解密 HarmonyOS UI 框架

开发多设备场景下应用，开发者面临着设备形态差异带来的开发挑战：不同设备屏幕大小、屏幕分辨率以及屏幕形状不尽相同，由此让不同设备 UI 样式开发难度加大。同时，不同设备上交互模式不同也带来了交互维度的开发挑战。

面对设备形态差异带来的开发挑战，HarmonyOS 针对性地提出了两方面的解决策略，一方面是围绕 UI 维度，另一方面围绕交互维度。

针对终端形态差异挑战的解决策略

围绕 UI 维度的解决策略，主要包括以下方面：

多态控件

同样一个控件，在不同设备上有不同的形态以及交互模式，这就是多态。举个例子，一个开关按钮在不同设备上的 UI 效果诉求是不一样的。在手机一般采用滑动条的形式，大屏基本上是打勾打叉选择的形式，而不是滑动条。

然而大家会发现，虽然形态不同，但对描述来说，或者实现的结果，都是相同的。HarmonyOS 的多态控件可以做到对同一种按钮做一样的描述，但在不同设备上有不同呈现，有不同的体

验，同时设计能够直接贴合相应设备的设计规范。HarmonyOS 通过多态控件做”表述统一”从而帮助开发者进一步降低开发成本。

动态布局

我们都知道，单单解决控件问题是不够的，开发还会涉及布局的问题。布局问题如果再细分的话，又可分两个维度：内容框架和界面元素。

内容框架指的是布局的整个内容，比如说横屏或竖屏，内容呈现大致形式是什么样。

界面元素是指布局在内容框架中的元素，开发需要解决的问题是它们应该如何排列，能否进行扩展等问题。

HarmonyOS 通过动态布局，达到按需取用。本质上来说，就是针对不同分辨率可选不同的布局。同时，在分辨率变化的情况下，提供栅格化原子化布局能力，针对 UI 元素进行组合，如缩放，自动折行，自动隐藏等。通过这种基础能力实现更好的屏幕适配，帮助开发者在屏幕布局上做进一步增强。

