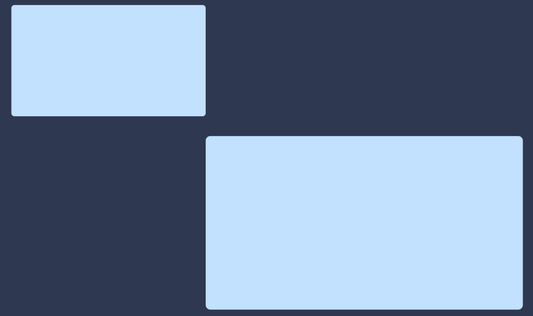


子元素需要矫正透明度、旋转、背景等属性。

相交 — 任意定位

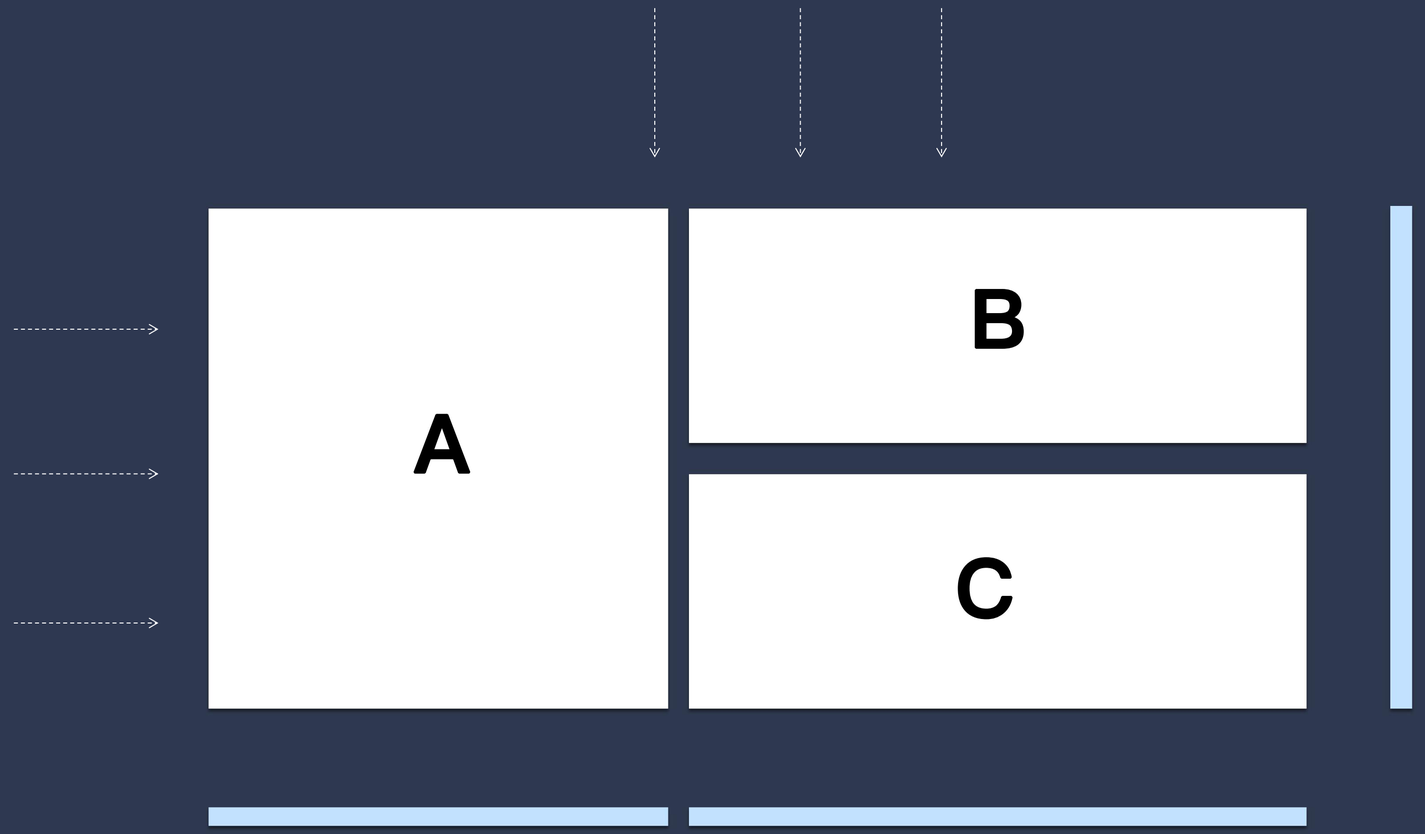
任意定位元素有相对父级

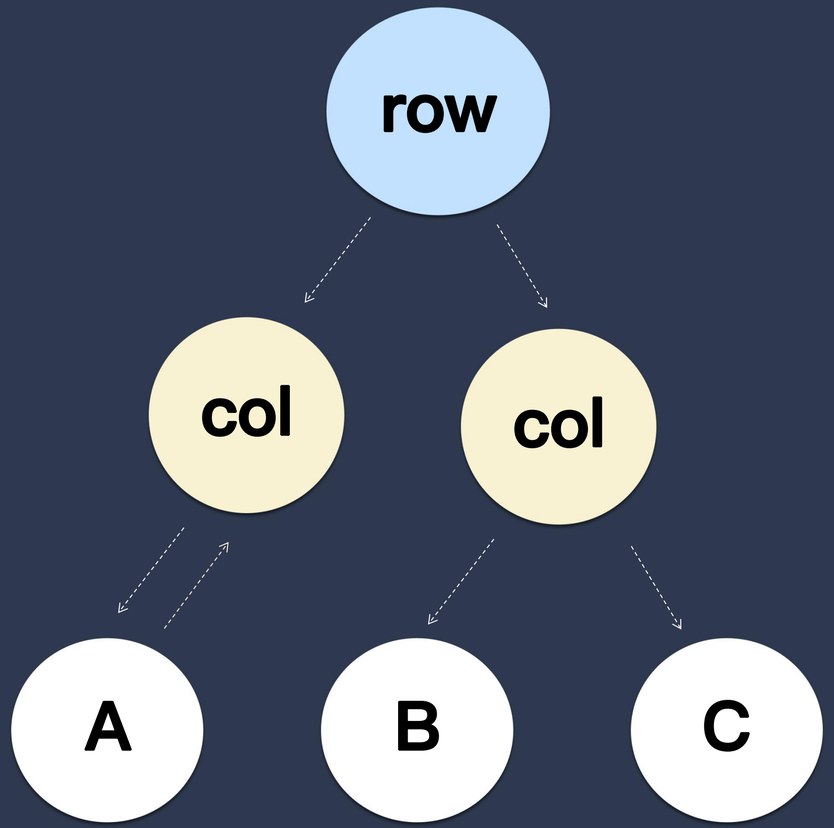
相离 — 正常定位



相离的节点，进入分行分列计算

