
GOTC 2023
全球开源技术峰会
THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE
OPEN SOURCE, INTO THE FUTURE
「大前端新趋势」专场
万物智联下的应用框架的思考和探索
余枝强 2023年05月28日

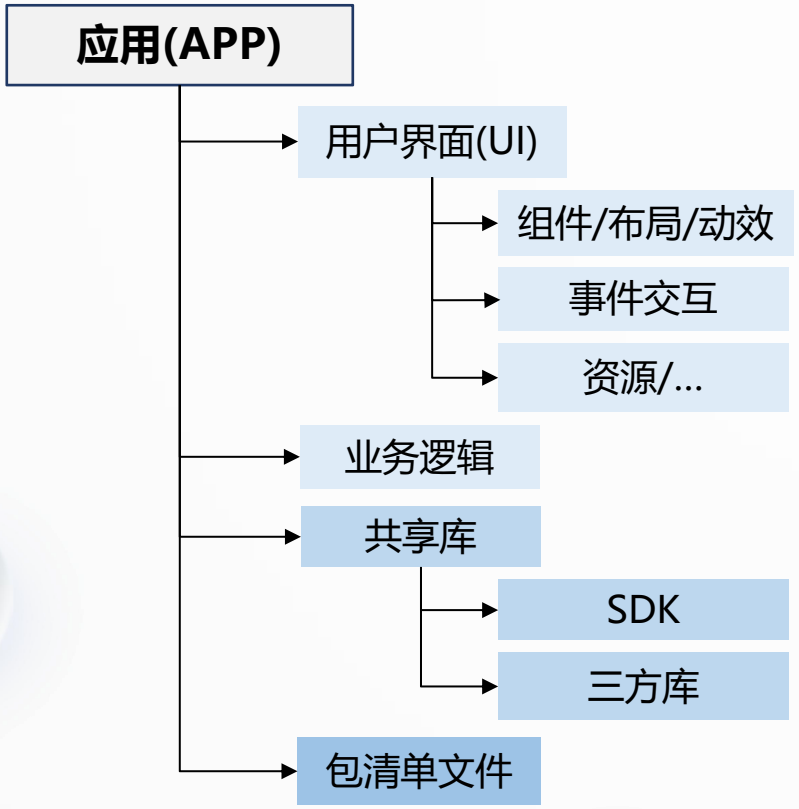
业界应用框架
的演进

万物智联下的
应用框架设计思考

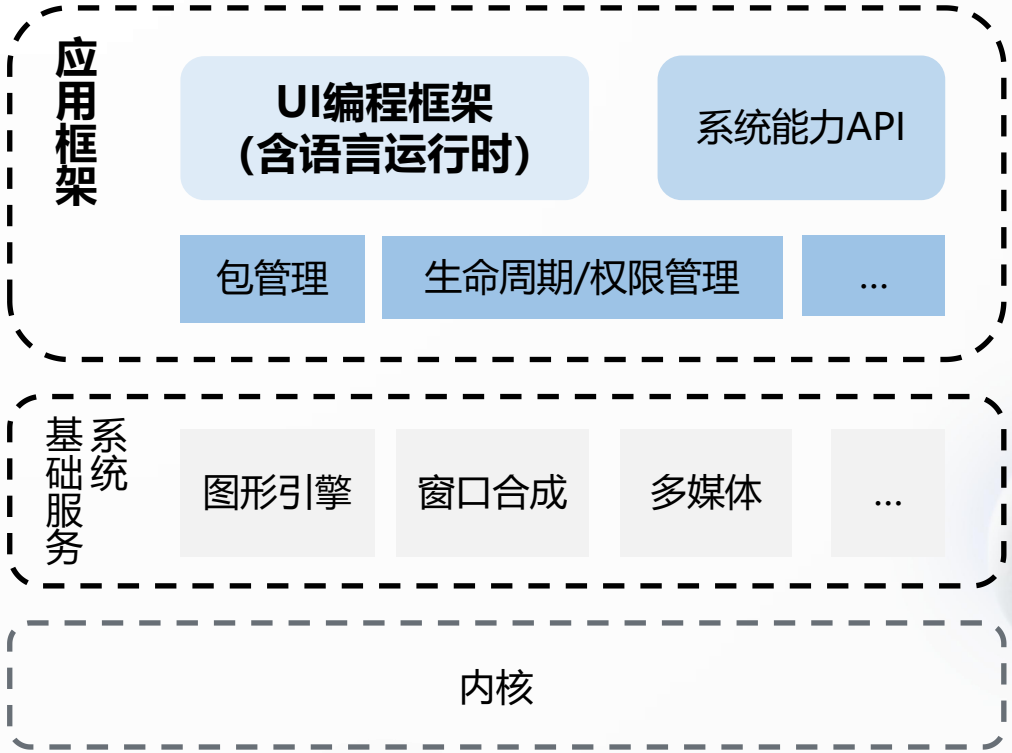
ArkUI的实践
创新和生态建设

应用的组成 - 以移动端的应用为例

典型的应用结构



系统运行环境



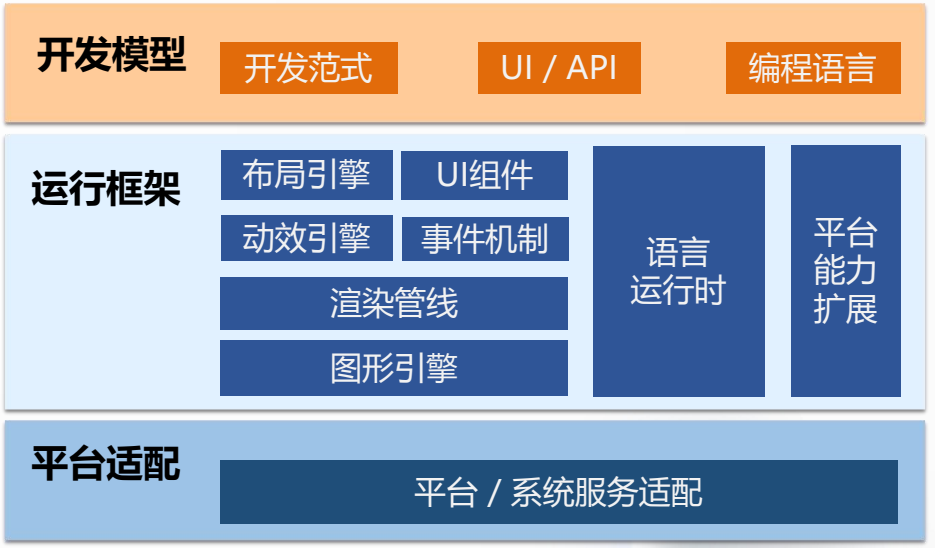
UI编程框架概览

开发者视角
- 编程语言
- 开发界面
- 业务逻辑

用户视角
- 视觉
- 交互
- 体验

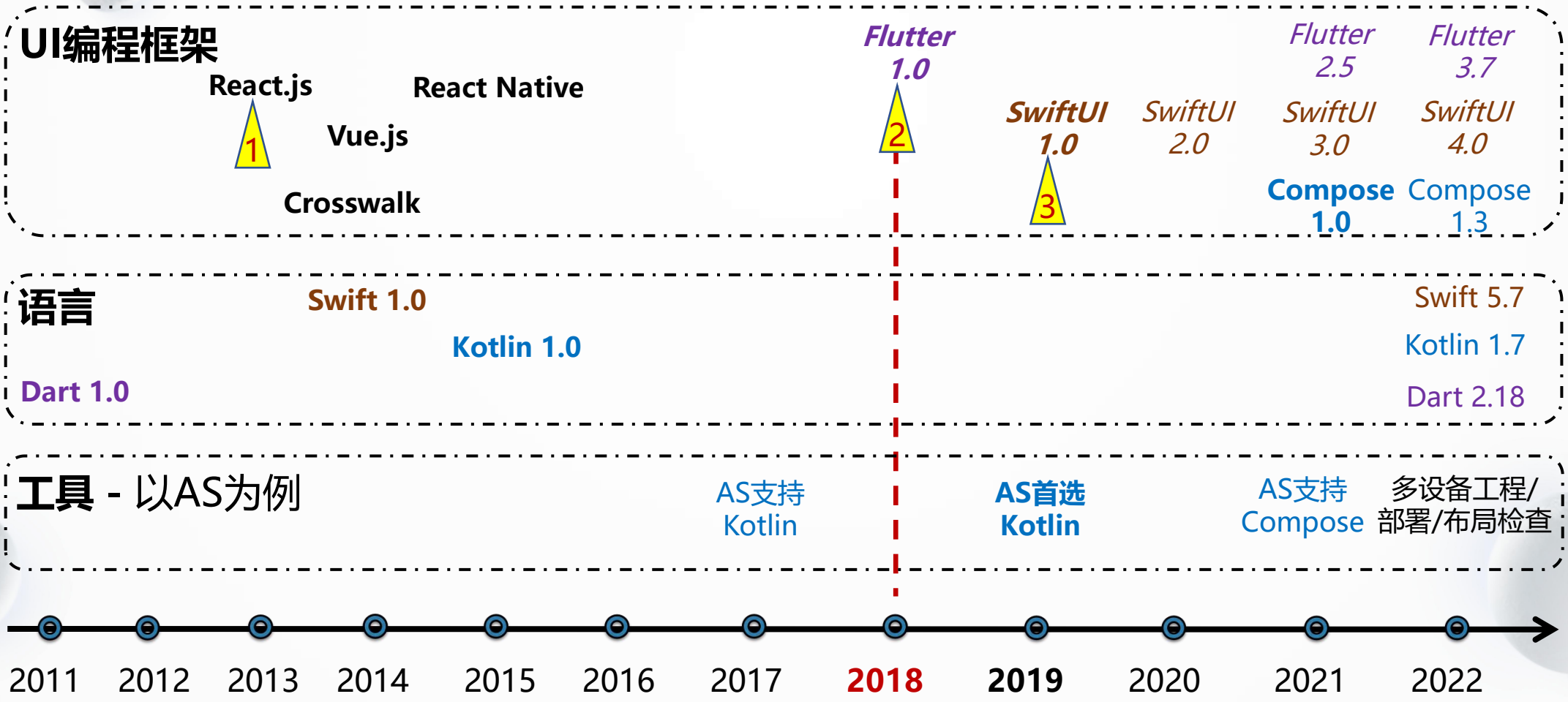
UI编程框架

系统视角
- 运行环境
- 图形显示



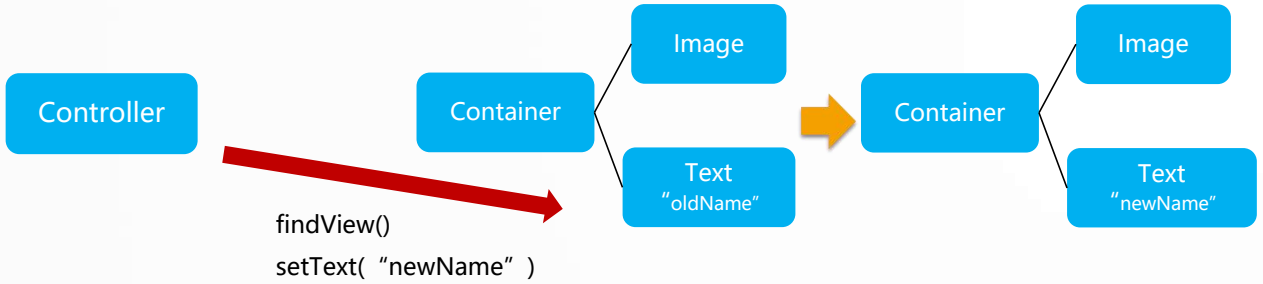
UI编程框架演进概览

AS : Android Studio ; **Compose** : Jetpack Compose

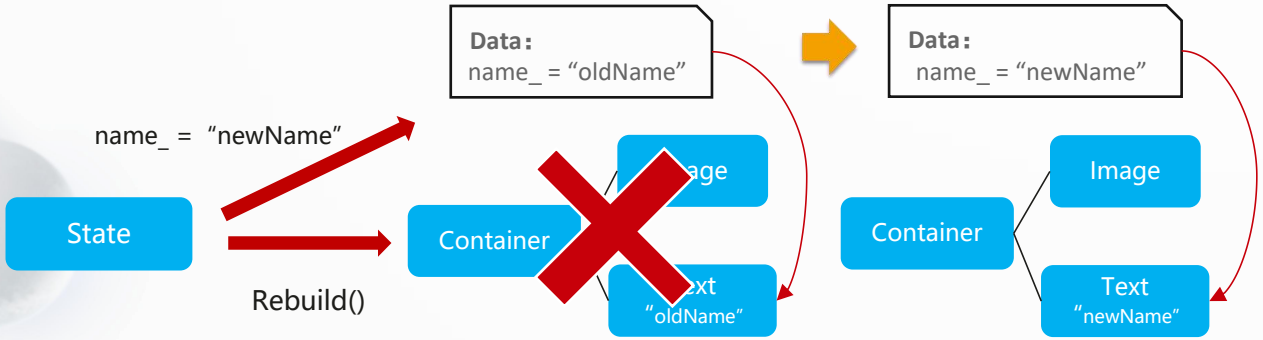


命令式UI vs 声明式 UI

命令式UI – **面向过程**



声明式UI – **面向结果**



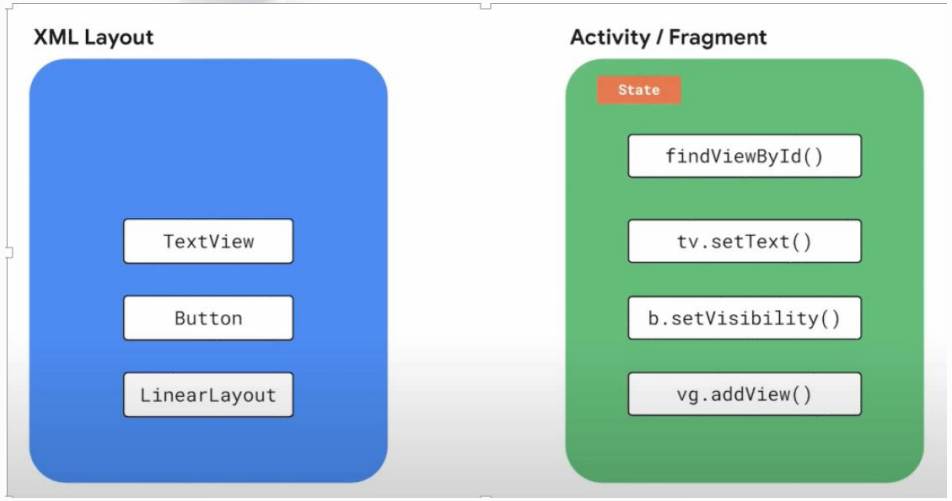
更多能力:

条件循环/响应式数据绑定/高级组件...