围绕交互维度，HarmonyOS 引入交互事件归一的解决策略：

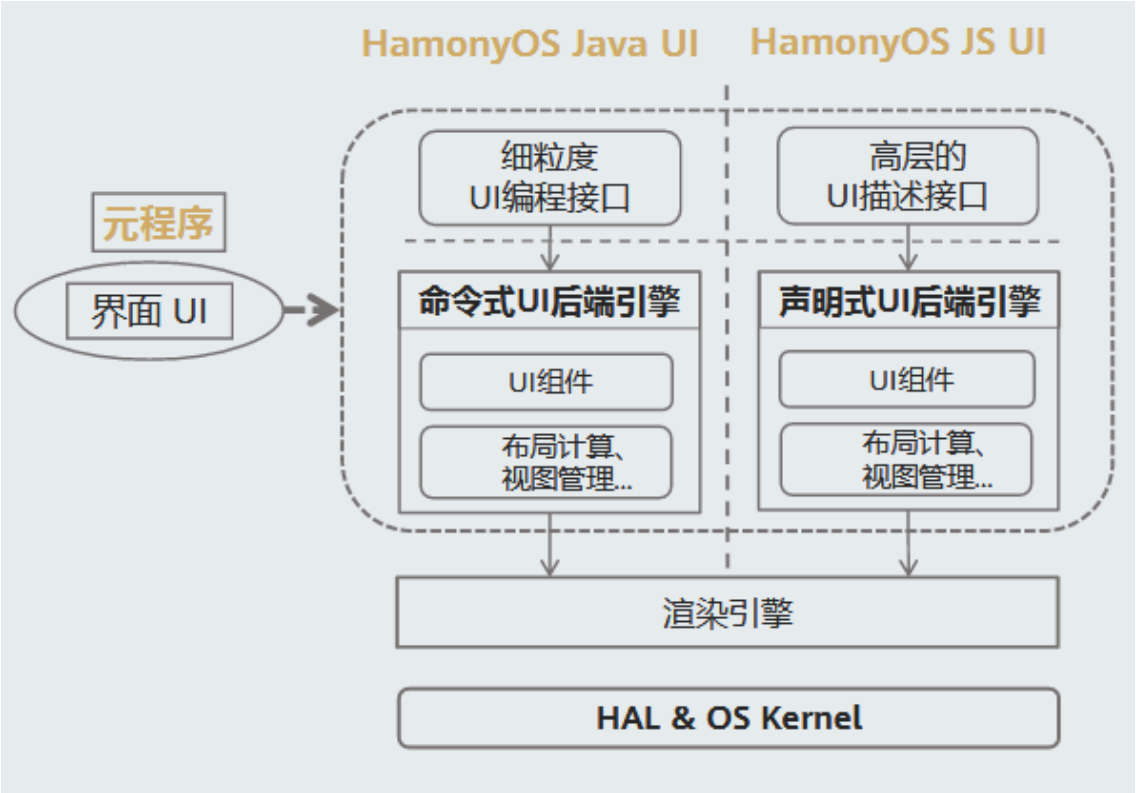
事件归一

虽然我们都知道在不同平台上交互模式是不一样的，但其想达到的效果却是一样的，在这样想法的驱使下，HarmonyOS 通过框架层，屏蔽输入差异，把不同的输入变成统一的事件，把交互事件的接口尽量一致化， 使得开发者在响应控件时，只需关心一致化的事件，达到更好的一致化响应交互行为。

以上是 HarmonyOS 针对设备形态差异挑战的解决策略。这些解决策略将被封装成 HarmonyOS UI 框架，让开发者可以通过调用相应接口直接开发，提高开发效率。

HarmonyOS UI 框架

HarmonyOS 的这一套 UI 框架，全称为 AbilityCross-platform Environment，ACE UI 框架，可支持主流的开发语言——Java 以及 JavaScript，分别对应命令式和声明式两种开发模式。



对于 Java UI 框架，命令式开发模式而言，相对来说都是细粒度的 API，完全由开发者控制。开发者操纵 UI 实现具体变更，通过调用 API 来实现整个 UI 编程的目标。这是一种由开发者来处理的较为常用的开发模式。

JavaUI 框架基本架构是上层细粒度 UI 编程接口，中间是命令式 UI 后端引擎（包含 UI 组件，布局计算，视图管理....），紧接着是跨平台渲染引擎基础设施。

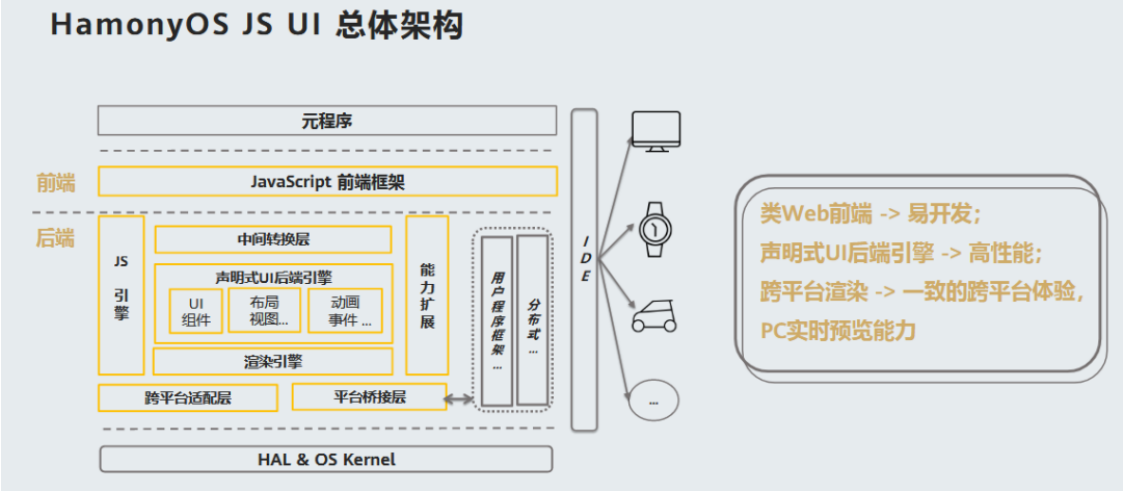
同时，HarmonyOS 也提供 JavaScript UI 框架，即 JS UI 框架，声明式模式，对 UI 描述是相对高层的，本质上声明式对 UI 操纵主要靠描述，对于开发者来说，只需描述即可，剩下的 UI 变更是通过数据驱动实现，这些变更的真正操作是通过框架层来处理。相当于开发者只要通过声明式描述好 UI，以及明确什么样的数据状态改变，涉及什么样的 UI 变更，其