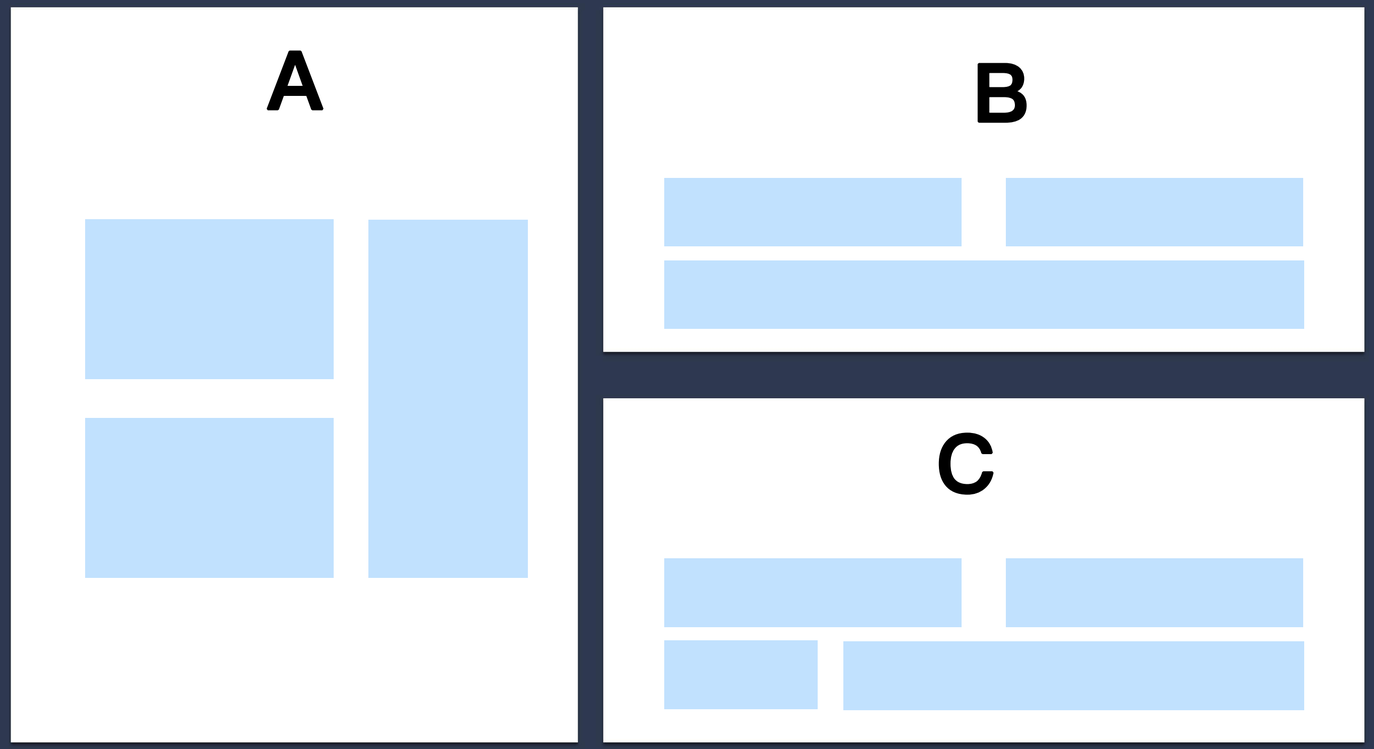
D2C 转码关键实现 / 4.分行分列





求多个连续区间在某个方向上的并集

D2C 转码关键实现 / 思考

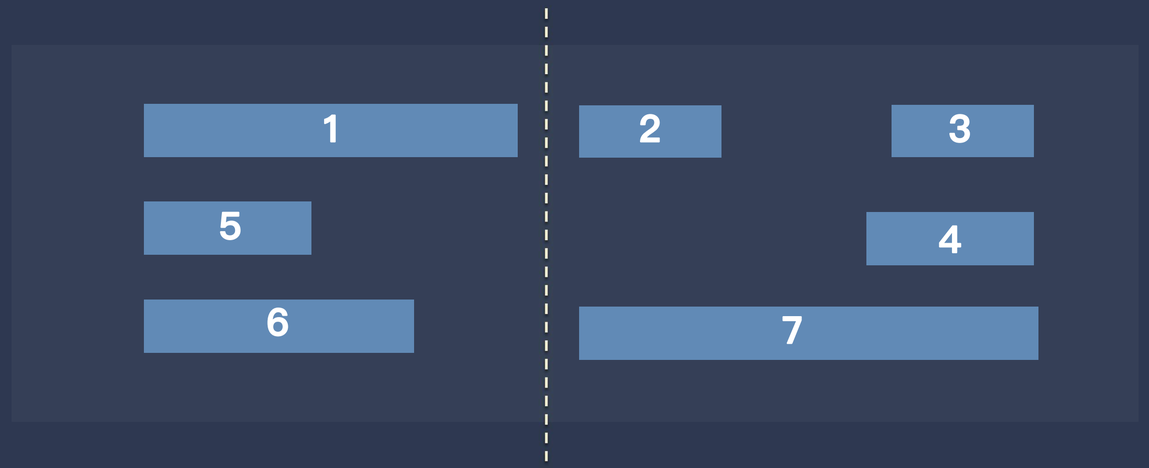


A、B、C

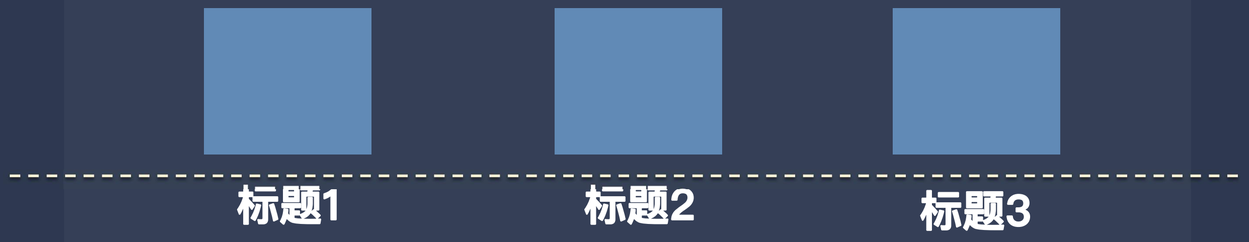
规律：依次分行分列?