更直观，更高效，开发者只要关注核心数据即可

Apple，Google 都选择**声明式UI**作为新一代应用开发模式， 并在**底层重构**相应的UI组件设计

Android UI编程框架的演进 (View -> Jetpack Compose)

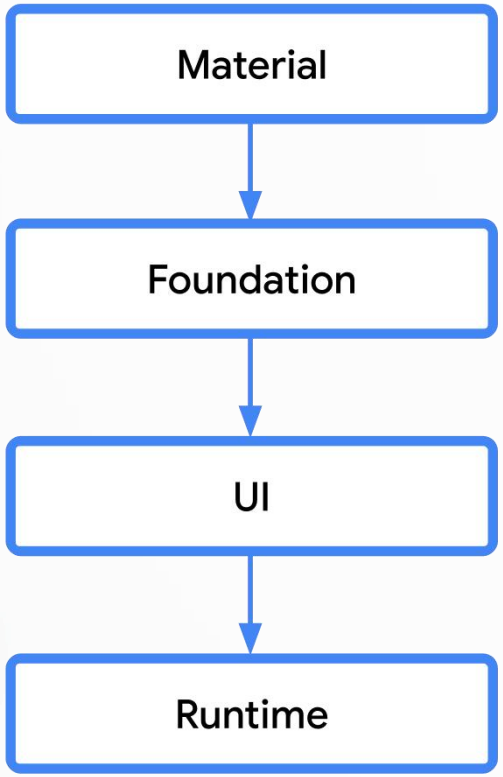



```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.h2,
                )
            }
        }
    }
}
```

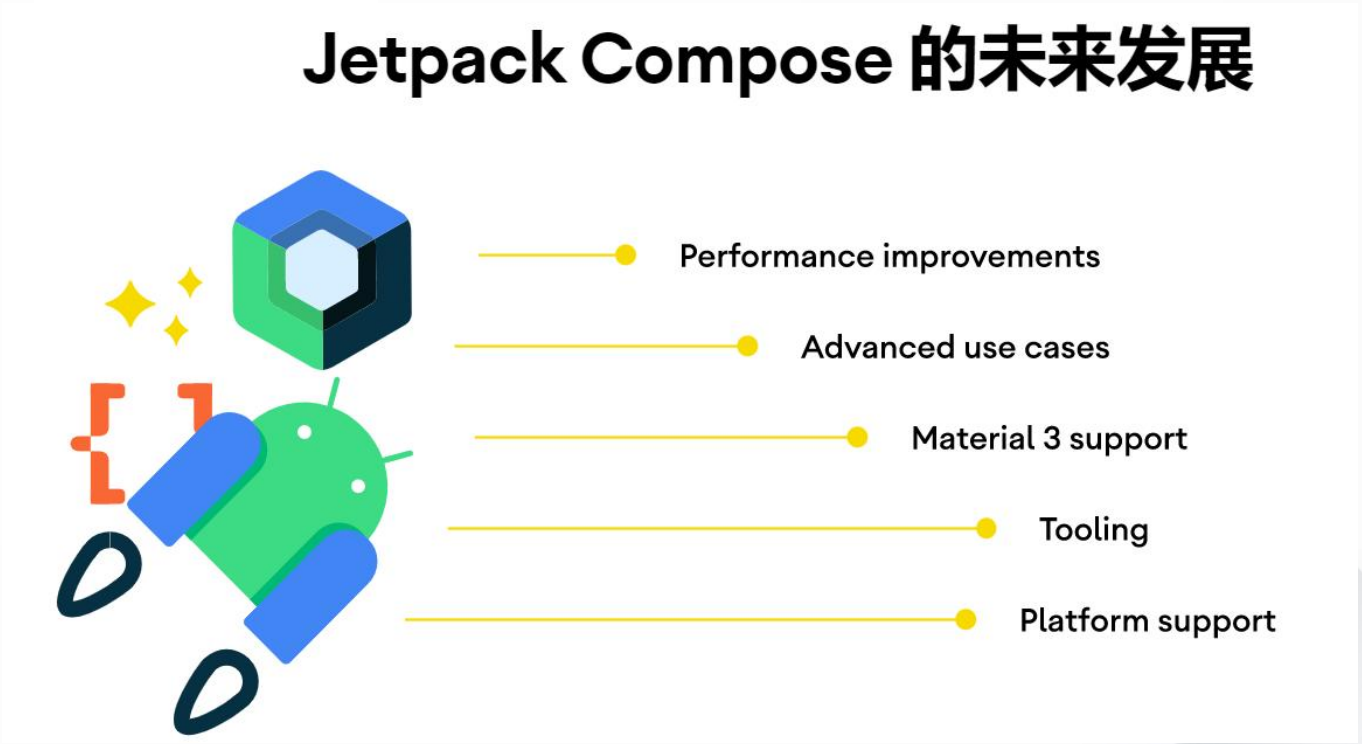
(来源：https://developer.android.com/jetpack/compose/layering?hl=zh-cn)

Jetpack Compose架构分层以及路线图

架构分层

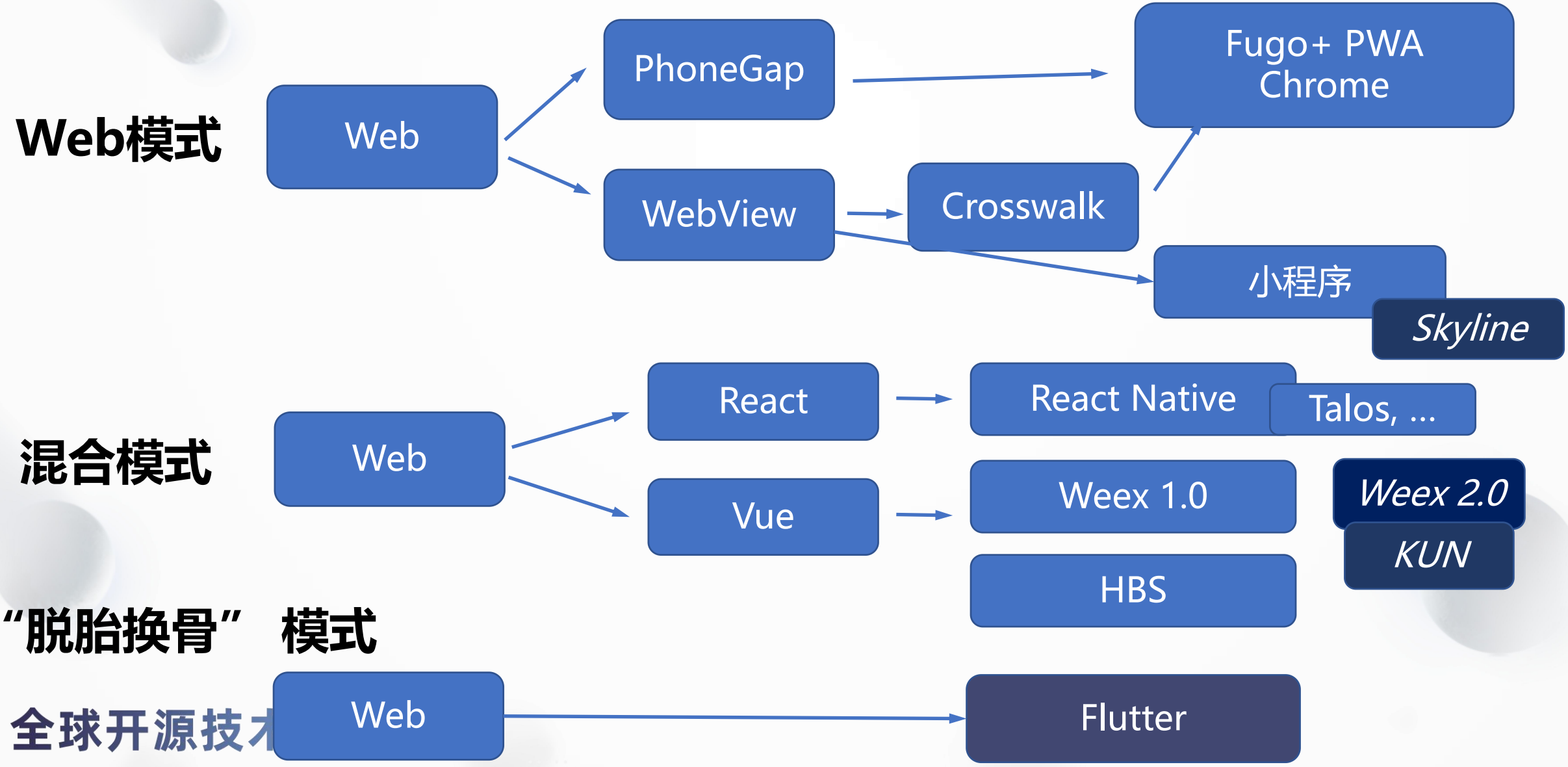


路线图

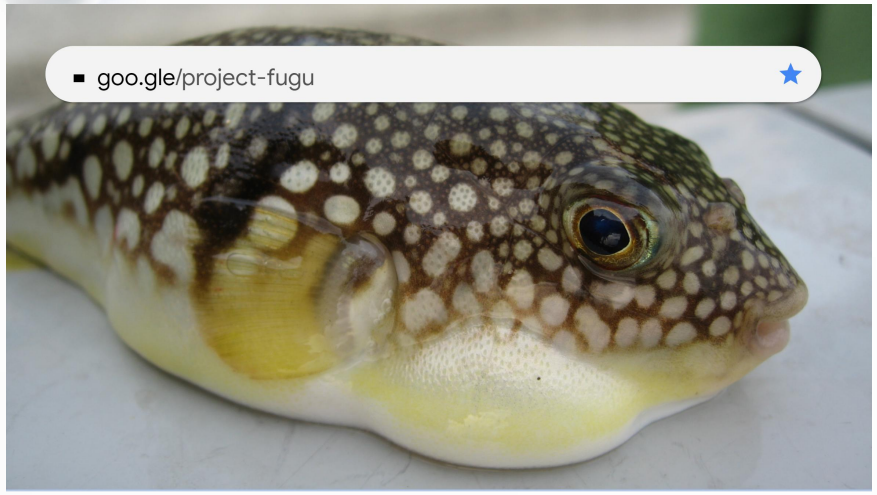


(来源：https://developer.android.com/jetpack/compose?hl=zh-cn)

Web类应用框架的演进

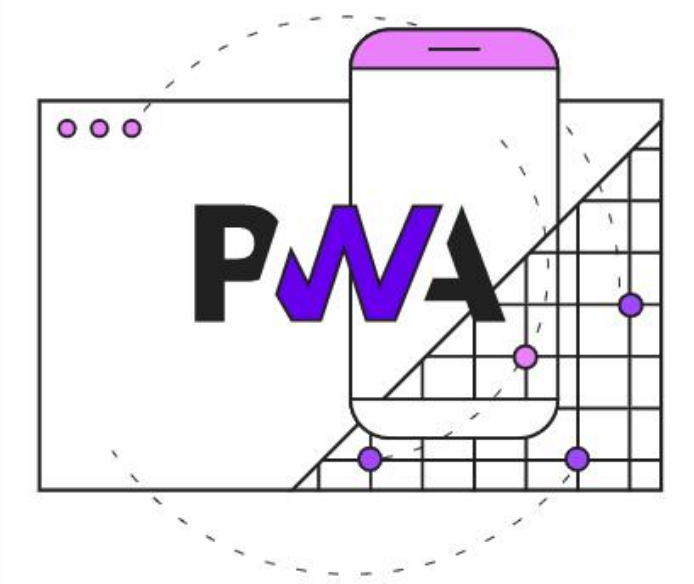


Google & W3C的类似项目（依托浏览器）



Fugu – 河豚

不断增强的JS API
–通讯录、文件读取、物体识别...

