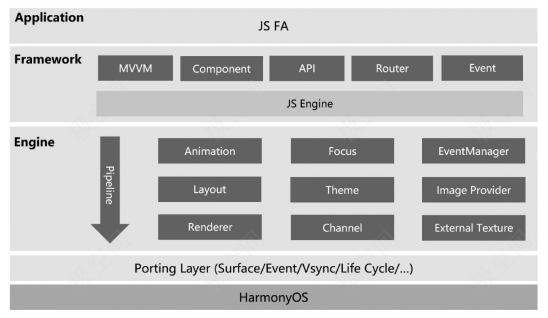
* JS UI框架是一种跨设备的高性能UI开发框架，支持声明式编程和跨设备多态UI
* JS UI框架提供了相对高层的UI描述，使应用开发更加简单
* JS UI框架支持纯JS、JS和Java混合语言开发
* JS FA指基于JS或JS和Java混合开发的FA
* JS UI的基础能力主要体现在3个方面：
  + 声明式编程
    - JS UI 框架采用HML（类 HTML，但非HTML） 和 CSS 声明式编程语言作为页面布局和页面样式的开发语言，页面业务逻辑则支持 ECMAScript 规范的 JavaScript 语言
    - JS UI 框架提供的声明式编程，可以让开发者避免编写 UI 状态切换的代码，视图配置信息更加直观
  + 跨设备
    - 开发框架架构上支持UI跨设备显示能力，运行时自动映射到不同设备类型，开发者无感知，从而降低开发者多设备适配成本
  + 高性能
    - 开发框架包含了许多核心的控件，如列表、图片和各类容器组件等，针对声明式语法进行了渲染流程的优化
* JS UI的架构图如下，包括应用层、前端框架层、引擎层和平台适配层
  + Application-应用层
    - 应用层表示开发者使用JS UI框架开发的FA应用，这里的FA应用特指JS FA应用
    - JS UI 框架支持纯 JavaScript、JavaScript 和 Java 混合语言开发。JS FA 指基于 JavaScript或 JavaScript 和 Java 混合开发的 FA
  + Framework-前端框架层
    - 前端框架层主要完成前端页面解析，以及提供MVVM（Mode-lView-ViewModel）开发模式、页面路由机制和自定义组件等能力
  + Engine-引擎层
    - 引擎层主要提供动画解析、DOM树构建、布局计算、渲染命令构建与绘制、事件管理等能力
  + Porting Layer-平台适配层
    - 适配层主要完成对平台层进行抽象，提供抽象接口，可以对接到系统平台。例如：事件对接、渲染管线对接和系统生命周期对接等



* AceAbility：AceAbility 类是 JS FA 在 HarmonyOS 上运行环境的基类，继承自 Ability。开发者的应用运行入口类应该从该类派生
* JS FA 生命周期事件分为应用生命周期和页面生命周期
  + 应用生命周期：应用通过 AceAbility 类中setInstanceName()接口设置该 Ability 的实例资源，并通过 AceAbility 窗口进行显示以及全局应用生命周期管理
* setInstanceName(String name)的参数 name 指实例名称，实例名与 config.json 文件中 module.js.name 的值对应。若开发者未修改实例名，而使用了缺省值 default，则无需调用此接口。若开发者修改了实例名，则需在应用 Ability 实例的 onStart()中调用此接口，并将参数 name 设置为修改后的实例名称（多实例应用的 module.js 字段中有多个实例项，使用时请选择相应的实例名称）
* setInstanceName()接口使用方法：在 MainAbility 的 onStart()中的 super.onStart()前调用此接口。以 JSComponentName 作为实例名称

# JS FA调用PA

* JS UI框架提供了JS FA调用Java PA的机制。该机制提供了一种通道来传递方法调用、处理数据返回以及订阅事件上报。FA提供了以下三个JS接口：
  + FeatureAbility.callAbility(OBJECT)：调用PA能力
  + FeatureAbility.subscribeAbilityEvent(OBJECT,Function)：订阅PA能力
  + FeatureAbility.unsubscribeAbilityEvent(OBJECT)：取消订阅PA能力
* 在PA端提供的接口为（当前提供的两种调用PA的方式）：
  + Ability 调用方式，FA 使用远端进程通信拉起并请求 PA 服务：
    - 拥有独立的 Ability 生命周期，适用于基本服务供多 FA 调用或者服务在后台独立运行的场景
    - IRemoteObject.onRemoteRequest(int, MessageParcel, MessageParcel, MessageOption)
  + Internal Ability 调用方式，采用内部函数调用的方式和 FA 进行通信：
    - AceInternalAbility.AceInternalAbilityHandler.onRemoteRequest(int, MessageParcel, MessageParcel, MessageOption)
    - 与 FA 共进程，适用于对服务响应时延要求较高的场景。该方式下 PA 不支持其他 FA 访问调用
* JS 端与 Java 端通过 bundleName 和 abilityName 来进行关联
  + 在系统收到 JS 调用请求后，根据开发者在 JS 接口中设置的参数来选择对应的处理方式
  + 开发者在onRemoteRequest()中实现 PA 提供的业务逻辑

# 组件

* 组件的通用属性有：常规属性和渲染属性
  + 常规属性：指的是组件普遍支持的用来设置组件基本标识和外观小时特征的属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 必填 | 描述 |
| id | string | - | 否 | 组件的唯一标识 |
| style | 组件的样式声明 |
| class | 组件的样式类，用于引用样式表 |
| ref | 用来指定指向子元素或子组件的引用信息，该引用将注册到父组件的$refs属性对象上 |
| disabled | boolean | false | 当前组件是否被禁用，在禁用场景下，组件将无法响应用户交互 |
| focusable | 当前组件是否可以获取焦点  当focusabled设置为true时，组件可以响应焦点事件和按键事件  当组件额外设置了按键事件或者点击事件时，框架会设置该属性为true |
| data-\* | string | - | 给当前组件设置data-\*属性，进行相应的数据存储和读取  大小写不敏感，如data-A和data-a默认相同 |
| click-effect | 通过这个属性可以设置组件的弹性点击效果 |
| dir | auto | 设置元素布局模式，支持设置rtl、ltr和auto三种属性值 |

* + 渲染属性：指的是组件普遍支持的用来设置组件是否渲染的属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| for | Array | 根据设置的数据列表，展开当前元素 |
| if | boolean | 根据设置的boolean值，显示或隐藏当前元素 |
| show |

* 属性和样式不能混用，不能在属性字段中进行样式设置