D2C 转码关键实现 / 4.分行分列 / 最优布局算法

分行 ⇢ 分列？

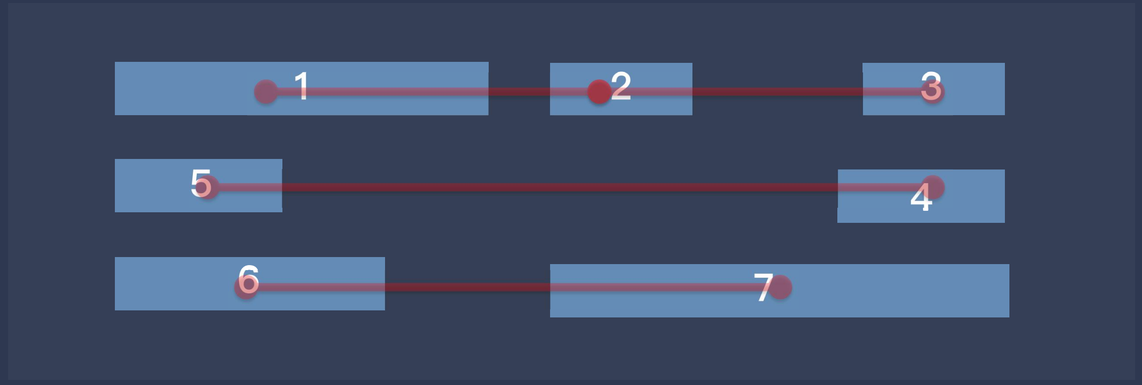


分列 ⇢ 分行？



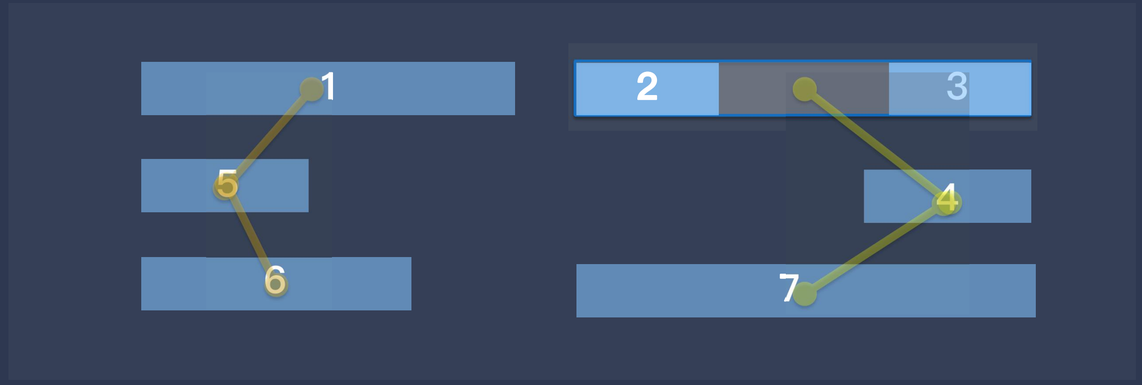
D2C 转码关键实现 / 4.分行分列 / 最优布局算法

1.分别求行列布局特征



求【行】布局特征

每一行布局特征：上、中、下

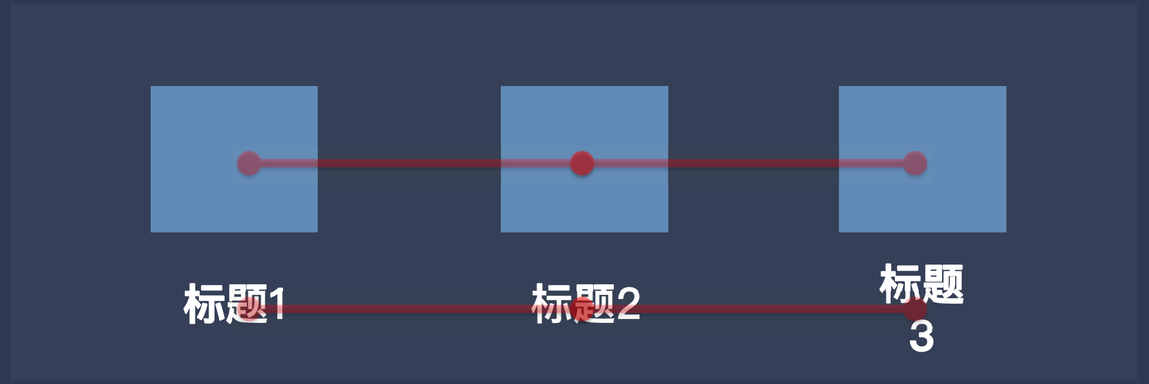


求【列】布局特征

每一列布局特征：左、中、右

结论：6个布局特征中，最小【斜率】特征的归属，则为最优布局

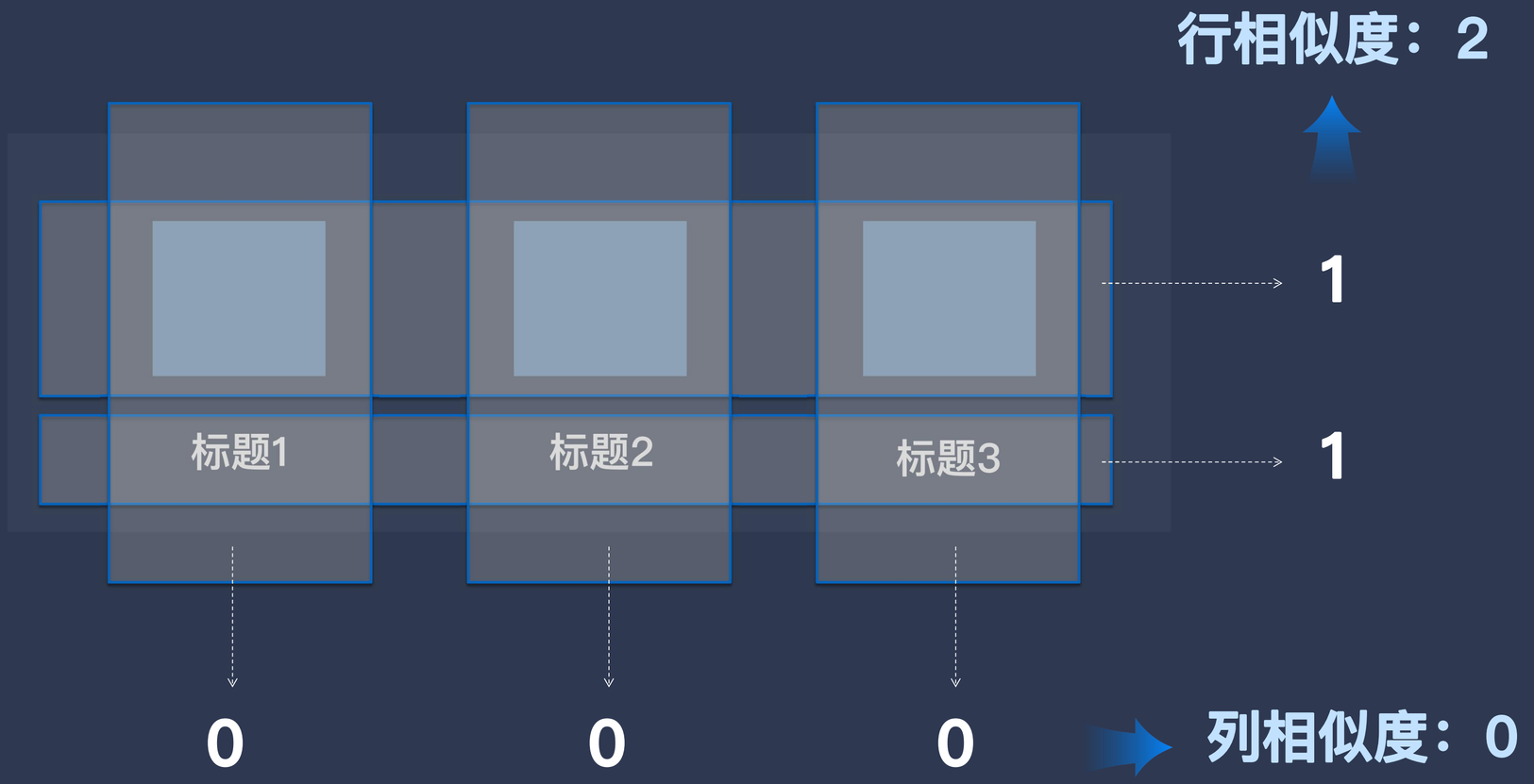
D2C 转码关键实现 / 4.分行分列 / 最优布局算法



思考：当特征【斜率】相等，该怎么分?

D2C 转码关键实现 / 4.分行分列 / 最优布局算法

2.对比行列相似度



结论：行相似度 > 列相似度？ 分列 ： 分行

D2C 转码关键实现 / 思考

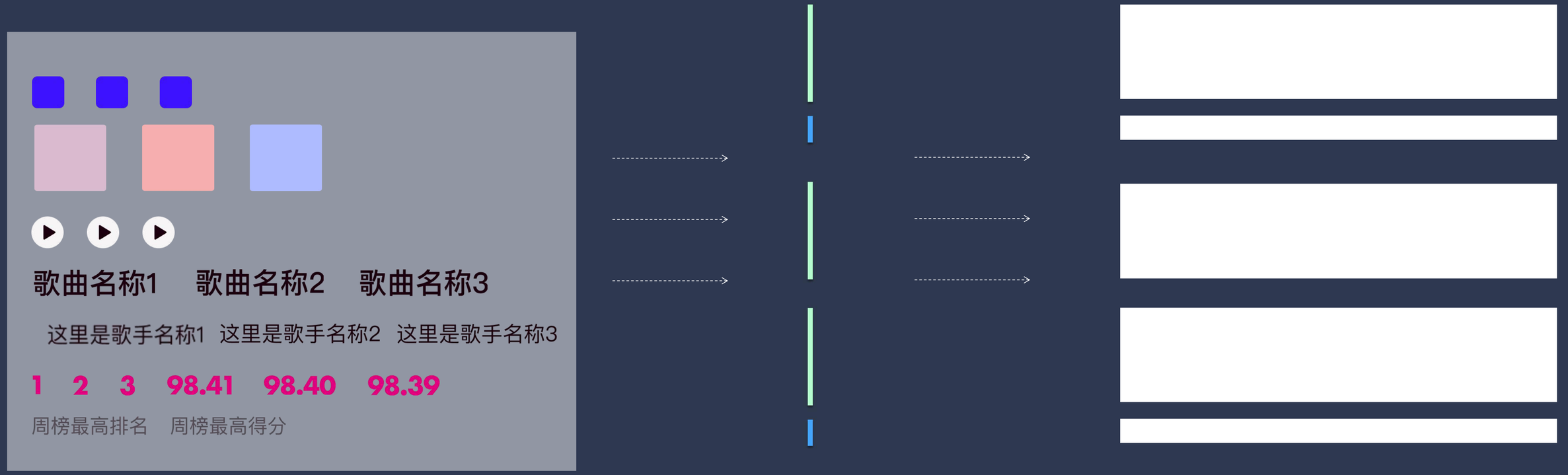
思考：如何提取可循环节点?