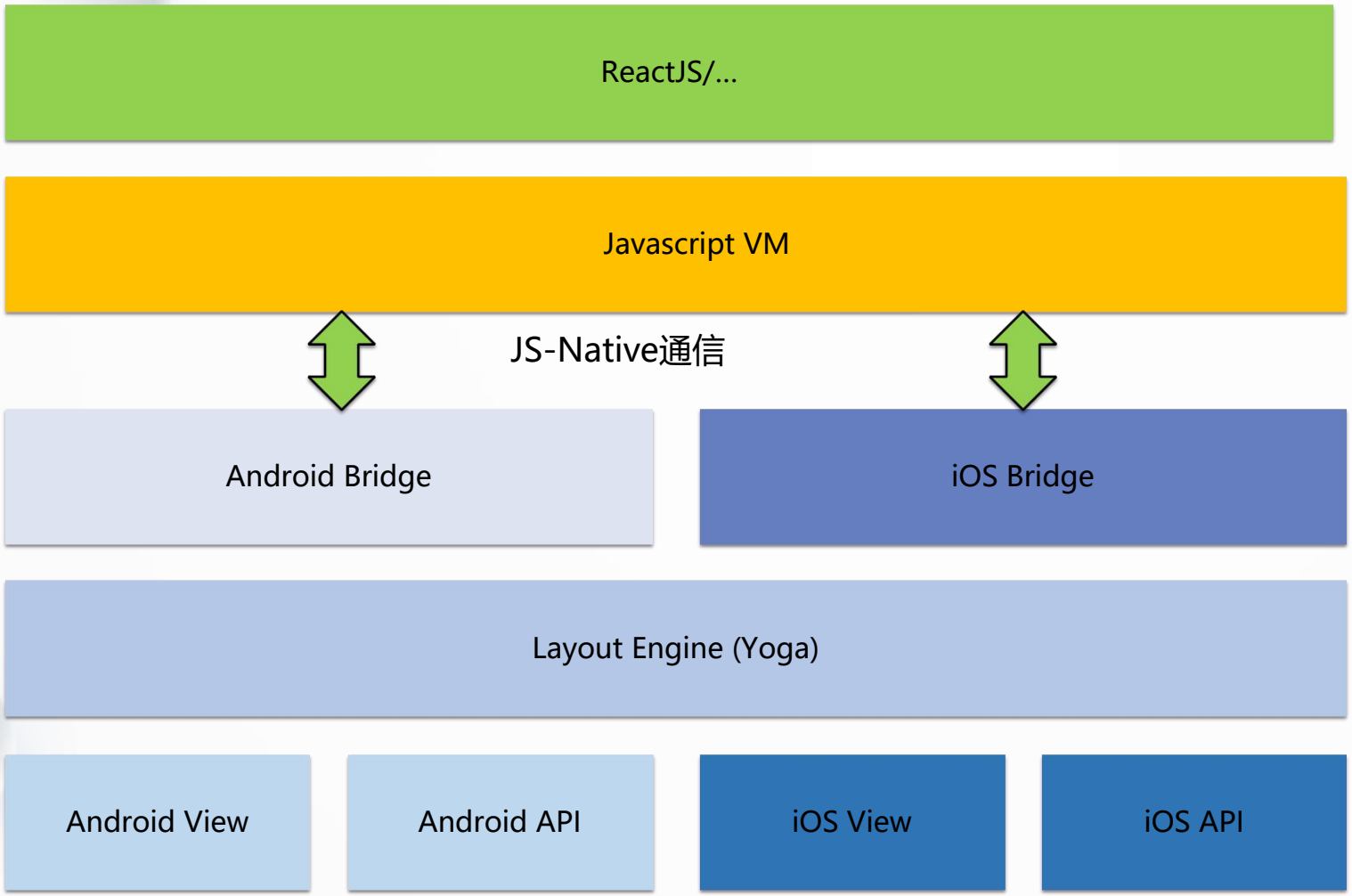


PWA – 渐进式 Web 应用

可安装
可离线访问
可和系统整合

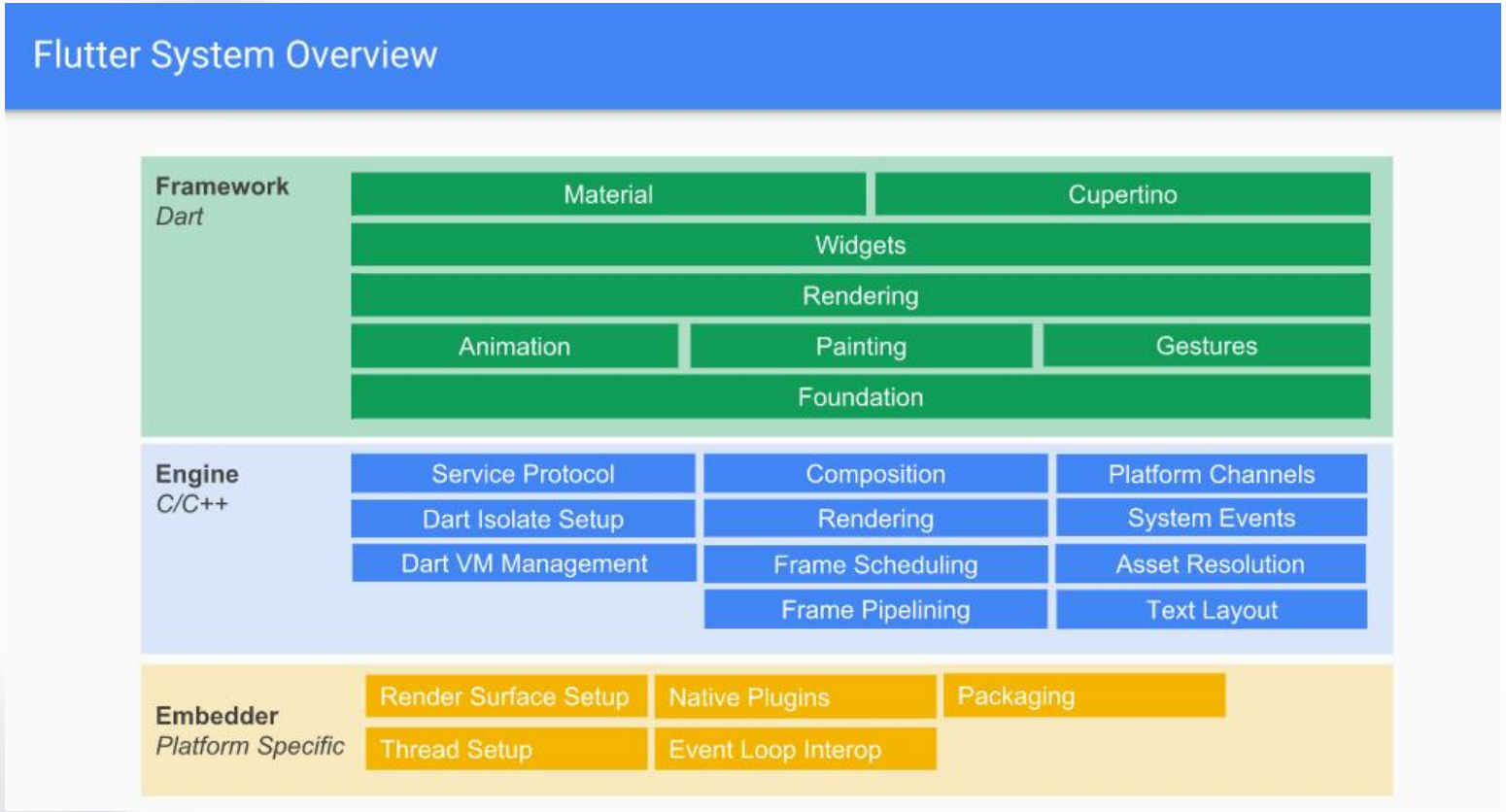
来源：https://www.chromium.org/teams/web-capabilities-fugu/
https://web.dev/progressive-web-apps/

ReactNative架构概览



- JS-Native机制
- 布局引擎Yoga
- Fabric渲染器
- Hermes引擎
- ...

Flutter架构概览



Framework

框架层是由Dart实现的SDK，包含基础的组件

Engine

引擎层由C++实现，包含运行渲染的基础能力

Embedder

平台嵌入层负责将Flutter框架桥接到不同平台

语言特性

- 语法与C++/Java接近，是类型确定的语言
- 支持async异步编程
- 通过Pub管理软件包，共享生态
- 面向对象的设计以及语法

运行调试

- 开发时可JIT运行，快速调试，支持hotReload
- 发布时采用AOT编译成机器码，性能接近原生语言

声明式UI

- 语法上适合UI开发，代码更简洁
- 轻量化Widget/精简布局等提升效率
- 短期对象管理优化，快速对象分配和分代垃圾收集器

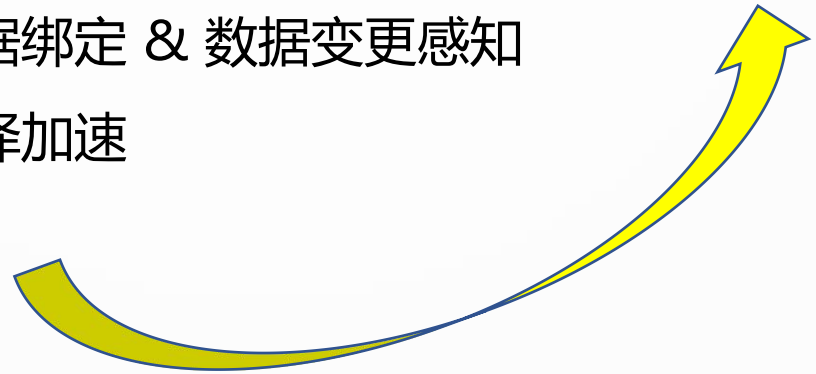
跨平台

- 可编译成ARM和x86的字节码，支持iOS/Android/Windows/macOS/Linux
- 在标准Web平台可以转换为JavaScript运行

(图片来源于Flutter官网)

业界UI编程框架的演进小结

- **开发工具向多设备演进**
  - 多设备UI
  - 实时预览、组件级预览等
- **UI编程框架和语言融合增强**
  - 数据绑定 & 数据变更感知
  - 编译加速
- **UI编程框架的变迁**
  - 从命令式UI到声明式UI
  - 跨平台能力提升
  - 个性化，多设备能力



万物智联下的新场景和新需求

GOTC

1. 不同形态的设备

- 不同屏幕
  (分辨率/长宽/形状/尺寸/...)
- 不同交互
  (触控/键鼠/遥控/语音/3D手势/...)

2. 不同能力的设备

- 不同处理能力
  (CPU/GPU/NPU/蓝牙/...)
- 不同规格
  (RAM/ROM 百KB~GB/...)

3. 不同设备之间的交互

- 无缝流转/协同/...

4. 多个主流OS的并存 → 跨OS

5. 内容快速部署的需求 → 动态化内容部署机制

如何设计

相应的应用框架?