余由框架层来具体实现。

JS UI 框架基本架构是上层为高层 UI 描述接口，中间是声明式 UI 后端引擎，包含 UI 组件，布局计算，视图管理....，紧接着是跨平台渲染引擎基础设施。 下面让我们展开看一下 JS UI 和 Java UI 的框架到底是怎么样的。

JS UI 框架

总体架构



从上面的图可以看到，JS UI 总体架构大的维度主要分为前端和后端。

一、前端

前端主要是 JavaScript 的前端框架。这里采用的是类 web 的前端开发模式，开发相应的 UI。

二、后端

后端主要分为以下几个部分：

UI 引擎部分

即呈现的构建部分，这当中包含 JS 执行引擎本身， 由 C++构建的声明式 UI 后端引擎（包括 UI 组件实现、布局视图、动画事件）和渲染引擎，这些共同构建了整个 UI 的呈现。

中间转换层

通过中间转化层，把前面 JS 的 UI 描述，转化成声明式 UI，让后端引擎去执行。

能力扩展

HarmonyOS 提供“扩展 API“包括各种各样的分布式能力、系统的基础能力。通过能力扩展基础设施，开发者可以调用 JavaScript API 来实现更丰富的逻辑功能。

跨平台的适配层

从设计上来说，HarmonyOS 可以实现跨设备跨 OS。主要是因为整个渲染路径是由后端完全自己绘制的，通过一个底层画布来实现 UI 功能，这样对 OS 的依赖相对较少，能达到跨平台的效果。

平台桥接层

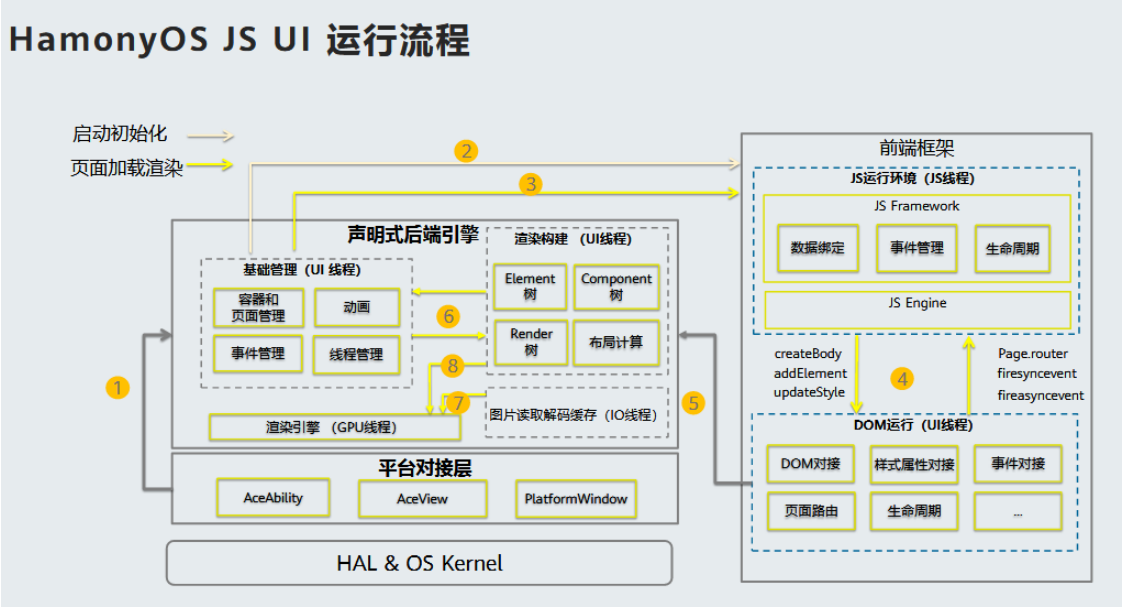
从架构设计上来说，我们是可以运行在 HarmonyOS 上，其实也可以在其他 OS 上运行。