D2C 转码关键实现 / 5.提取循环节点 / 提取相似元素

1、根据元素类型将元素进行相似提取

D2C 转码关键实现 / 5.提取循环节点 / 相似投影

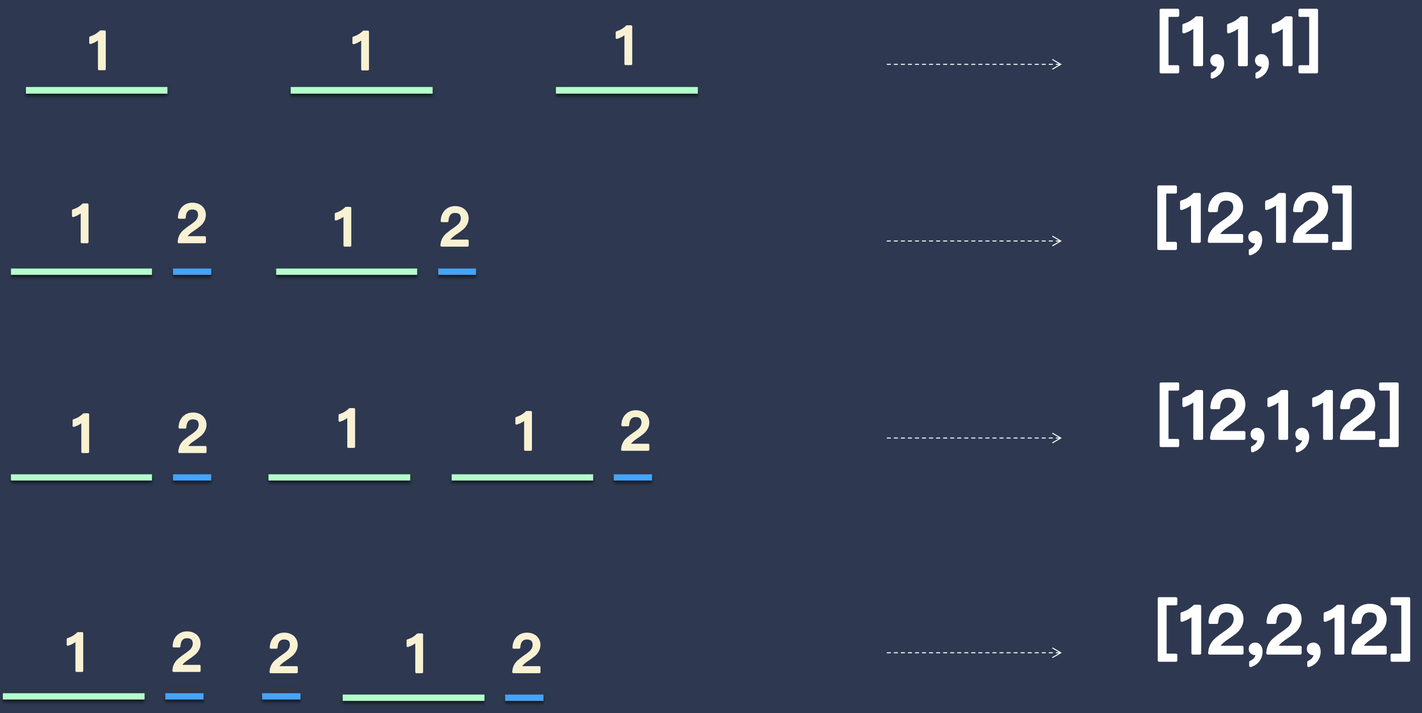
2、对相似元素进行方向投影编组



D2C 转码关键实现 / 5.提取循环节点 / 投影分组

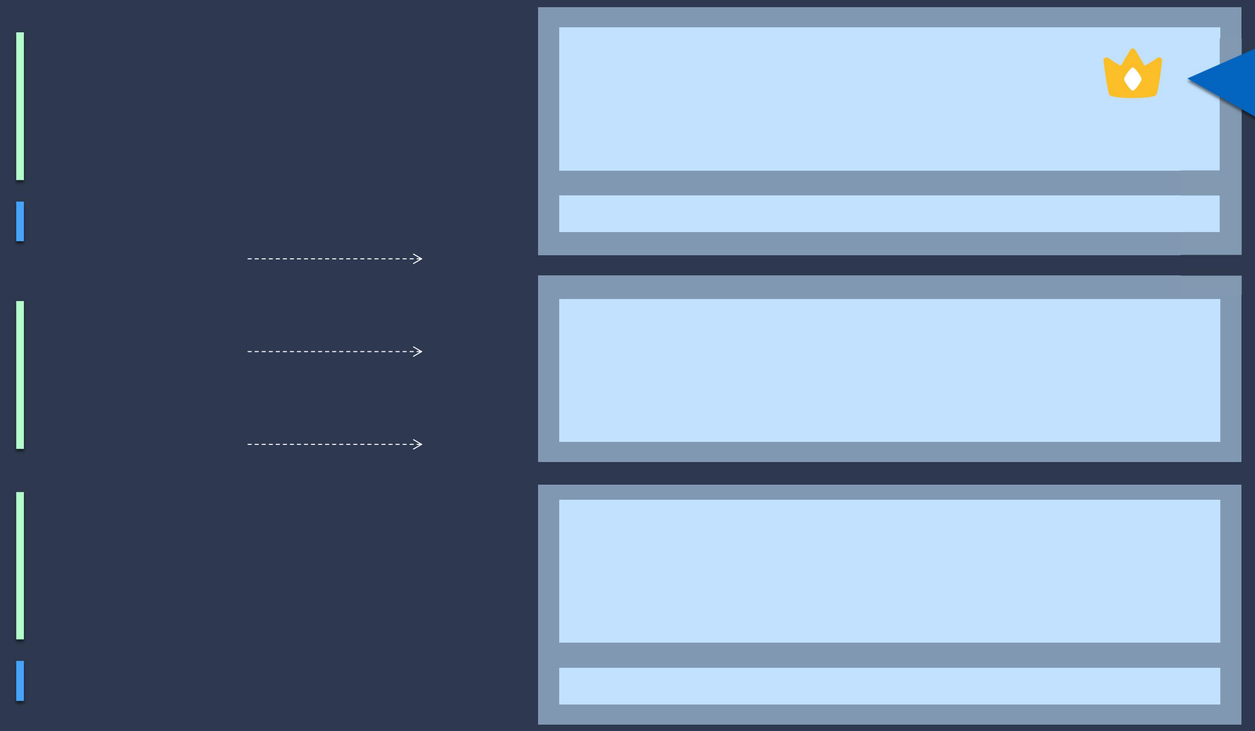
3、对投影特征进行算法分组

规则：分组队列中出现重复项，则单独成组，否则聚合继续检测。



D2C 转码关键实现 / 5.提取循环节点 / 节点归组

4、将相似分组 +噪声元素进行区块还原



噪声元素回填 布局重塑

D2C 转码关键实现 / 6. 转化样式

基础类 baseStyle	宽度　高度　横坐标　纵坐标　旋转角度
效果类 effect	透明度　阴影　填充背景　混合模式
描边类 stroke	圆角　边框样式
文字类 text	颜色　字号　行高　字距　粗细　字体