编程语言

UI编程框架

根

API & 三方库

IDE

主干

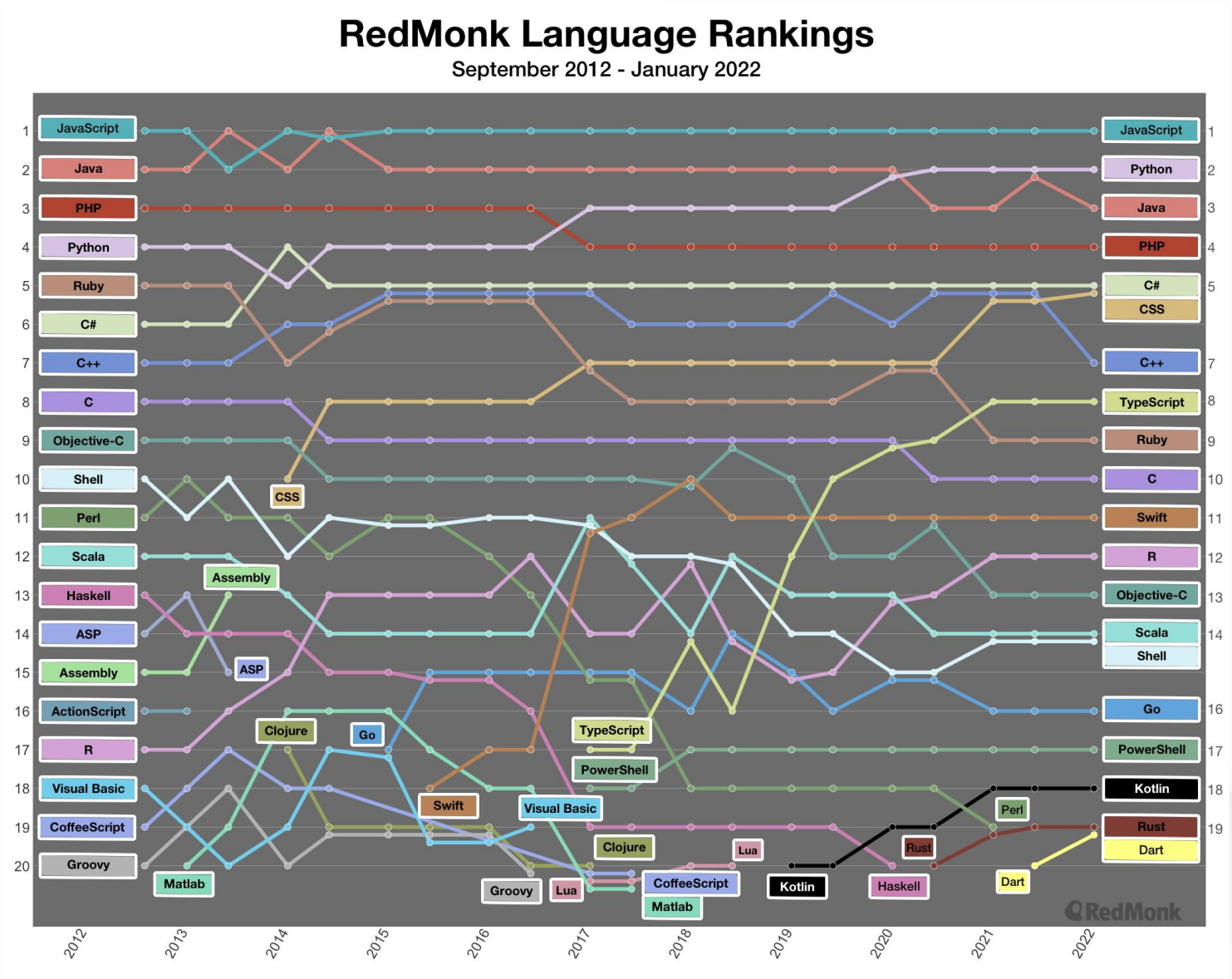
编程语言

编程语言热度排行榜

2012-2022

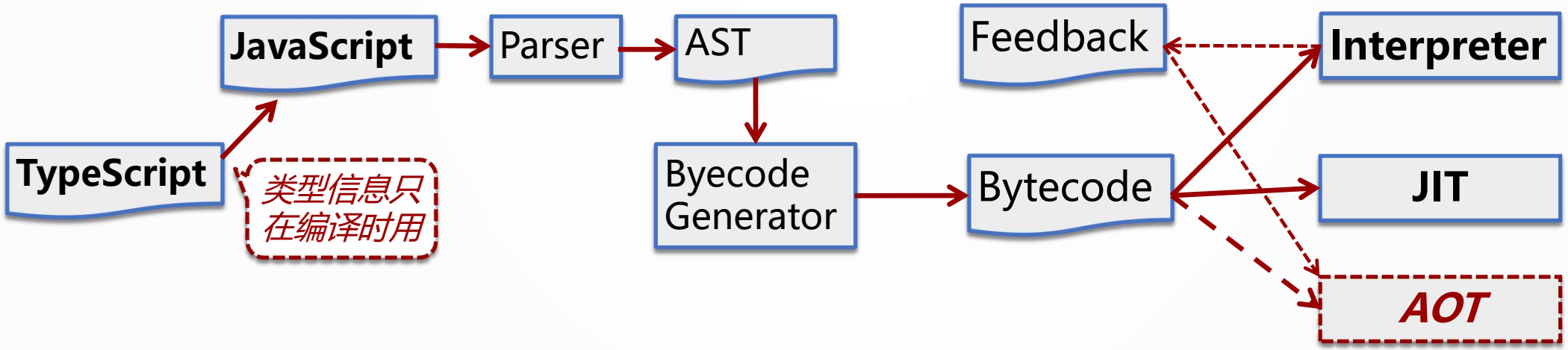
综合考虑了Github & StackOverflow

JS/TS持续领先



图片来源于RedMonk官网： https://redmonk.com/rstephens/2022/03/28/top-20-jan-2022/

对标原生：JS/TS的现状以及不足



主要不足：

- 未能在运行时有效利用类型信息；缺失精细化类型
- AOT缺失
- 细粒度的并行化缺失

UI编程框架

对标原生：相关UI编程框架的现状以及不足

关键技术：标准Web渲染模式，响应式前端框架
典型代表：**Web引擎**

资源占用高，性能较弱

关键技术：JS+原生组件直通渲染，轻量布局引擎&JS引擎
典型代表：**React Native**

依赖原生的渲染路径
语言性能问题&桥接层损耗
平台一致性弱

关键技术：自绘制，AOT，组合式按需渲染
典型代表：**Flutter**

语言生态较弱，开发范式的简洁度不够
如需兼容JS生态需额外桥接，性能和内存都有额外损耗

主要不足

- 性能体验，以及性能体验和生态的平衡
- HTML+CSS+JS三段式开发方式和业界领先的声明式开发（SwiftUI）

相比**有较大差距**，需进一步提升

业界相关的改进持续进行，

但缺失系统性的跨越！

架构设计思考

单平台开发框架要解决的核心问题

开发效率
- 开发范式
- 语言/基础库/三方库
- 调试、调优、预览

性能体验
- 性能/内存/功耗
- 能力完备度

跨设备
- 不同设备的UI适配
- 不同设备的能力适配

+

跨平台开发框架要解决的核心问题

开发效率

不同平台上的代码复用度

性能体验

不同平台的性能

不同平台的体验一致性

+

其他要素

动态化内容机制

应用部署

SDK包大小

ArkUI – 鸿蒙生态的原生开发框架

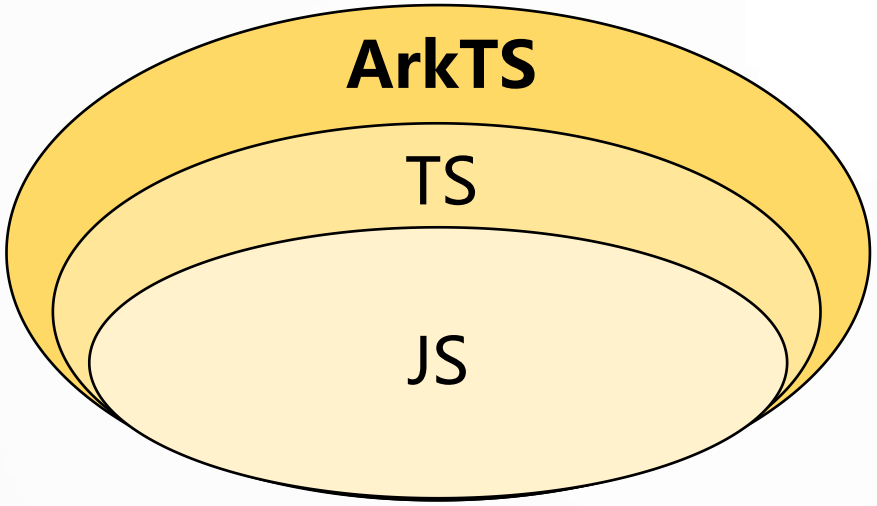
ArkUI开发语言&运行时： 基于JS/TS进一步演进出ArkTS

- 自研编译器/运行时 – **类型信息优化，AOT**，…
- 在JS/TS基础上，扩展了**声明式UI语法**
- 进一步扩展更多语言/标准库的能力 – **轻量级并行/分布式/严格类型**等

ArkUI框架设计： 极简开发/高性能/跨设备跨平台

- 新一代声明式开发范式
- 一体化渲染管线
- 轻量化，组件化，跨平台化

ArkTS – 鸿蒙生态的原生开发语言



声明式语法，轻量并行

类型系统

基础逻辑，异步，动态

围绕着**极简表达，极致性能**演进

声明式语法扩展

```
@Entry
@Component
struct MyComponent{
    @State msg : string =  'Hello ArkUI'
    build(){
        Column(){
            Text(this.msg)
            .fontSize(100)
        }
    }
}
```

- 组件化，入口定义
- 自定义build函数
- 装饰器

```
@State, @Prop, @Link

@Provide, @Consume
```

```
AppStorage, LocalStorage

PersistentStorage, DistributedStorage

Environment
```

- 多维状态管理

（组件内、组件间、页面间、全局、分布式）

- 单向传递，双向传递

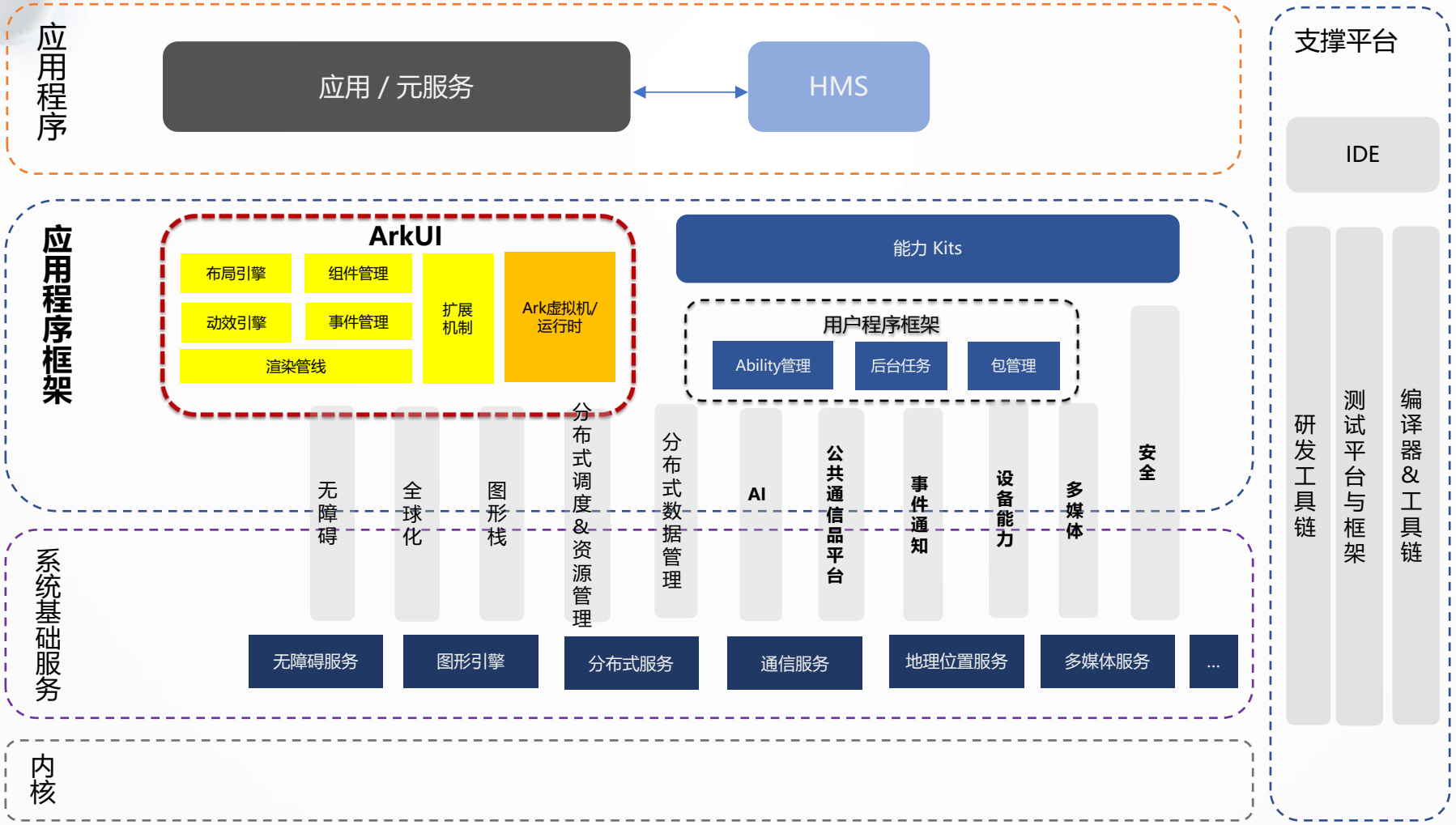
```
@Extend(Text) function fancy(fontSize: number) {
    .fontColor(Color.Red) .fontSize(fontSize)
}
```

```
@Builder SquareText(label: string) {
    Text(label) .width(1 * this.size1).height(1 * this.size1)
 }
@Builder RowOfSquareTexts(label1: string, label2: string) {
    Row() {
        this.SquareText(label1)
        this.SquareText(label2)
    }.width(2 * this.size1) .height(1 * this.size1) }
}
```

- 动态扩展组件属性
- 动态Build，运行时动态创建内容

极简描述 → 多维状态 → 动态组合

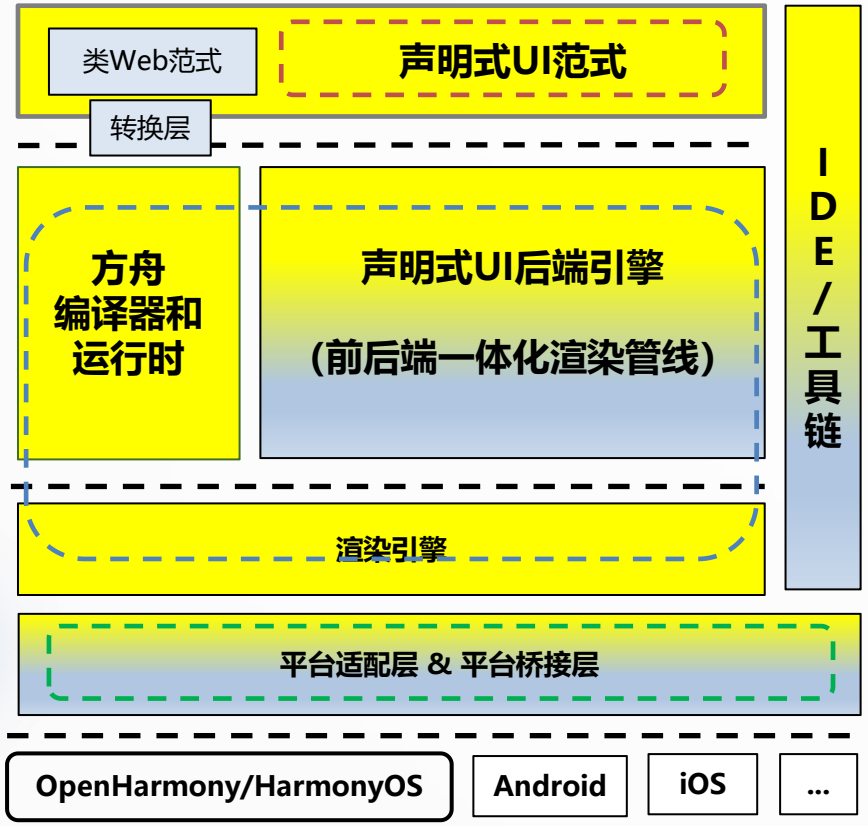
ArkUI开发框架在OpenHarmony中的位置



ArkUI开发框架概览

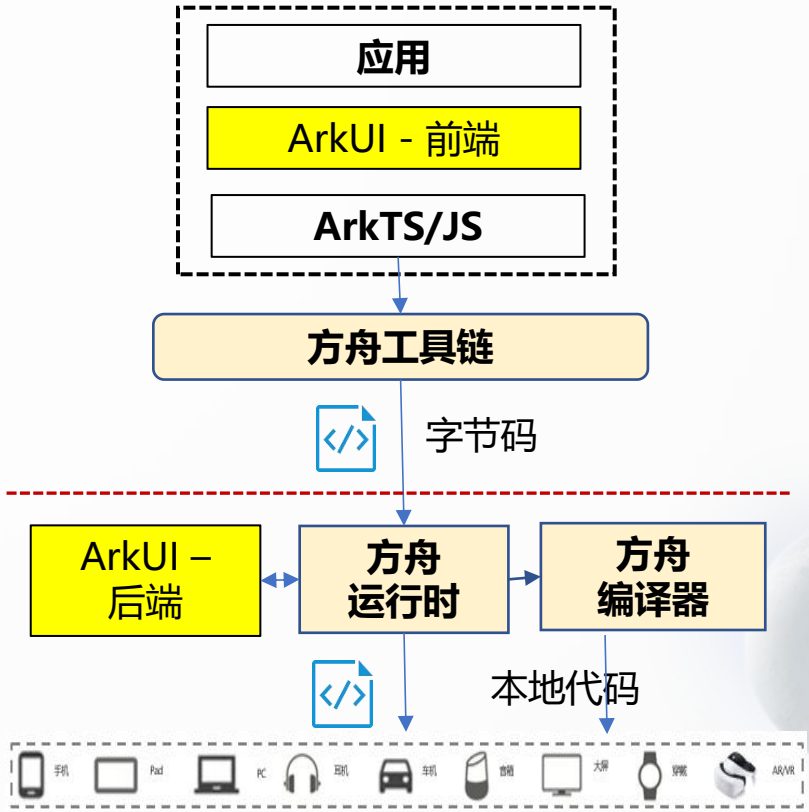
ArkUI

(声明式范式 + 高性能声明式后端引擎 + 一致性渲染)