这里前提条件是 HarmonyOS 会与其他 OS 内的基础设施以及基础能力做桥接，从而达到能力扩展的目标。

当然同时我们离不开工具的帮助，通过 IDE 包括其他相应一些工具链就实现通过 JS 开发后，部署到不同的设备上的目标。

总的来说，通过声明式 UI 框架，借用类 Web 前端的简易开发能力，同时结合后端引擎的高性能能力，HarmonyOS 帮助开发者达到易开发，高性能的目标。通过 HarmonyOS 的跨平台渲染基础设施，能够达到相对比较一致的跨平台的体验。如电视或手表上的预览路径、渲染路径基本保持一致，达到一个较好的实时预览效果，从而得到较一致的渲染体验。

运行流程



通过上述流程图我们可以看到，当一个应用启动时，最早是从 Ability（HarmonyOS 运行的最基础单元）出发，Ability 内有 UI 的框架部分。前端框架的整体职责就是加载解析和运

行 JS 应用，并完成前端开发范式的组件树到声明式后端渲染框架层 Component 树的复杂对接。后端渲染框架是实现整个渲染流水线管理的核心部分，维护了三棵渲染相关的树：Component 树、Element 树和 Render 树，一些耗时的 IO 操作，例如图片相关的获取和加载放在了单独的 IO 线程，这些都纳入到了容器的统一管理之中，配合动画、事件等，完成 UI 线程的绘制，最终由渲染引擎负责光栅化以及合成上屏，构建了高效的渲染流水线。

这当中完全是多线程设计的，也就是说前端部分，JS 线程，UI 线程，以及 IO 线程都可以并行化处理，从而达到较好的执行效率，以及较高的性能，这是一个大致 ACE JS 的运行流程。