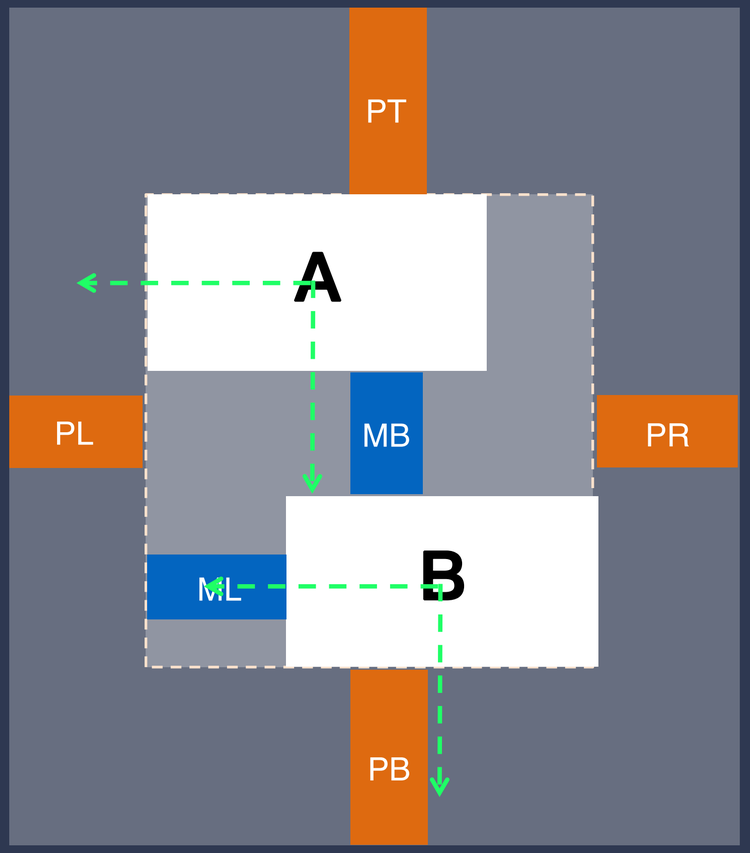
D2C 转码关键实现 / 7.构建骨架

根坐标归一化
（0，0）

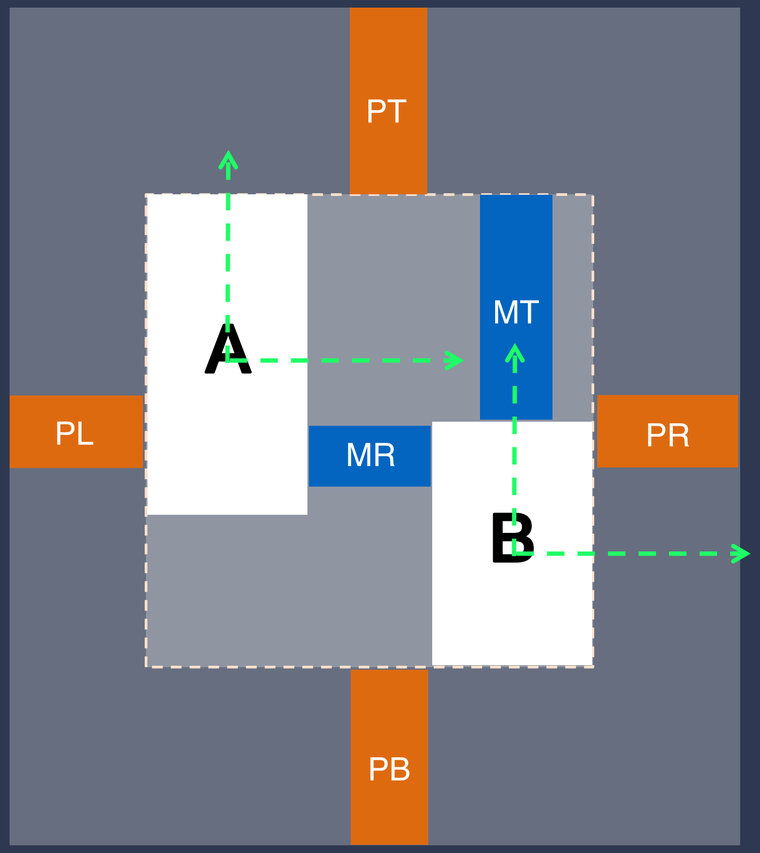
生成绝对坐标
（X，Y）

转换相对坐标

分行

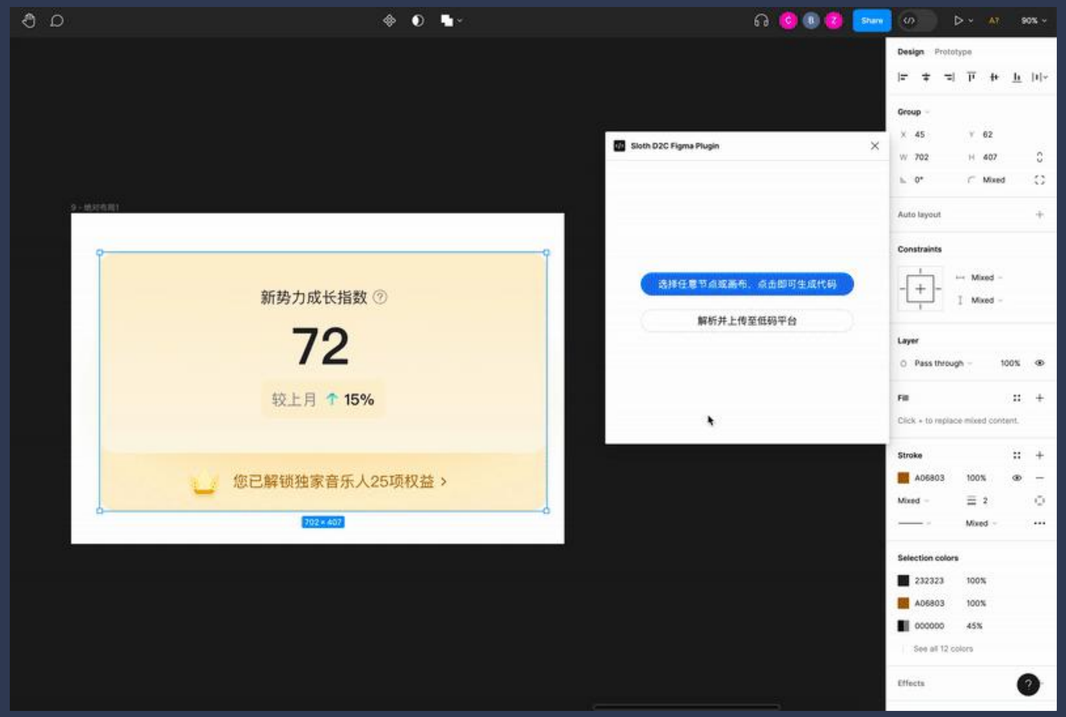


分列



瀑布流的特性：从上而下，从左到右

D2C 转码关键实现 / 8.多平台生成




```
{
    id: "node-1",
    name:"",
    baseStyle: {
        X:10,
        y:10,
        height:10,
        width:10,
        margin: [],
        padding: [],
    },
    effect: {
        opacity: 0.5,
        mixMode:"",
        background: [],
        shadow: [],
    },
    storke: {
        borderRadius: [],
        borderStyle: [],
    },
    text: [{
        content:'',
        color:'',
        fontSize:'',
    }],
    type: 'container',
    layout:'row',
    isAbsolute:false,
    children: [Node, Node, Node]
}
```

Code Schema

Vue

React

Taro

Hippy

理论上 D2C 技术可完成任何图形编码语言的静态UI还原需求

D2C 转码关键实现 / 更多思考

图层集合

- 文本
- 图标
- 图片
- 容器

没有含义和交互

思考：前面只完成切图工作，我们如何知道图层的含义?

3.低码与D2C合力完成“端到端”