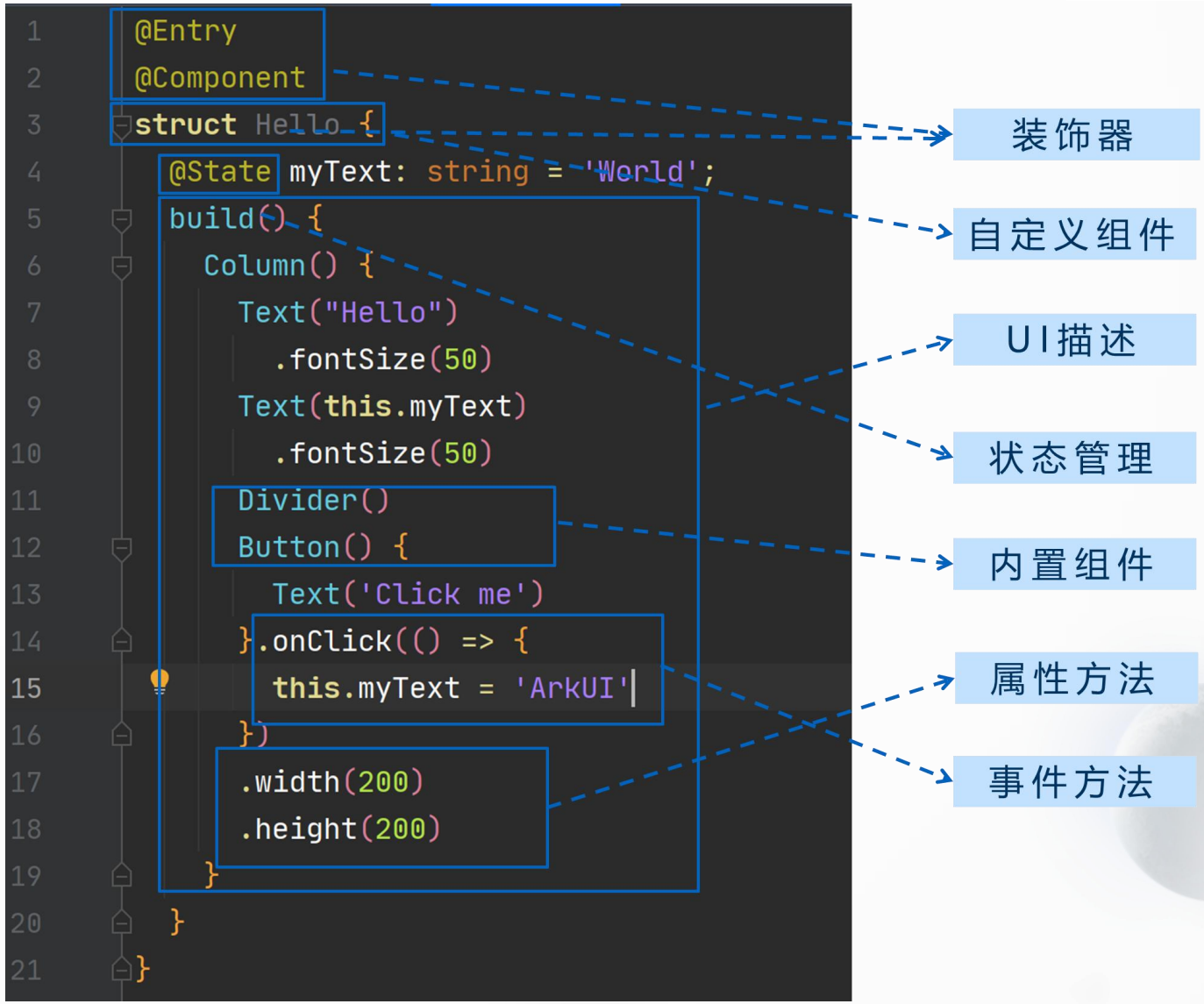


ArkCompiler

(统一字节码 + 高效FFI + 类型加速)



ArkUI开发范式概念简介



ArkUI开发范式关键特征示例

1、极简自然声明式语法，统一UI表达

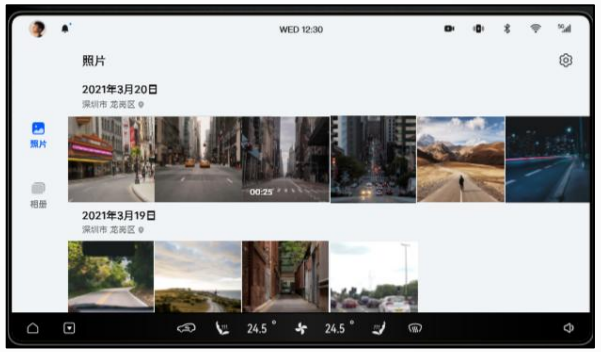
2、多态组件/动态布局，简化多设备UI适配

3、多维度状态管理，简化数据传递&UI变更

类自然语言的UI结构描述，丰富的开箱即用的多态组件，积木式组件组合

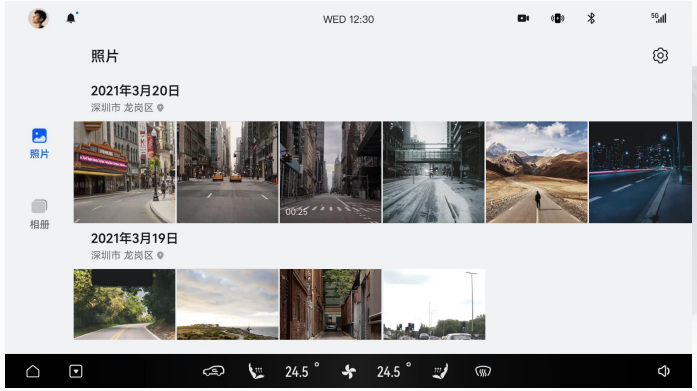
```
@Component
struct IndexPage {
  build() {
    Row() {
      TabBar()
      Tabs() {
        TimelinePage()
        AlbumSetPage()
      }
    }
  }
}
```

```
@Component
struct TimelinePage {
  build() {
    TabContent() {
      Stack() {
        Grid() {
          LazyForEach(data, (item) => {
            GridItem() {
              ImageGridItemComponent(item)
            }
.....
}
```

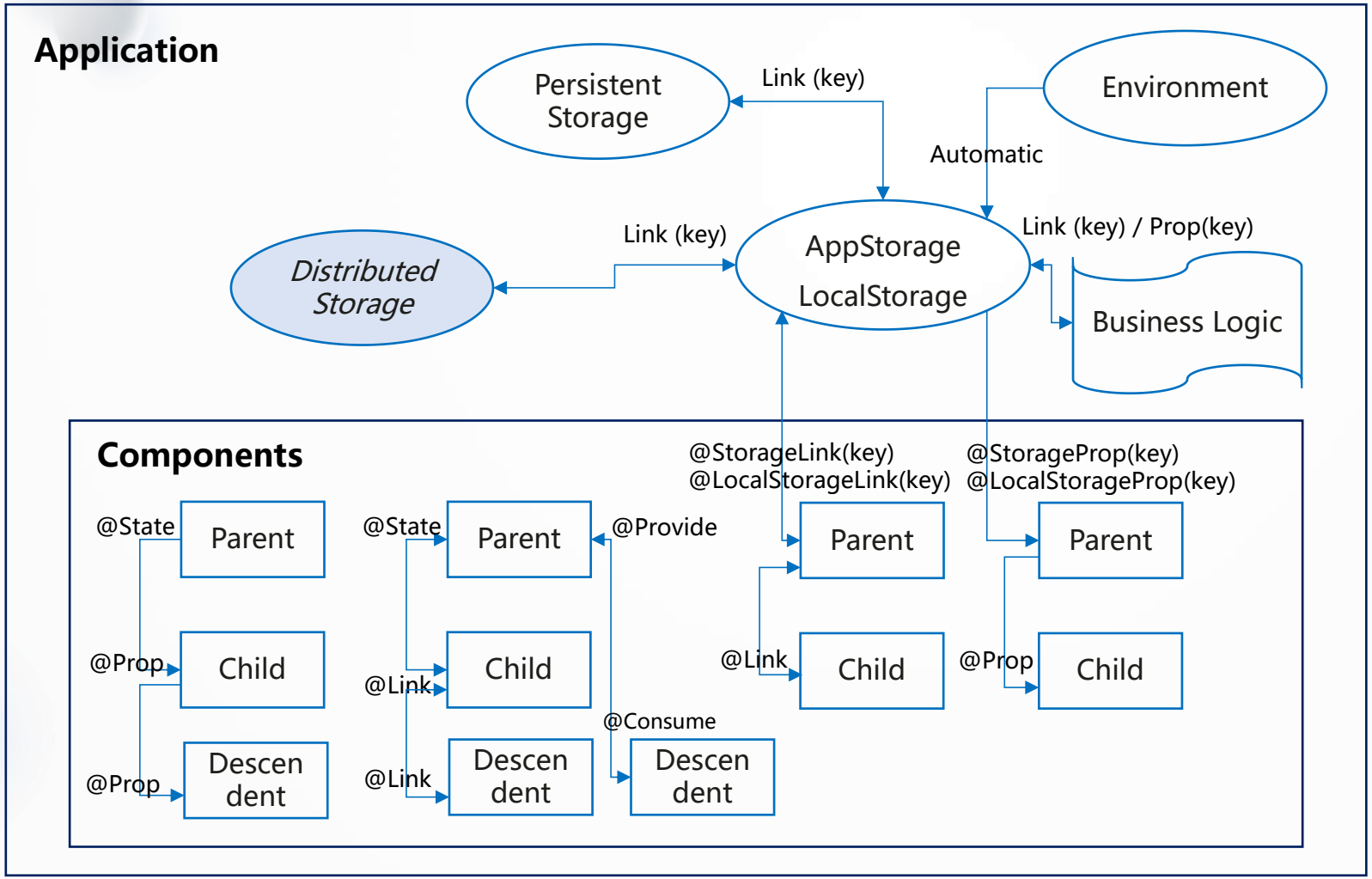


极简语法 – 动效

```
@Component
struct ImageGridItemComponent {
  build() {
    Stack() {
      Image(this.pixelMap)
        .sharedTransition(this.shareId)
    }.onClick(() => {
      // 路由跳转
      router.push({uri:'/pages/PhotoItem'})
    })
  }
}
```



ArkUI多维状态管理 – UI<>数据



多维状态管理：
组件，页面，全局，*分布式*

数据传递方式：
单向，双向

ArkUI运行机制概览

源代码:

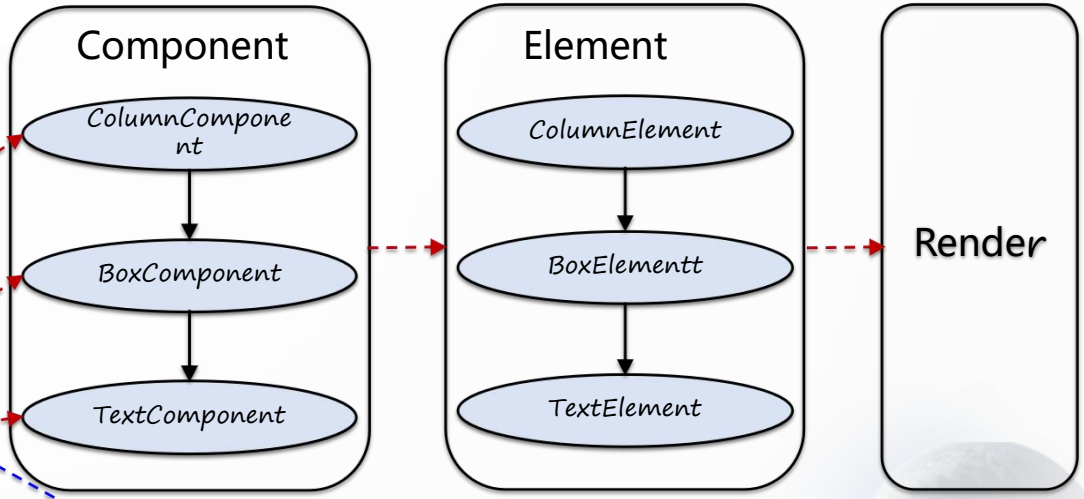
```
@Component struct CompA {
  @State aVar : number = 7;
  build() {
    Column() {
      Text("Hello ${this.aVar}")
        .width(100)
        .height(50)
    }
  }
}
```

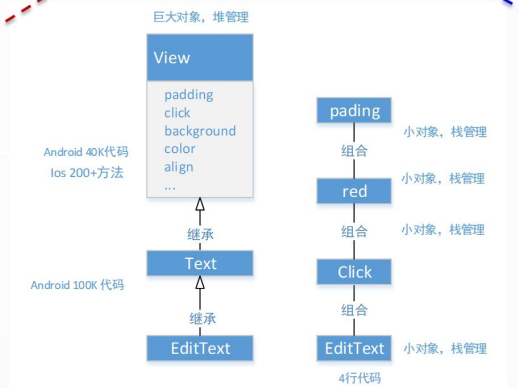
编译后代码:

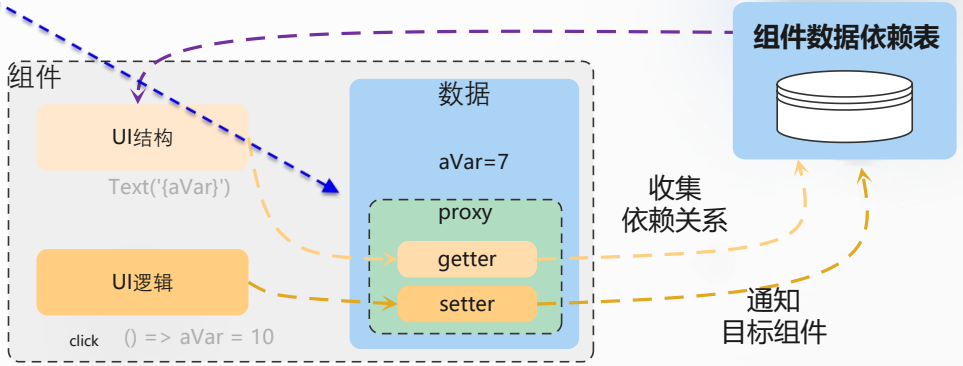
```
class CompA extends View {
  private ObservedPropertySimple<number> __aVar =
    new ObservedPropertySimple(/* default value
*/ 7,
      /* subscribe View for get + set access
notifications */ this);
  private get aVar() : number { return
this.__aVar.get(); }
  private set aVar(newValue : number)
{ this.__aVar.set(newValue); }

  private render() : void {
    Column.create(..);
        Text.create(..);
        Text.setLabel(`Hello ${this.aVar}`);
        Text.width(100);
        Text.height(50);
    Column.pop();
  }
}
```