应用示例

我们来看具体一个例子——一个音乐应用，可以在手表和大屏上同时运行。

代码组成分为三个部分——模板、样式及业务逻辑。

- 模板：基本描述了整个 UI 结构。

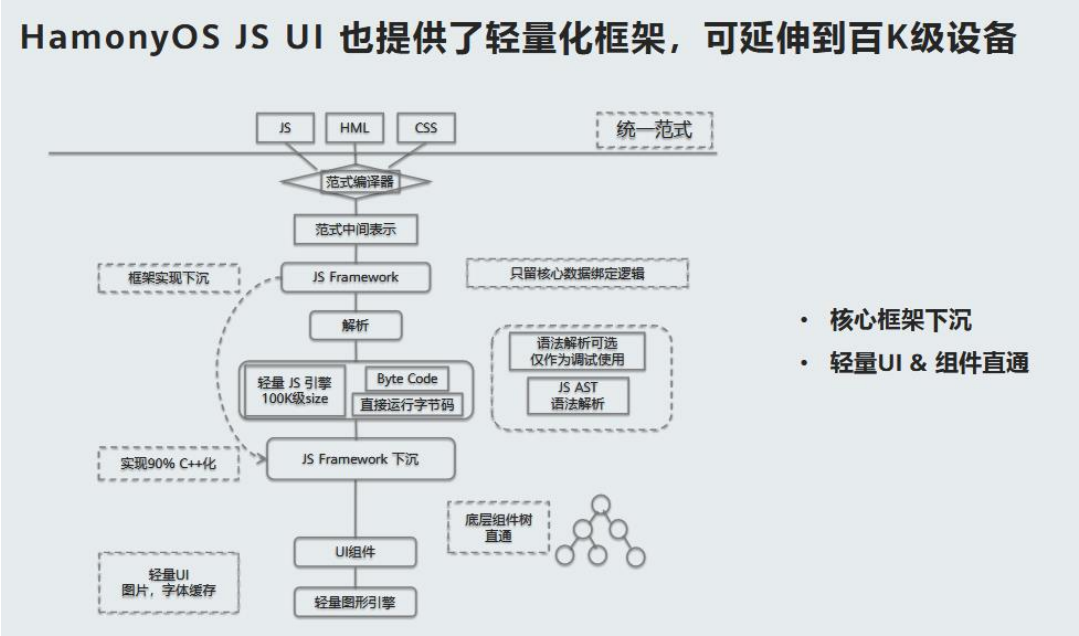
来源：HarmonyOS 微信号 https://mp.weixin.qq.com/s/0RZL09vKppIZmpqTJUeRSA

- 样式：对整个 UI 结构的呈现模式进行描述。
- 业务逻辑：具备数据绑定的声明，以此获取数据之间的关联。扩展能力声明，可调用系统里面各种各样的 Ability 来做功能扩展。

在多 UI 自适应上面，HarmonyOS 支持业界主流机制，做到动态布局以及 UI 自适应能力。

JS UI 为百 K 级设备提供轻量化框架

上述说提到的都是关于富设备的设备 UI 架构。所谓的富设备，就是内存相对比较充足的，至少有 512Mb 以上的。那对于轻设备，内存较小的设备来说，HarmonyOS 提供轻量化框架。



在轻量化框架中，HarmonyOS 把一些核心框架做下沉，C++化及提供轻量级的 JS 引擎，包括 UI 组件，达到非常轻量的目标，通过统一的开发范式，可以在百 K 资源上运行起来。

来源：HarmonyOS 微信号 https://mp.weixin.qq.com/s/0RZL09vKpplZmpqTJUeRSA

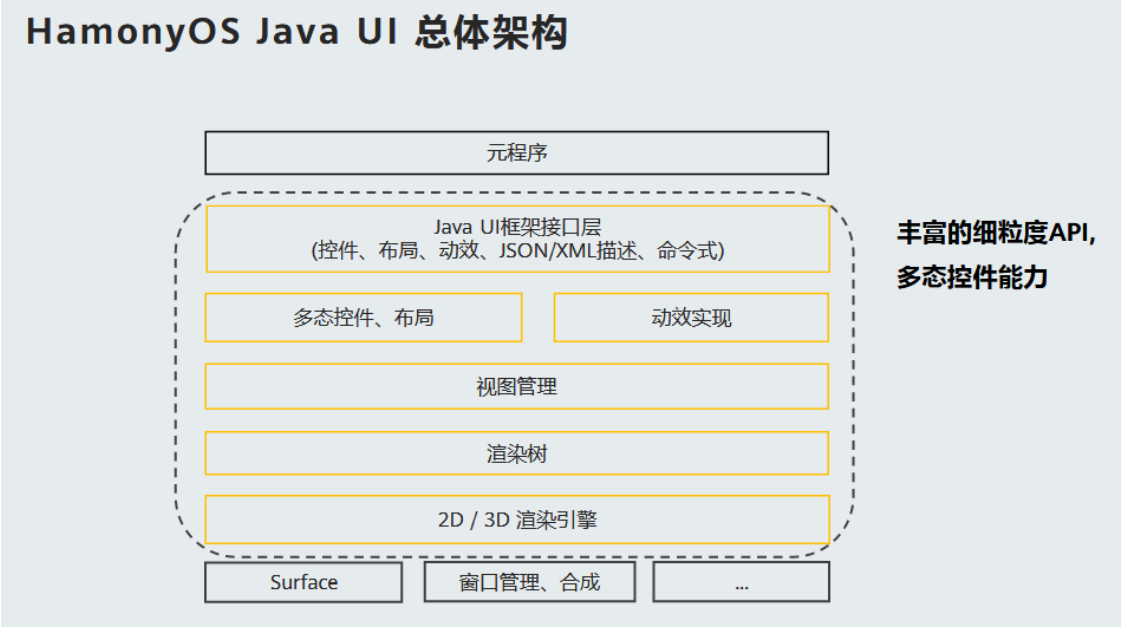
但由于是在轻量化的框架设备上，能力还是有限的，有些资源的限制可能 API 没办法提供，但对公共部分的 API 是完全共通的。