3.1 基于 YOLO 完成目标识别

3.2 搭建WEB训练平台

3.3 存量组件识别

识别是什么组件

3.4 低代码组件开发

3.5 低码组件转化器

3.6 人工介入

转化真实组件代码

端到端的演进

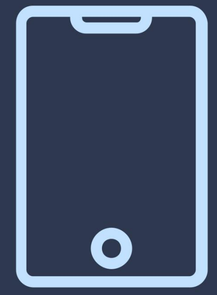
设计端

样式还原是下限

代码还原是上限

1.节点合理布局

2.组件含义识别 + 功能还原

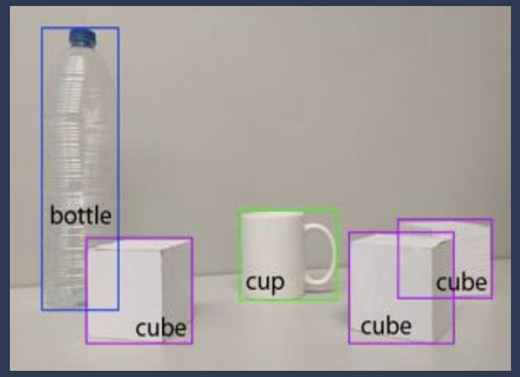


用户终端

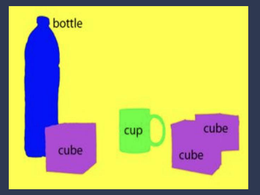
端到端的演进 / 1.基于 Yolo 完成目标检测

目标分类

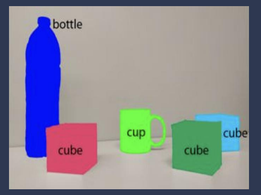
目标检测



语意分割



实例分割



目标检测更符合组件识别的场景，单阶段的 YOLO 算法精度和速度都非常出色。

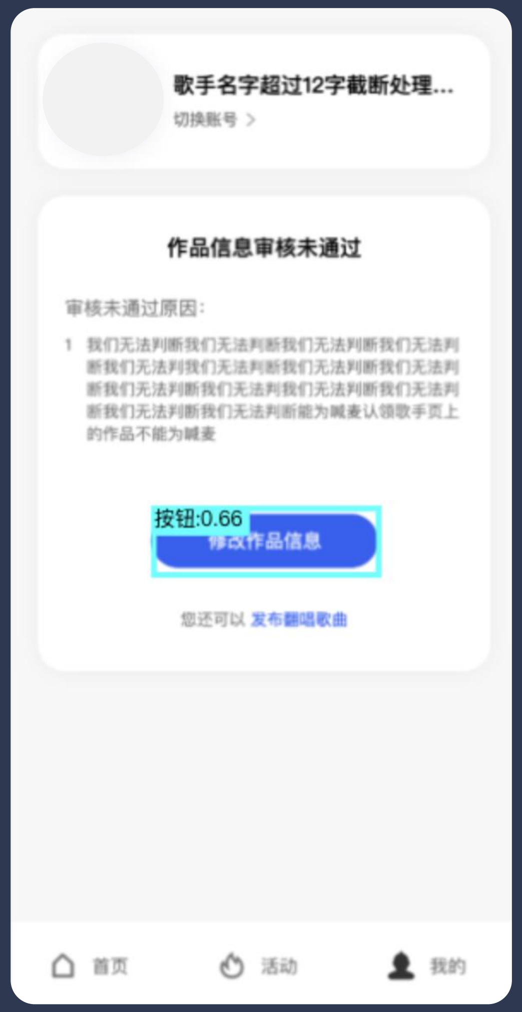
端到端的演进 / 1.基于 Yolo 完成目标检测 / 识别步骤

1. 标注数据图片，收集YOLO训练数据

2. 调整训练参数，进行模型训练

```
├── runs
    ├── detect    # 预测结果
    └── train # 训练的结果
├── data
    ├── VOC.yaml  # 配置训练和评估模型时的相关参数和路径
├── models     # 不同版本的配置文件，配置模型的结构、超参数等
    ├── yolov5l.yaml
    ├── yolov5m.yaml
    ├── yolov5n.yaml
    ├── yolov5s.yaml
    └── yolov5x.yaml
├── detect.py  # 选择不同的模型版本和权重文件用于预测
├── export.py  # 将训练好的模型导出为其他格式
├── train.py   # 指定的数据集对YOLOv5模型进行训练
└── val.py     #指定数据集对训练好的YOLOv5模型进行验证
```

3. 调用预测模型



问题：如何将模型训练和预测 应用在WEB端?

ArchSummit 全球架构师峰会
InfoQ 极客传媒

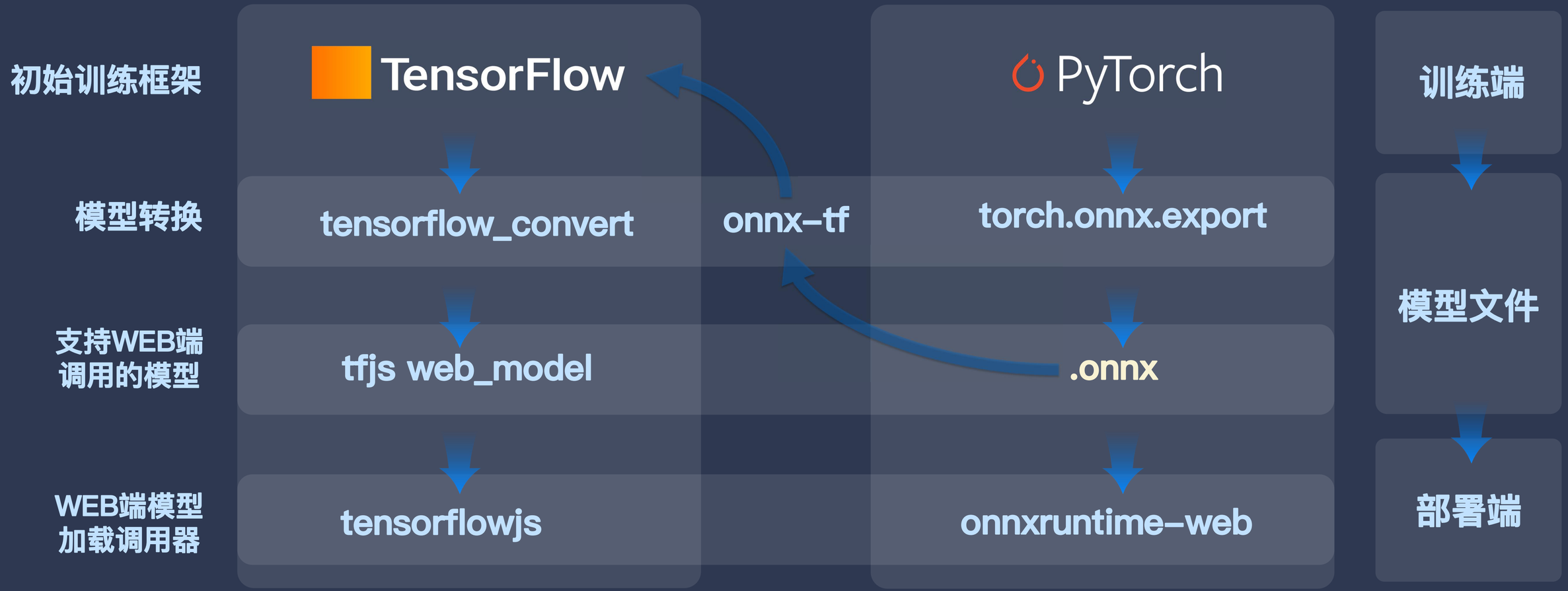
端到端的演进 / 2.构建训练WEB平台

共享训练素材

组件间互不干扰

团队间互不干扰

端到端的演进 / 2.构建训练WEB平台 / WEB模型导出应用



端到端的演进 / 思考

思考：难道所有的组件，都要经过模型训练，才能检测识别?

模型训练量暴增，数据收集困难

端到端的演进 / 3.存量组件识别 / 采集标记

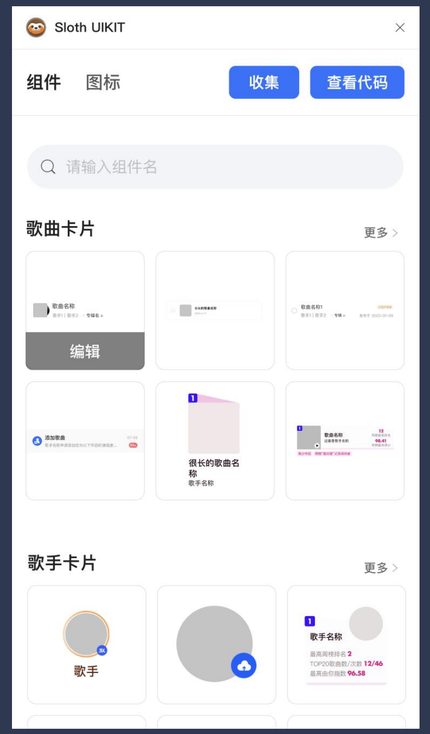
识别方案1：标记

设计师采集节点

选中图层 → 节点清洗 → 节点存储

设计师应用节点

选中组件 → 拖拽至画布 → 节点还原



端到端的演进 / 思考

思考：存量设计稿中，应用的组件太多，无法全量标记?

识别方案2：视觉算法

端到端的演进 / 3.存量组件识别 / 视觉算法

1.边缘提取

去除色彩干扰，提高匹配准确度

业务组件

设计稿快照

灰阶图

组件的透明背景灰阶后是黑色

端到端的演进 / 3.存量组件识别 / 视觉算法

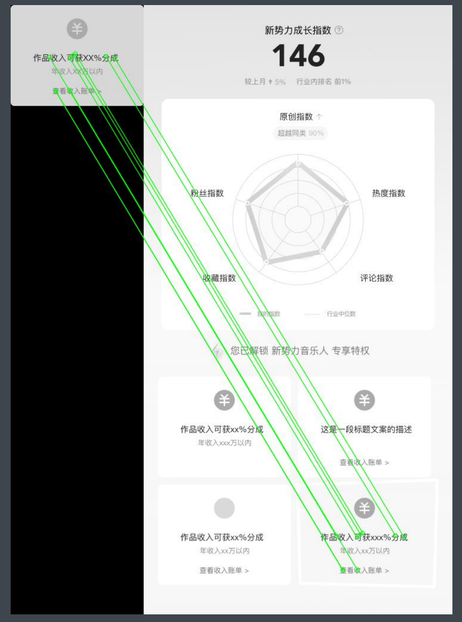
2.模版匹配

计算组件图与设计稿相似度高的位置

阈值0.7

阈值0.4

SIFT算法

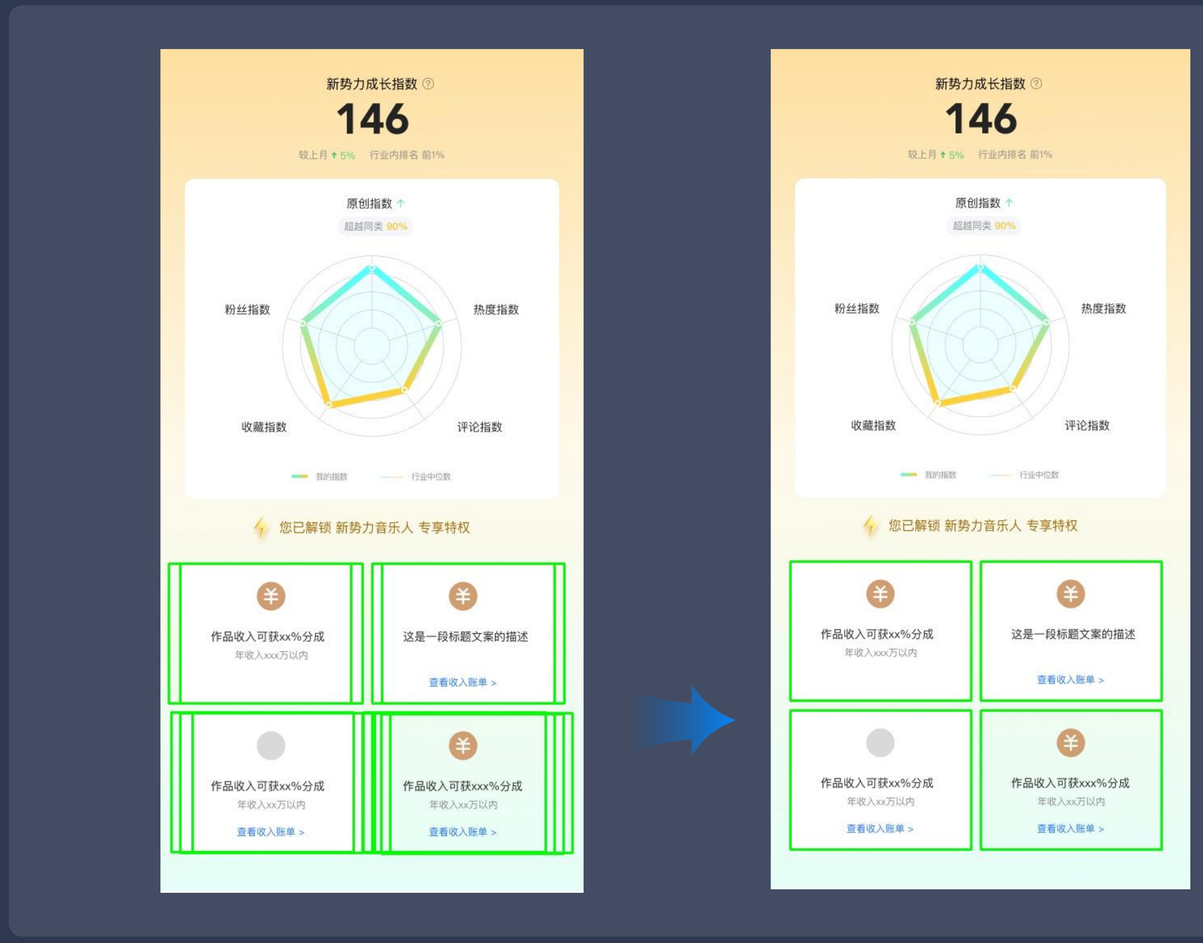


额外确定组件透明背景的边界

端到端的演进 / 3.存量组件识别 / 视觉算法

3.非最大值抑制

NMS去除重复检测结果和保留最优目标框



端到端的演进 / 3.存量组件识别 / 总结

基于 CNN 的目标检测模型

适用于变化多，尺寸、形状不固定的组件，如按钮，表单等组件。

基于视觉计算的检测算法

适用于尺寸固定，差异较小的组件，如业务卡片等固化的搭配。

端到端的演进 / 图层如何转代码?

含义：按钮

组件代码

工程代码

根据标识替换对应源码

接入低代码?

端到端的演进 / 4.低代码组件开发

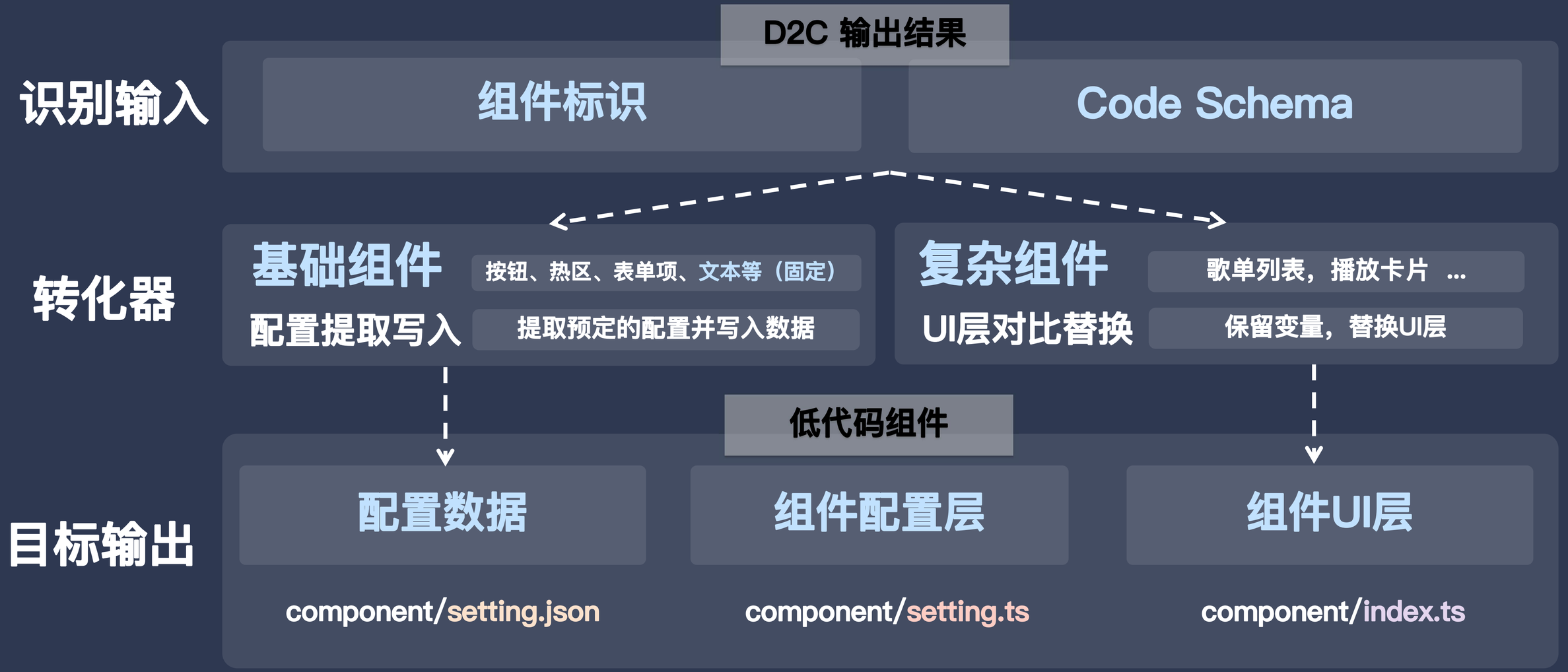
组件开发规范

子组件目录

- UI 代码
- 配置代码
- 预设配置数据

低码系统

端到端的演进 / 5.低码组件转换器



端到端的演进 / 思考

思考：完成了图层转译，组件映射，是否完成了全链路?

缺少了真实场景上下文辅助，算法覆盖度约在95%。

组件间，与载体都会有逻辑关系，独立组件不足以满足完整需求。

依托低码平台，人工介入来完成最后一公里

端到端的演进 / 6.人工介入 / 预设配置

各业务团队可以根据自己的工程化需求，进行定制转化配置

- 预设全局脚本
- 尺寸单位
- COS地址&前缀
- 特殊字体上传

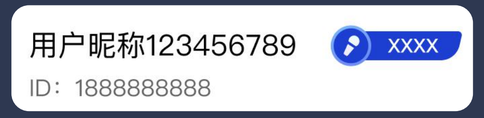
端到端的演进 / 6.人工介入 / 调整转化结果

节点删除 / 隐藏

处理冗余算法误伤的节点

行列二次拆合

行列算法斜率、相似度的误差



上下文和开发习惯

二次节点编组

优化分组算法不正确划分

自定义合图

解决合图算法误差&补充字体转图

端到端的演进 / 6.人工介入 / 低码能力复用

逻辑绑定

事件编排 | 循环渲染 | 条件渲染 | 自定义方法

数据源绑定

事件编排 | 自定义变量 | 条件渲染 | 自定义方法

元素配置

属性编辑 | 样式编辑 | 动画预设 | 容器插槽

工程化能力

版本管理 | 性能优化 | 埋点监控 | 远程调试

端到端的演进 / 6.人工介入 / 低码能力复用 / 逻辑绑定

让逻辑轻松被理解，低码交接无负担，逻辑快速被定位。

```ts
// logic 的定义
declare type SComponentLogic = {
  // 暴露的方法，被别人调用，由组件定义
  defineExpose?: LabeledValue[];
  // 自身的事件，由组件定义，如 onClick
  defineEvent?: LabeledValue[];
  // 记录自身的事件调用了其它组件的方法，通过连线配置产生
  relation?: SComponentLogicRelation[];
  // 自定义函数
  fns?: SComponentLogicFn[];
};

// 编辑语义化，通过label配置展示名称
declare type LabeledValue = {
  // 实际值
  value: string;
  // 展示名称
  label: string;
};

// 记录自身的事件调用了其它组件的方法或者函数，通过连线配置产生
declare type SComponentLogicRelation = {
  when: LabeledValue['value'];
  // 被调用的组件id
  callId: SComponentId;
  // 被调用的组件的方法（defineExpose），或者函数（fns）
  call: LabeledValue['value'];
  // 调用后使用 withFn 来转换参数
  withFn?: SComponentLogicFnName;
  // 执行条件
  condition?: string;
  // 执行层级，默认0 无执行顺序关系
  order?: number
};

declare type SComponentLogicFnName = string;
declare type SComponentLogicFn = {
  // 自定义的函数名称
  name: SComponentLogicFnName;
  // 函数类型，可以是props更新器，参数更新器
  type: string;
  // 函数体
  content: string;
};
```

4.总结与展望

业务回顾

业务端

音乐人开平

由你榜单

生态厂牌

涉及多端
提升30%人效

低码端

宣推活动

线下演出

管理后台

上传设计稿

低码提效的基础上
再次提升30%

整体架构回顾

静态转化
desgin-to-ui-code

清洗设计稿
- 冗余元素删除
- 元素转换
- 蒙层裁剪
- 图标合并

还原节点关系
- 父子关系
- 分行分列
- 求最优布局策略
- 识别循环节点

生成多端代码
- 构建样式
- 构建骨架
- 全局元素注入
- 多平台生成

动态映射
layer-to-component

在线模型训练&预测
数据采集 → 模型训练 → 模型验证 → 模型应用

CV算法匹配推测
特征提取 → 模版匹配 → 阈值调整 → 二次确认

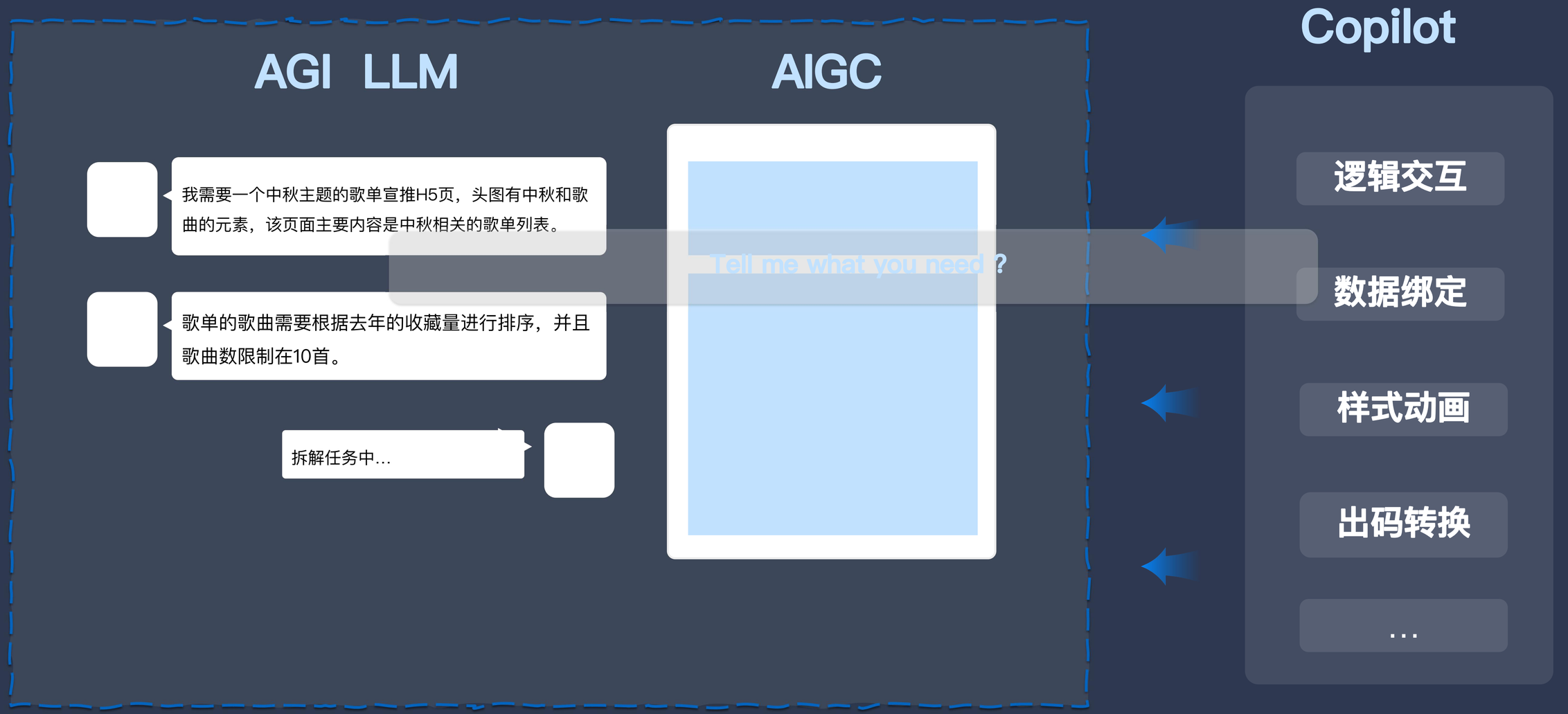
接入低代码
code-to-development

组件转化器
- 低码组件开发
- 配套转化器
- 渲染变量提取

人工介入，二次调整
- 预设尺寸调整
- 节点二次重组
- 基础属性调整
- 节点二次合并

基础能力复用
- 数据源接入
- 逻辑编排
- 动画设置
- 版本管理

未来展望


AGI LLM
AIGC
Copilot
我需要一个中秋主题的歌单宣推H5页，头图有中秋和歌曲的元素，该页面主要内容是中秋相关的歌单列表。
歌单的歌曲需要根据去年的收藏量进行排序，并且歌曲数限制在10首。
拆解任务中…
Tell me what you need ?
逻辑交互
数据绑定
样式动画
出码转换
…

THANKS