扁平化管线
+按需渲染
+AOT
= 对标原生的体验

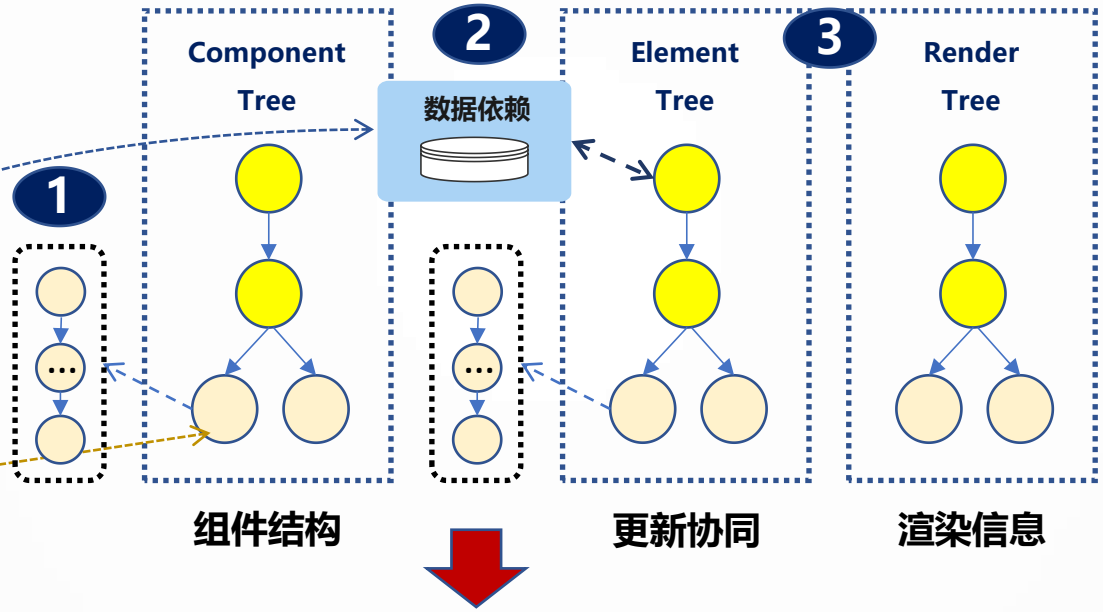


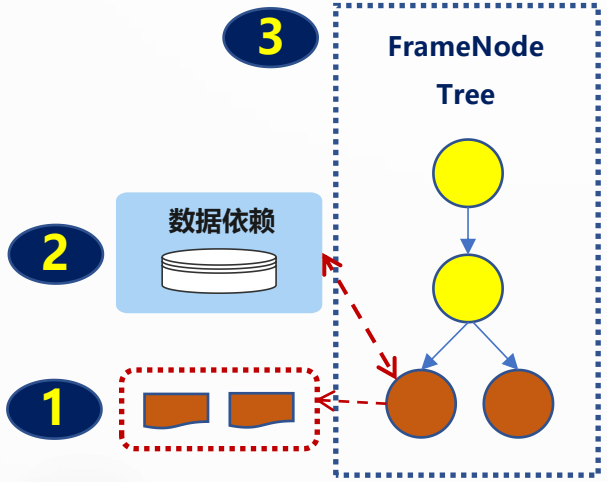




ArkUI渲染架构升级

```
@Component
struct Index {
  @State message: string = "Hi,ArkUI"
  build() {
    Column() {
      Text(this.message)
        .width(100)
        .height(50)
        .click(()=>{
          this.message = "Hi,HDC"
        })
      Image("Hello.png")
        .width(100)
        .height(50)
    }
  }
}
```





1. **多节点组合模型 → 单节点+属性组合模型**
   进一步扁平化 & 按需构建

2. **数据依赖组件级更新 → 细粒度函数级更新**
   进一步精细化更新范围

3. **三颗树 → 一颗树**
   进一步提升组件构建速度 & 降低内存

页面渲染速度提升20%，内存降低30%，指令数减少20%

单设备 → 跨设备

ArkUI – 跨设备UI适配

场景样例

典型场景样例

顶部

入口图标

分栏

瀑布流

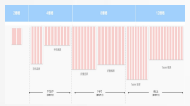
图文/图片详情等大图大字体典型场景

布局能力

响应式布局：响应式组件+响应式能力

列表　侧边栏　网格　媒体查询

栅格断点系统



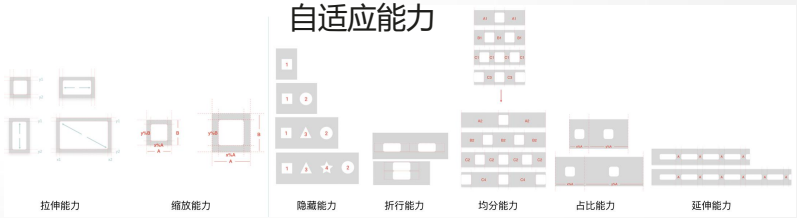
断点

类型	默认宽度	典型设备
XS	[0,320)	手表
SM	[320.600)	普通手机竖屏，折叠屏折叠态
MD	[600,840)	普通手机横屏，折叠屏展开态（包括横竖屏），平板竖屏
LG	[840,+∞)	平板横屏

自适应布局：自适应组件+自适应能力

Row/Column/Stack/Flex自适应布局组件

自适应能力

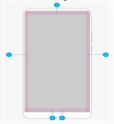


视觉交互

分层参数/主题风格

圆角尺寸

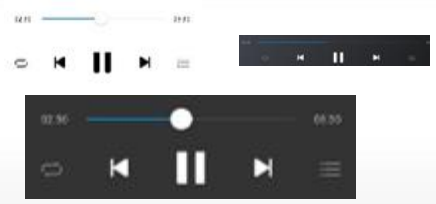
边框



字体

颜色

多态组件



交互归一

关键能力

① 多设备典型场景样例

典型布局：联合UX对常用组件的组合场景作出定义和输出Sample Code，包含：顶部、入口、分栏、瀑布、Banner、视频列表、视频宫格、图文、图片详情等）

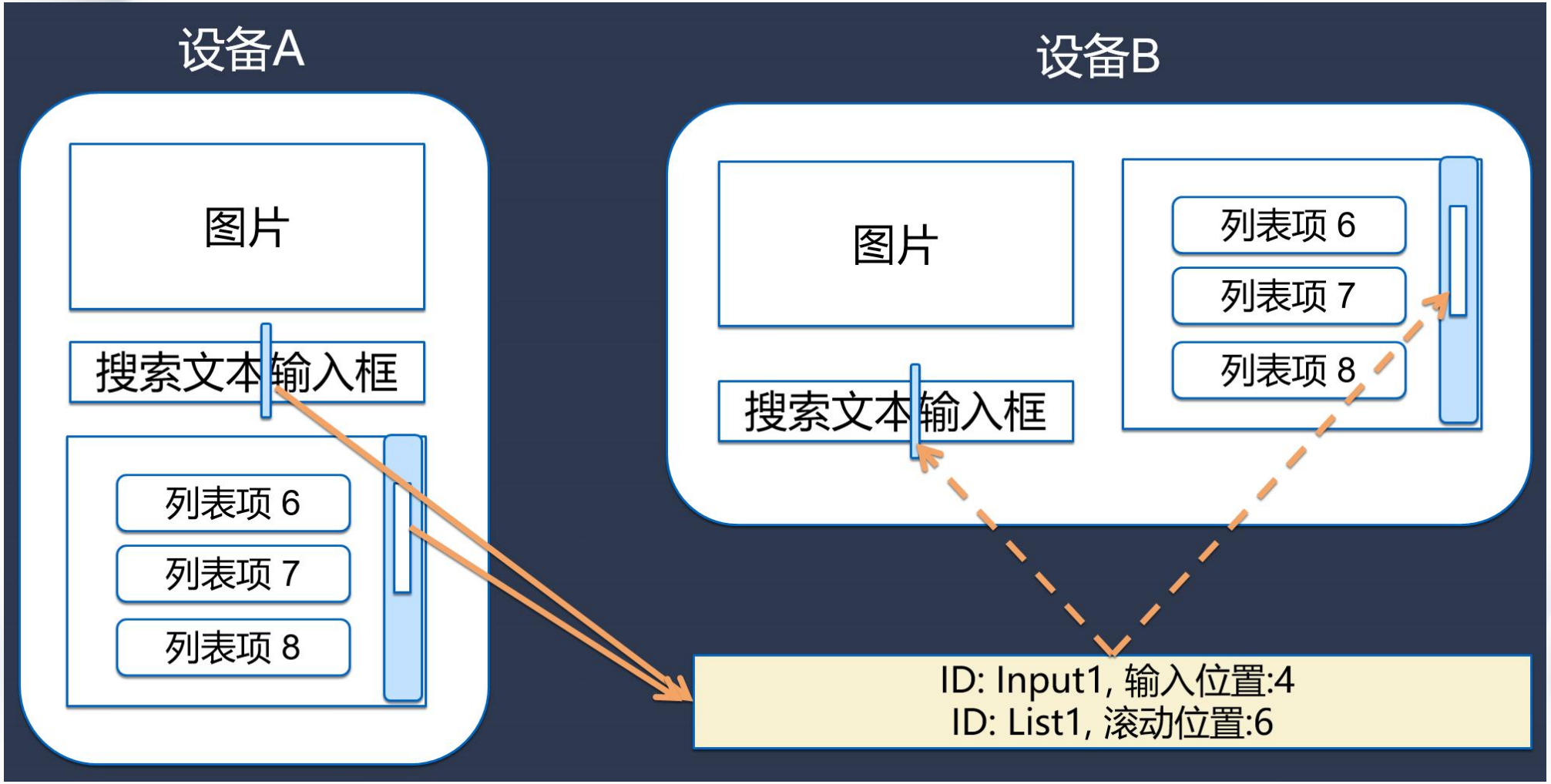
② 多设备响应式/自适应布局

- **响应式布局**：6个响应式组件（List 、Navigation、Grid、Swiper、Tabs、栅格布局组件）、3个响应式能力（栅格系统、自定义断点系统、媒体查询）
- **自适应布局**：4个常用自适应组件+7个自适应能力（隐藏、延伸、缩放、均分、占比、拉伸、折行）

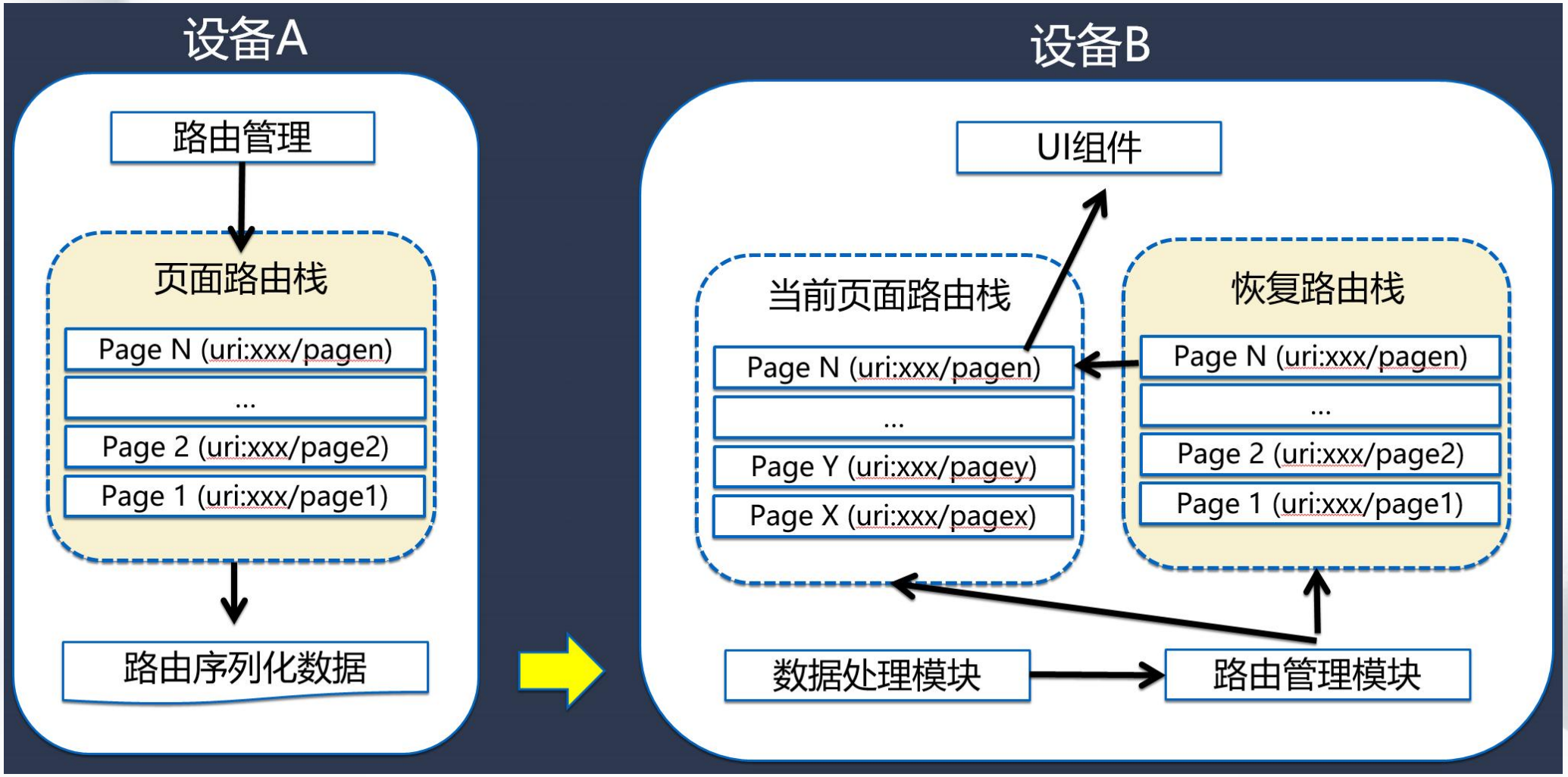
③ 多设备个性化视觉/交互归一

- **分层参数**：支持多设备上采用不同的主题风格，满足应用的个性化皮肤
- **页面主题**：深浅主题，自定义主题样式
- **交互归一**：屏蔽设备差异统一交互逻辑

ArkUI – 跨设备组件状态迁移

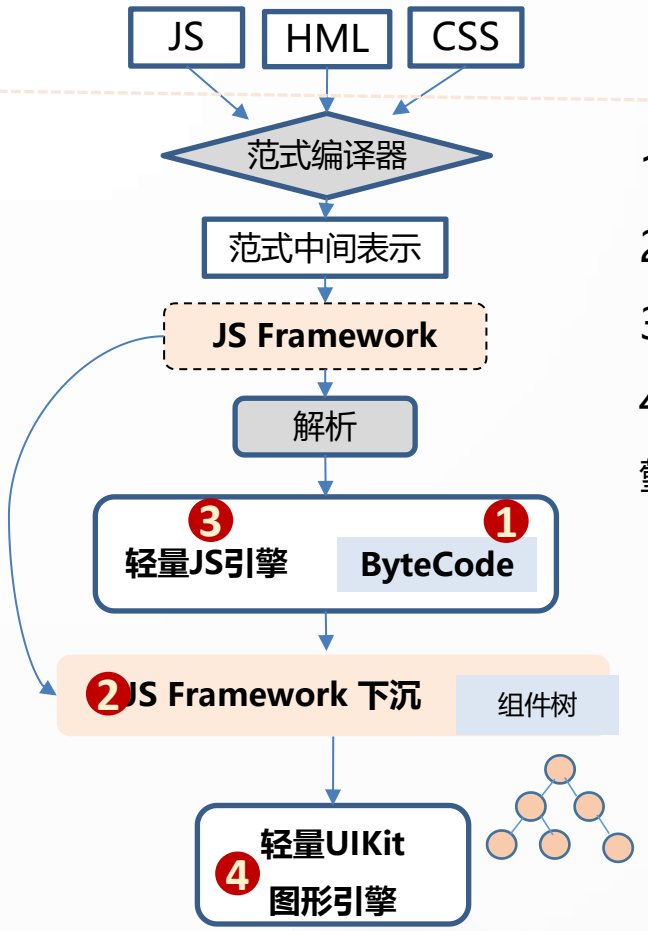


ArkUI – 跨设备路由栈迁移



ArkUI - 跨设备能力适配（延伸到百K级/M级设备）

轻设备



类Web范式

1. 预编译
2. JS框架下沉
3. 轻量JS引擎
4. 轻量UIKit/图形引擎

生态构建思考

生态构建概览

开发者

标准

社区

工具

框架

W3C

ECMA

软件绿色联盟 Software Green Alliance

W3C MiniApp

开放原子开源基金会 OPENATOM FOUNDATION

三方开源库

组件仓库

NPM

DevEco Studio ...