Java UI 框架

与 JS 的声明式不同，Java 更多是面向命令式开发模式，从 API 维度而言会更加丰富更加细粒度一点。也拥有多态控件，布局等能力。

总体架构

从逻辑上来说，它的整个架构分为五个部分。



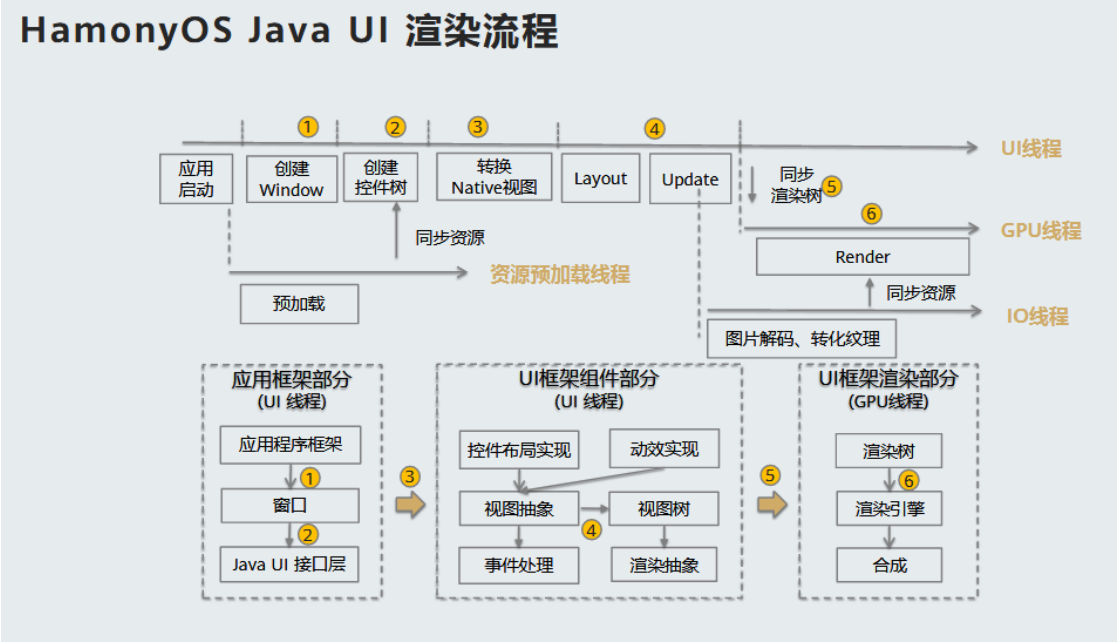
- JavaUI 框架接口层：包括控件、布局、动效和描述。
- 多态控件、布局和动效实现：核心的 C++层，有相应的多态控件布局，各种各样动效

实现。

- 视图管理：生成 UI 控件对应的 View 对象，管理 View 对象的生命周期（更新，挂载，卸载，删除),维护 View 对象组成的 UI hierarchies 关系。
- 渲染树:维护 View 对象对应的 Render Node，维护 UI 组件变更引起的渲染树的变更，生成 Render Node 对应 Draw Command。
- 2D/3D 渲染引擎：执行 Draw Command，生成 UI 控件所包含的线条，面，文本对象。最终到系统的合成，总体构成了整个 ACE Java 架构。

渲染流程

这里简要描述整个渲染流程。

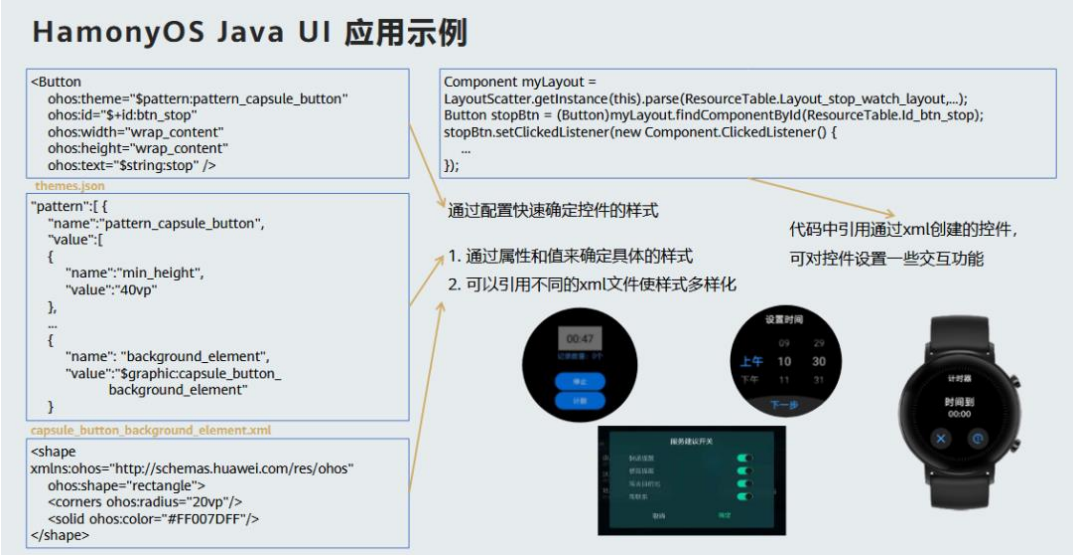


JavaUI 渲染采用多线程设计，分为几个阶段：在应用框架部分，创建窗口，创建相应的 Java UI 接口。在资源加载之后，由相应的 UI 框架组件部分转化成 Native 视图，Native 视图包

括整个控件布局实现、动效实现、视图抽象、事件处理、渲染抽象等，同步 UI 线程，渲染树会交给 UI 框架渲染部分（GPU 线程）来做相应的渲染以及最终的合成，这就是整个 ACE Java 的渲染过程。

应用示例

我们来看一个 Java 例子，其实跟传统的或说常用的 Java 开发模式类似。



通过 XML 描述整个布局，这当中少不了多态控件的支撑。通过属性和值来制定具体 XML 指定不同样式，以此创建组件。创建完成后，开发者可以设置各种各样的交互、相应的事件处理及后续 UI 变更。当然，HarmonyOS 提供相应的 API 为后续变更做支撑。

当然多设备场景开发还是有很多挑战，需要我们持续不断地增强和提升。当然也希望众多的开发者一起加入进来，与我们一起围绕着跨设备开发相关内容，多一些碰撞，共同把多设备开发体验逐步增强。

来源：HarmonyOS 微信号 https://mp.weixin.qq.com/s/0RZL09vKppIZmpqTJUeRSA