三方开发工具整合

VSCode

Chrome

关键自研应用使能

关键三方应用使能

电商

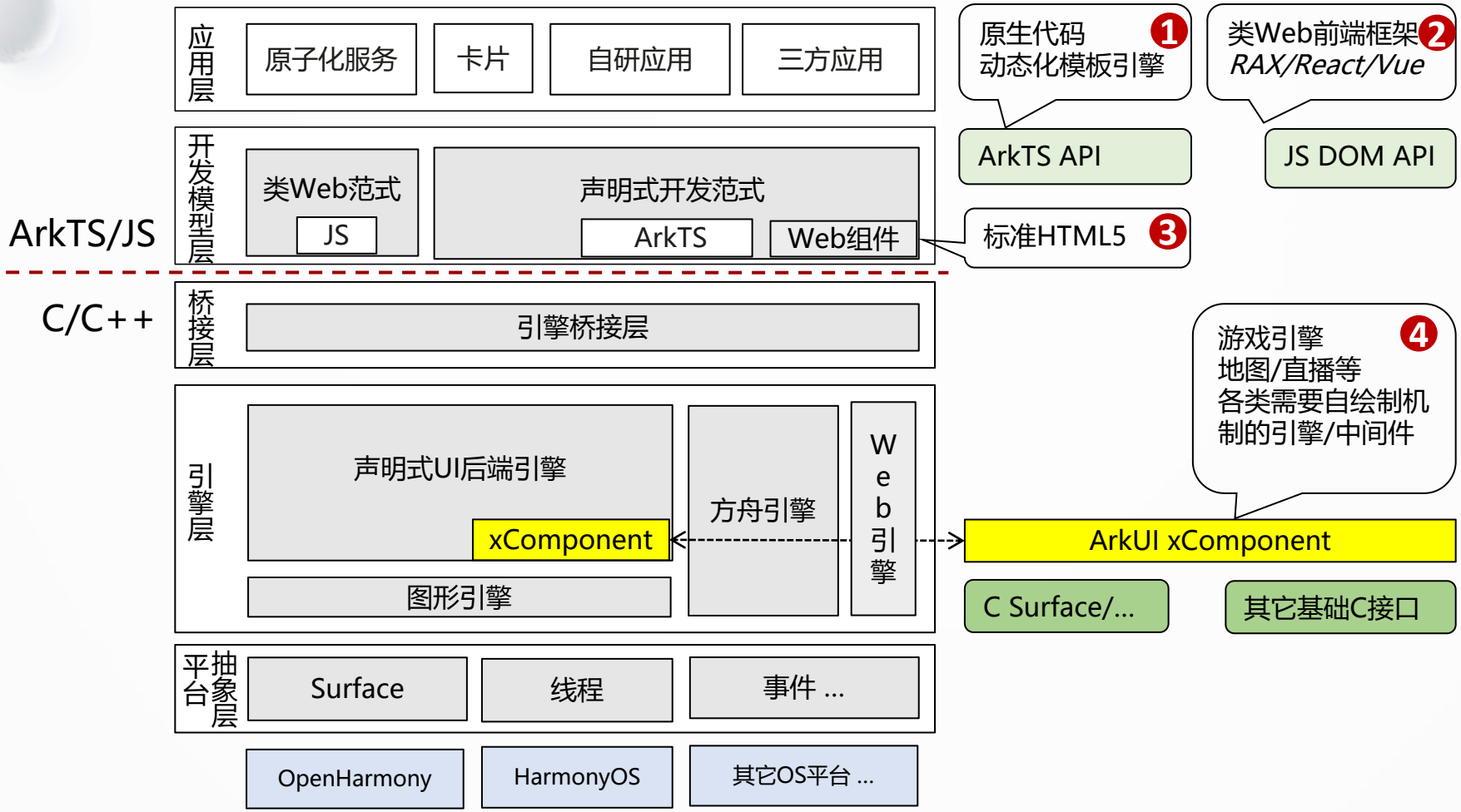
地图

游戏 ...

跨OS平台

以**关键应用**为牵引，逐步夯实相应能力构建，通过工具、社区协同；并布局标准培育长线影响力

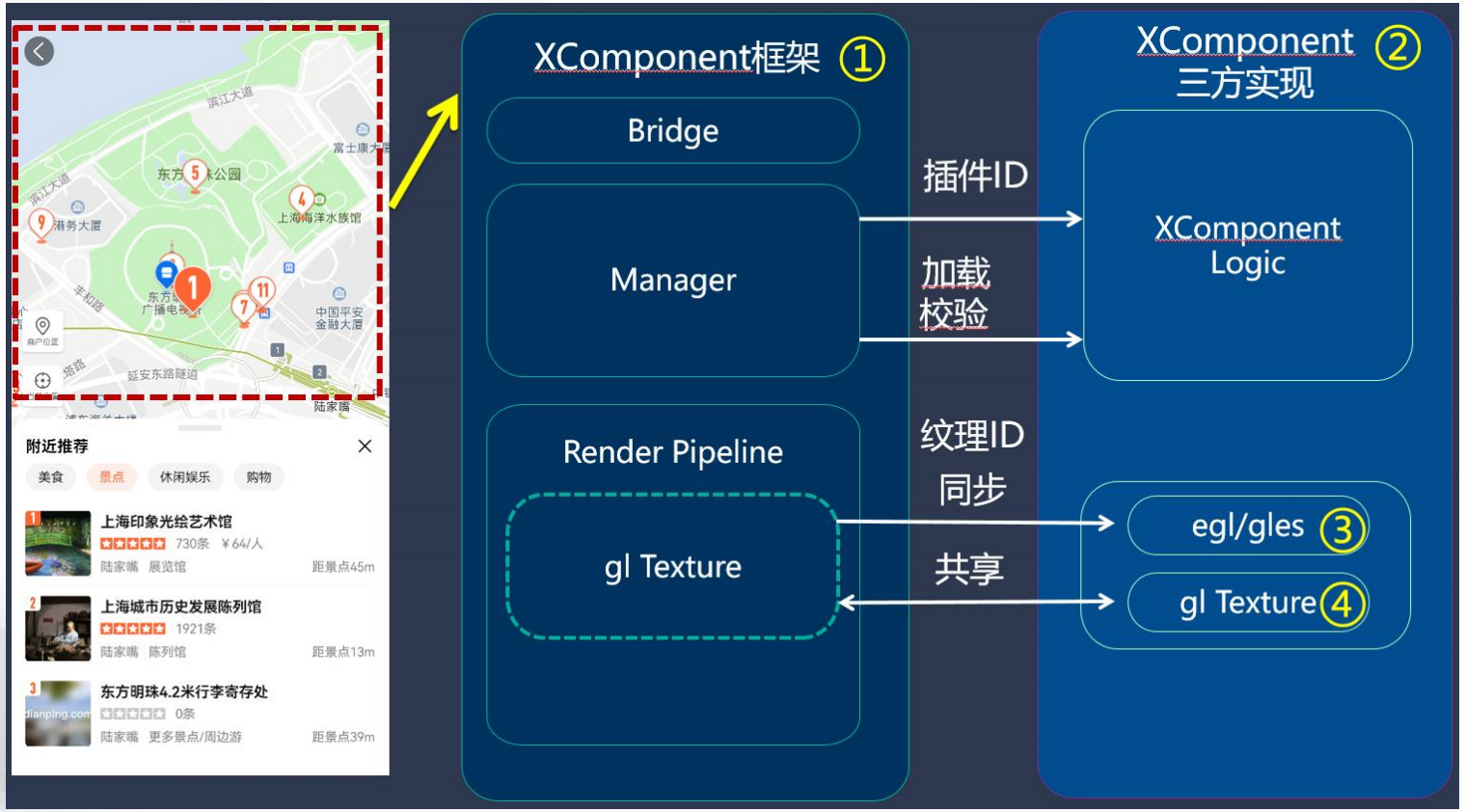
ArkUI生态分层对接概览



渐进式演进:

1. 存量业务生态通过桥接层适配到ArkUI，新的业务基于ArkUI
2. 基于ArkUI范式演进**动态内容部署能力，配合跨平台能力，逐步归一**

ArkUI生态扩展 – XComponent



三方自绘制引擎(地图/游戏等)接入:

① **低成本**
框架逻辑对开发者透明

② **独立管理**
插件可独立编译，发布

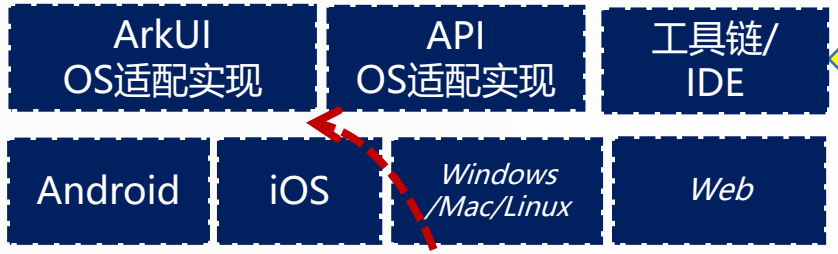
③ **egl/gles标准**
跨平台，独立线程

④ **共享纹理**
无冗余的数据拷贝，
插件和框架渲染解耦

ArkUI生态扩展 – 跨OS平台

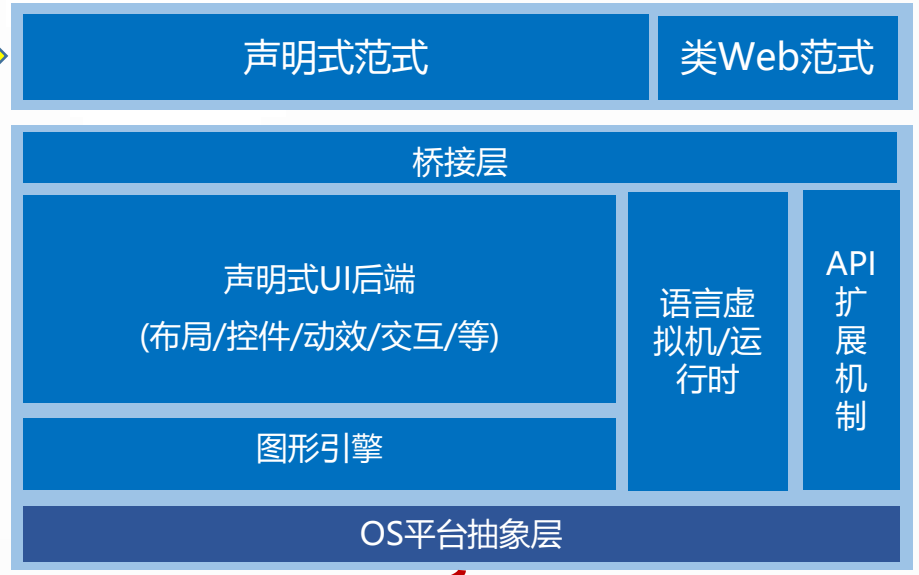
ArkUI-X开源项目社区

ArkUI OS适配/API OS适配/工具链



OpenHarmony开源项目社区

ArkUI/能力API定义和实现/...

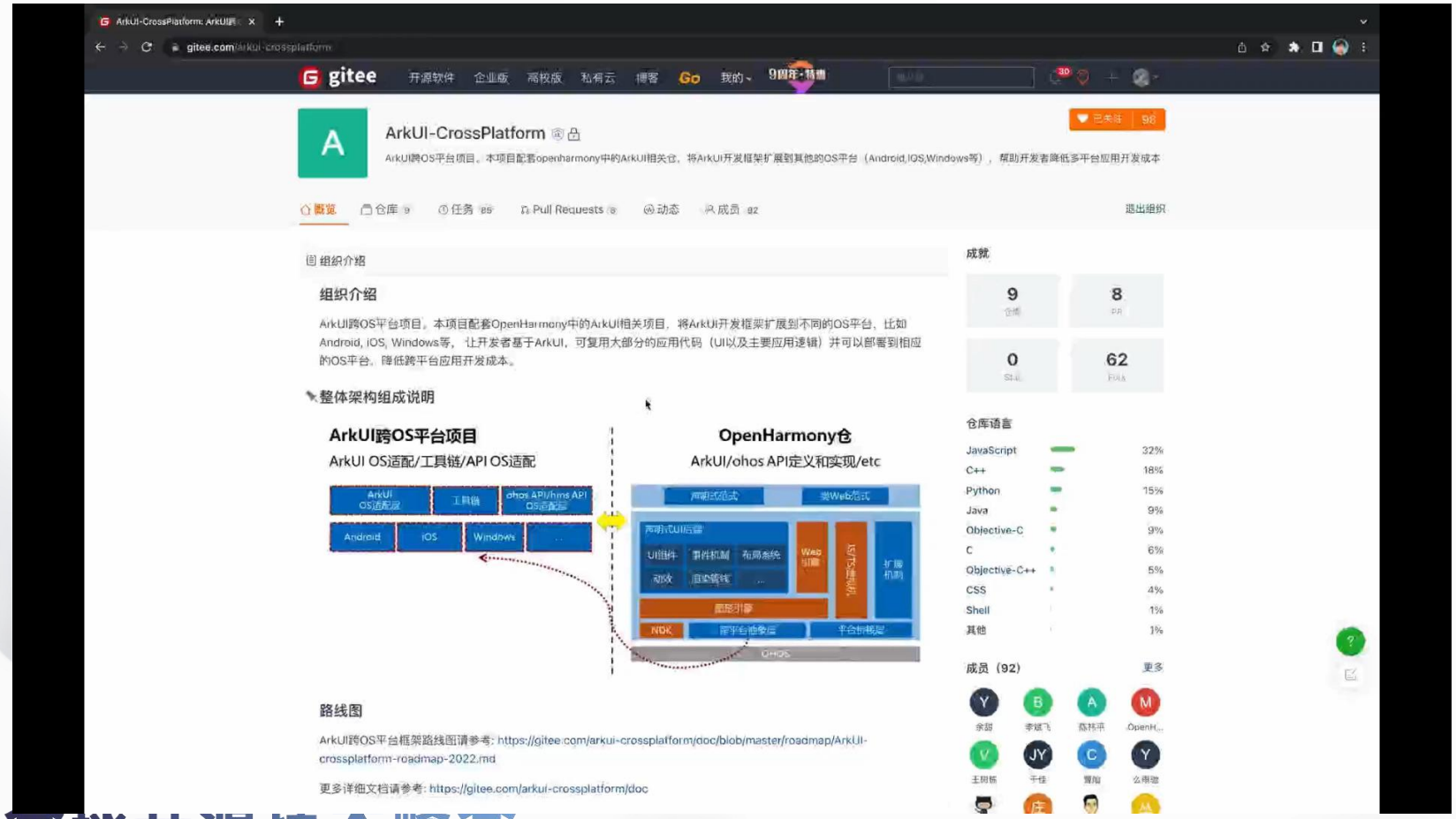


1. 简洁自然范式
2. 高效渲染管线
3. 基于类型的AOT加速机制
4. JS/TS生态以及统一的JS API

已有Android/iOS基础支持。 https://gitee.com/arkui-x （目前是定向开源）

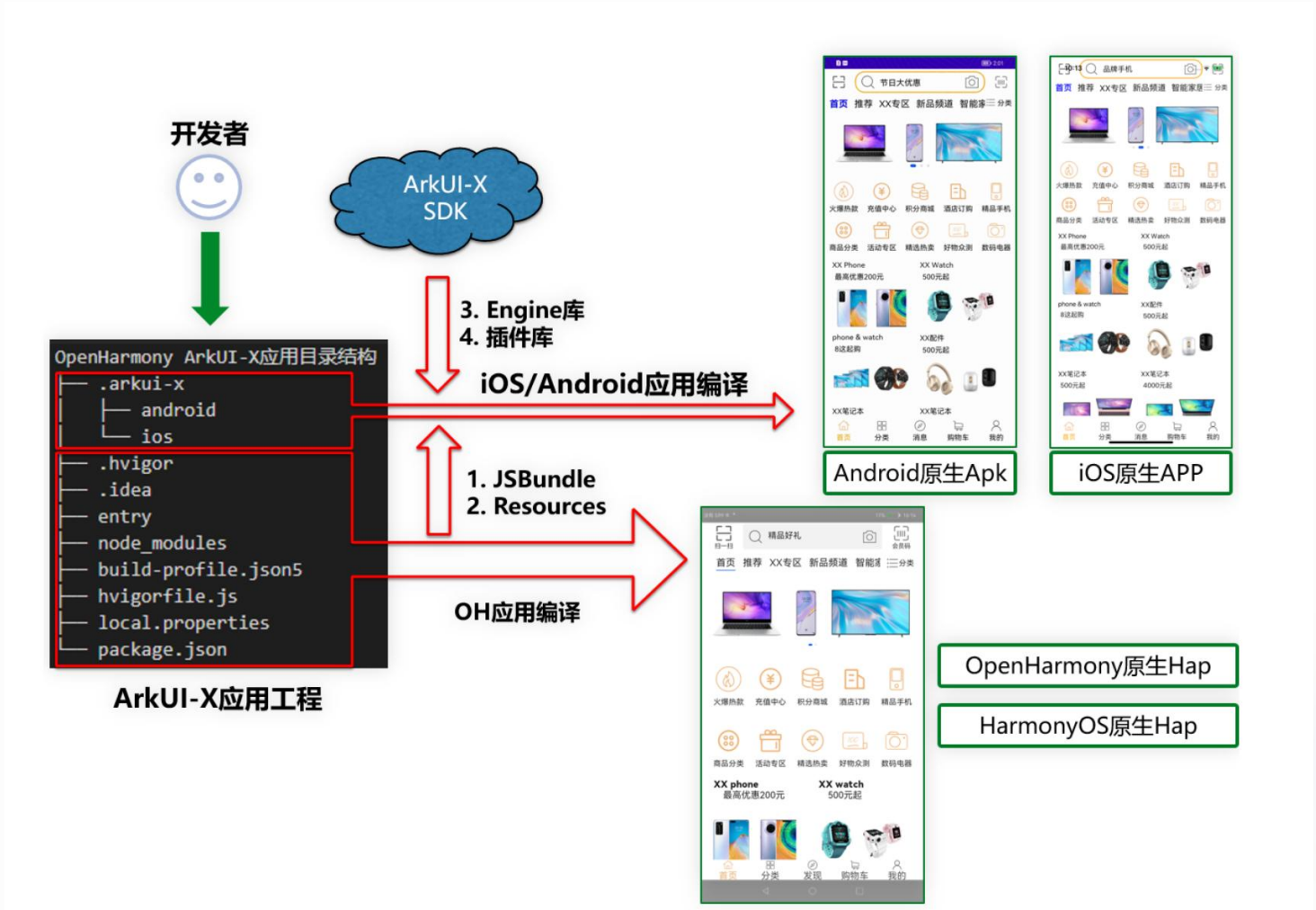
ArkUI-X跨平台项目基础演示

跨平台开发框架SDK和Sample工程构建演示
下载代码->构建SDK->创建工程->编译目标平台可执行文件



同一Sample工程
在iOS和Android上运行示例

开发部署示例


开发者
ArkUI-X
SDK
3. Engine库
4. 插件库
iOS/Android应用编译
1. JSBundle
2. Resources
OH应用编译
OpenHarmony ArkUI-X应用目录结构
.arkui-x
android
ios
.hvigor
.idea
entry
node_modules
build-profile.json5
hvigorfile.js
local.properties
package.json
ArkUI-X应用工程
Android原生Apk
iOS原生APP
OpenHarmony原生Hap
HarmonyOS原生Hap

小结

ArkUI设计 – 在TS基础上演进出可对标原生体验的声明式开发框架

- 语言运行时：声明式语法，类型加速，AOT
- 极简声明式范式；多维状态管理，一体化高效渲染管线
- 可伸缩的运行时，跨设备体验

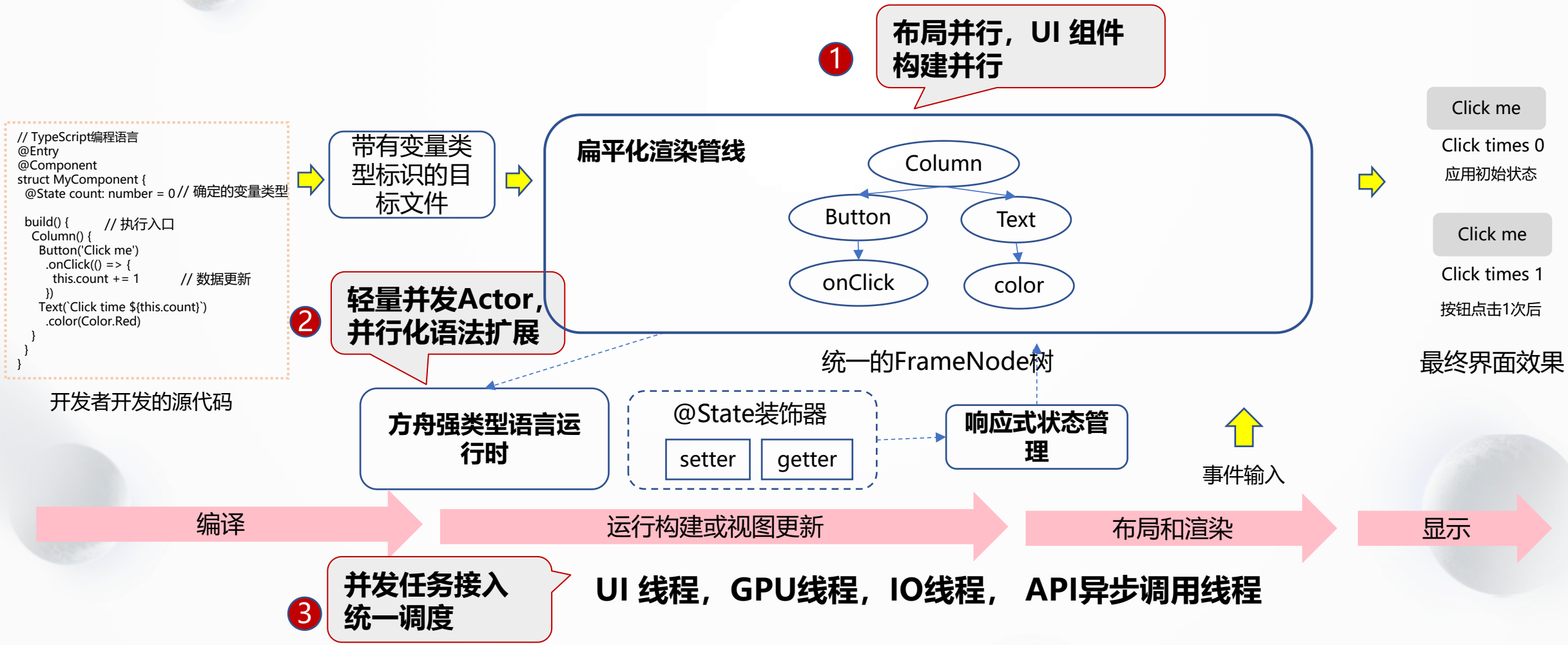
ArkUI生态

- 分层对接架构
- 跨OS平台能力

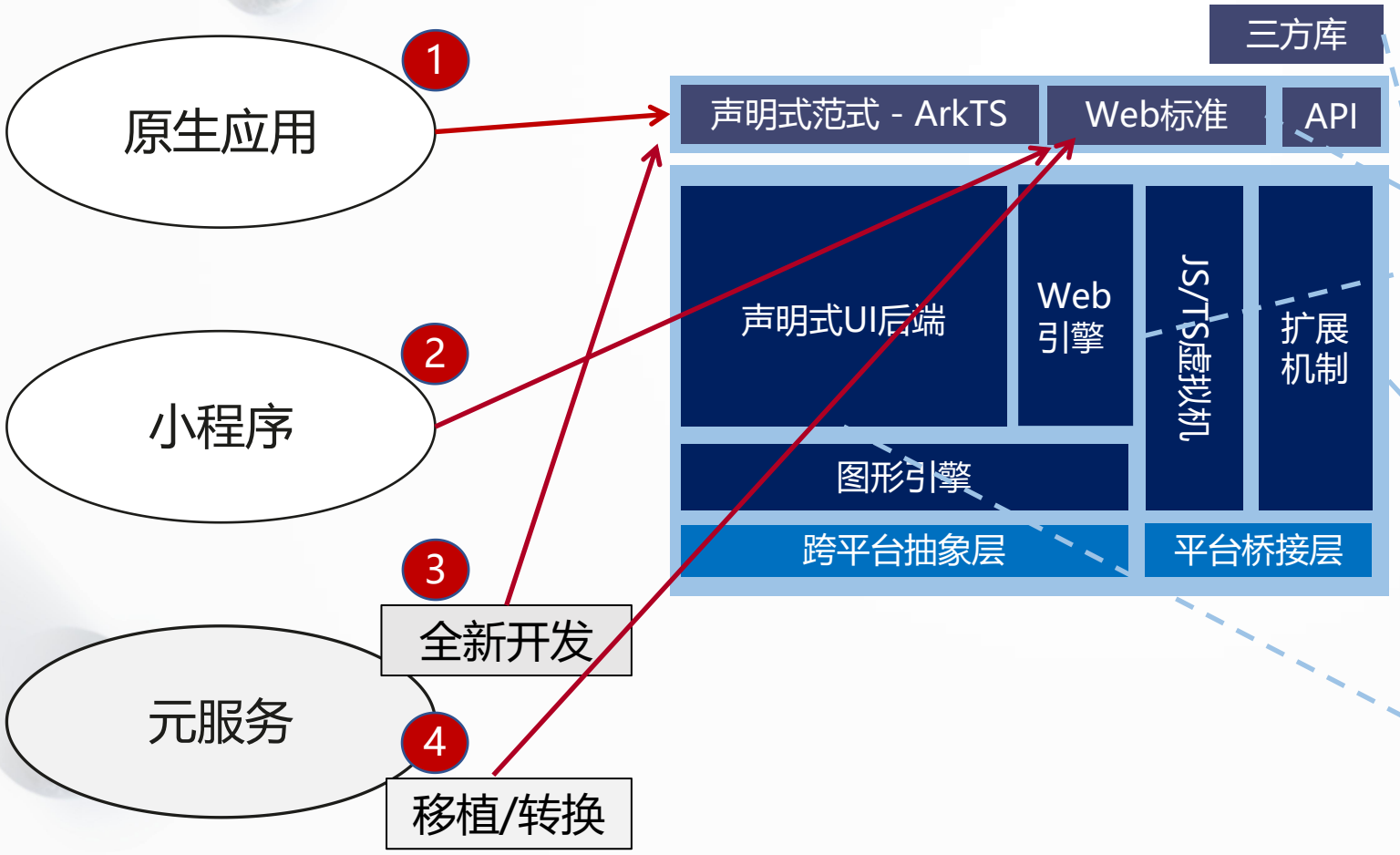
下一步，

持续围绕“竞争力” + “生态”演进

多维度并行和统一调度，进一步提升能效



应用生态演进视图思考



相关技术演进

Web标准子集
高效，轻量化，跨平台运行时
(对标原生体验)

高效跨平台API扩展机制
+
基于WASM能力扩展

并行化持续改造&提升
跨平台能力
...

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE