# **小程序iOS客户端框架——控件事件逻辑框架与控件原生化**

作者：[eldwinwang](http://gad.qq.com/user/index?id=386)

链接：<http://gad.qq.com/article/detail/43666>

导语 小程序自发布以来，为开发者和用户提供了一种轻量级的App。作为一种不需要下载安装即可使用的应用，它实现了应用“触手可及”的梦想，用户扫一扫或者搜一下即可打开应用。小程序也体现了“用完即走”的理念，用户不用关心是否安装太多应用的问题。 微信客户端为小程序的运行提供了框架支持，如service运行环境、页面缓存机制以及控件原生化支持等，本文将对这些部分实现原理做一一介绍。

**1.**   **内容概要**

微信小程序采用了传统的移动端H5浏览器作为页面运行环境，但是与传统的B/S结构的WEB应用不同，小程序为用户提供了普通H5页面无法达到、近似原生App的控件体验，同时也向开发者提供了功能丰富的API。本文将从小程序运行运行环境及框架开始，详细介绍iOS微信客户端对小程序控件层的框架支撑：用户的开发代码如何与用户界面交互、API的功能分类和设计，另外会简单介绍小程序的页面缓存机制。

另外，对于某些H5无法实现，或实现性能较差的控件，微信小程序采用了“控件原生化”方式，将客户端实现的原生控件提供给开发者使用，本文将对原生控件的设计和体验优化做详细的介绍。

**2.**   **小程序运行环境及框架简介**

为了对小程序的运行机制展开讨论，我们将从一个简单的小程序按钮开始，对小程序的事件处理流程作一个简单的了解。

在不同操作系统平台做应用开发时，通常开发工具都会以XML语言来描述应用的界面布局，如iOS采用storyboard文件，安卓使用了layout文件。在小程序中， 自定义了wxml文件来描述界面布局。以下是一个简单的界面文件示例，展示一个普通的按钮，并绑定了点击事件：



图1. 只有一个按钮的小程序界面布局

一个小程序界面除了必须的wxml来描述界面布局外，还可以提供wxss文件作为样式描述（可选）。另外，还需要编写这个页面对应的js文件，开发者的开发代码逻辑都在这个js文件中完成，在该js中处理用户事件、控制对应的界面的变化等等。下面是对图1的界面逻辑进行处理的js文件示例，脚本响应按钮的点击事件，并输出日志信息：

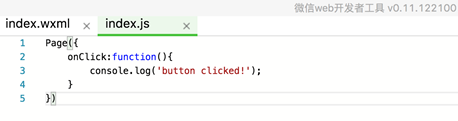


图2. js脚本中响应处理按钮事件

微信客户端通过 WKWebView以及JavaScriptCore提供了小程序的运行环境。WKWebView负责对wxml和wxss进行解析执行，并渲染展示；JavaScriptCore提供了开发者所写的逻辑代码（JavasScript）的运行环境，该运行环境我们称之为Service，Service中的代码与WebView中的代码完全隔离，如图3所示。

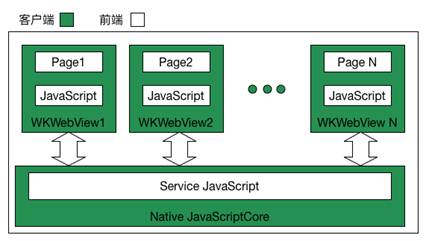


图3. 小程序运行环境框架

上图中，绿色部分为客户端提供的支持框架，白色部分为前端逻辑。如图所示，一个小程序就对应了一个Service，客户端通过JavaScriptCore为开发者的Service代码提供了运行环境；一个小程序可能有一个或多个Page作为向用户展示内容的交互界面，客户端由WKWebView提供了Page解析和渲染支持；页面与页面之间的通信通过Service环境中转。

用户点击页面中的Button控件后，点击流消息数据在微信客户端的流转时序如图4：

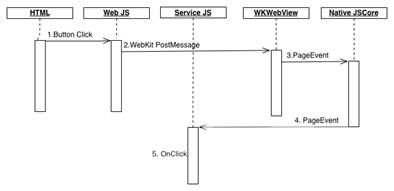


图4. 小程序按钮点击事件时序图

当前端Web JS监听到用户的按钮点击行为后，通过WebKit提供的消息传递机制（PostMessage）将点击事件发送给微信客户端当前页面的WKWebView，WKWebView再将该点击事件交由当前小程序的客户端Native Service环境，通过Native JSCore(JavaScriptCore)，回调执行到前端Service Js代码中的onClick监听函数。

下面依旧以按钮为例，通过伪代码实现来理解上述过程：

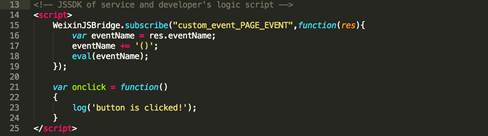
a、开发者在界面wxml中为button绑定监听函数：

小程序iOS客户端框架——控件事件逻辑框架与控件原生化

b、JSSDK将onClick事件发送到service



c、service中监听并执行绑定函数



上述流程中使用到的WeixinJSBridge对象承担了发送和监听事件消息的任务，publish函数负责发送消息到客户端，subsribe负责接收客户端publish的消息。将在下一节做详细的介绍

**3.**   **数据传输框架与WeixinJSBridge的实现**

在普通的H5页面开发模式下，每一个WebView页面是一个相对独立的运行环境，如果页面与页面之间有数据交互的需求，可以选择的通信方式较为单一，如采用cookie、localstorage，甚至通过query参数来进行数据传递。如前所述：一个小程序由多个WebView构成，H5的常规开发结构远远达不到小程序App开发的数据传输需求，也不符合App开发的习惯。

鉴于上述原因，微信客户端为每个小程序提供了独立的运行环境（小程序内部称为Service），该运行环境保持与小程序一致的生命周期，提供了该小程序运行中全部WebView的逻辑支撑能力：

A.          处理WebView控件上用户交互事件的能力

B.          为开发者提供相对隔离的逻辑开发环境

C.          提供WebView与WebView之间的数据通信能力

D.         监控小程序以及每个页面（WebView）的生命周期，以App事件的方式通知到开发者

上一节通过对按钮点击事件的处理，介绍了A能力的实现；对于B能力，iOS客户端采用了JavaScriptCore库作为小程序用户代码的运行环境，保证了运行环境的隔离；同时JavaScriptCore也提供了小程序能正常运行的核心功能C：即前端JavaScript脚本与客户端之间的数据通信能力的支持，该能力主要通过WeixinJSBridge对象来实现，下面就对WeixinJSBridge的设计做详细介绍。

为了满足小程序的通信需求，WeixinJSBridge需支持如下基本的通信接口：

l   通过JavaScript调用微信客户端（Objective C）中的函数；

l   微信客户端（Objective C）执行JavaScript脚本的function。

为了前端开发方便，WeixinJSBridge提供了同一套代码，同时对Webview和Service进行了能力支持。

**WeixinJSBridge.publish**

在Webview端，通过webkit提供的postMessage来将网页数据传输到Objective C监听函数，客户端直接透传到小程序service；在Service端调用执行Objective C中的block将数据传输到客户端，客户端再将数据透传到当前Webview。

**WeixinJSBridge.subscribe**

注册监听函数，监听客户端Objective C代码的函数调用。webview端监听Service中的publish调用；Service端则监听Webview中的publish调用。

**WeixinJSBridge.invoke**

传输逻辑与publish函数相同，不过该函数用来提供JSAPI的调用，函数调用到Objective C后，微信客户端将执行对应的JSAPI。

**WeixinJSBridge.on**

监听客户端主动抛出来的系统事件，比如小程序启动事件，页面切换事件，以及小程序切换后台事件。

客户端通过提供WeixinJSBridge对象，开发者就可以通过publish和subscribe实现在Service中通过js代码与小程序的WebView通信；通过invoke调用微信客户端的原生能力；并通过on接口监听微信传递过来的通知事件。

**4.**   **页面预加载与缓存机制**

在小程序中，为了提高页面运行速度，达到类原生体验，提供了页面预加载机制，开发者提交代码后，开发工具后台编译代码包时，会预生成page-frame.html（包含一些描述页面结构的 JavaScript 代码和所有页面通用样式的 CSS 代码）：

a、当小程序任务创建时，创建首页webview后，通过WKWebView提供的loadHTMLString接口，加载page-frame.html，页面特有的逻辑通过evaluateJavaScript执行插入到当前页面；

c、首页加载成功后，小程序会在后台预加载新的WebView，并通过loadHTMLString加载page-frame.html；

d、当需要跳转页面时，取缓存中的预加载页，并执行evaluateJavaScript执行页面特有的逻辑，同时需要补充缓存预加载页，为下一次跳转准备；

这种预加载机制极大减少了小程序页面跳转执行耗时，提高了用户的点击体验。

**5.**   **两种类型的API的设计与执行流程**

小程序的API分为两类：“组件API”和“开发API”。组件API并不直接暴露给开发者，开发API是直接提供给开发者调用的功能性API。开发者在开发过程中可以见到的API只有开发API；对于组件API，前端SDK会封装成组件提供给开发者使用，所以当开发者的页面中使用到了某个组件，并且这个组件使用到了客户端的某些原生功能，那么这个组件在初始化或运行过程中就会调用组件API。

图5展示的是两类API调用时，从前端调用到进入到微信客户端Objective C代码时，所经过的依赖模块，其中WeixinJSBridge在上一节已经做了详细的介绍，Service SDK和Webview SDK分别是前端对WeixinJSBridge的进一步功能性封装。

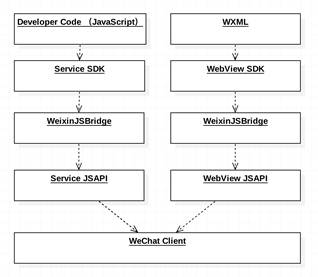


图5. 小程序组件相关模块依赖关系

**6.**   **原生控件的创建与交互机制**

小程序内部提供了部分非H5实现的原生控件。原生控件可以提供H5控件无法实现的一些功能， 原生控件的用户体验感受上也会更加流畅，另外，使用原生控件减少了Objective C代码与WebView通信的流程，降低了通信开销。

以画布为例，前端提供了wx-canvas控件给开发者，当开发者在页面中设置一个画布标签<canvas canvas-id="xxx" ></canvas>，并调用绘制接口时，前端SDK将会有如下JSAPI的调用流程：

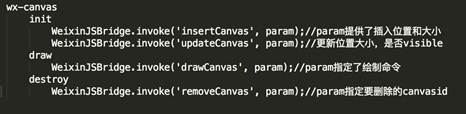


图6. 画布控件原生化创建逻辑

如上图所示，wx-canvas控件初始化时，将会通过Webview SDK的封装调用，执行客户端提供的“组件API”：insertCanvas接口以及updateCanvas接口（可选），绘制时通过调用客户端的drawCanvas接口，将绘制命令传递给客户端，客户端解析drawCanvas接口所带的参数，获取绘制命令集，并使用了Quarz2D来进行图形绘制。

insertCanvas通知客户端，在当前WebView上插入一个画布控件，客户端根据传入的位置和宽高参数来决定插入控件的位置和大小；

当开发者改变了wx-canvas控件的位置大小时，通过updateCanvas接口通知客户端，客户端对原生控件frame位置大小属性做对应的修改；

页面离开时，removeCanvas接口的调用将画布控件从webview上移除。

除了画布以外，Video组件对AVPlayer进行了封装，利用系统组件功能提供了边下边播的功能，并定制了原生化全屏等更加友好的用户操作界面；Map组件对QQ地图组件的封装将QQ地图的丰富功能引入到小程序，让开发者具有更广阔的开发想象空间；输入控件分别引入了iOS原生的UITexField和UITextView，提供了HTML输入框无法满足的定制化输入键盘等功能。

为了提供更加灵活可控的控件功能，小程序还对H5中的Toast、Alert、Picker、ActionSheet等控件做了原生化。这些组件是采用“开发API”的方式提供给开发者。

7.    **原生控件插入到网页DOM节点**

控件原生化带来了更加流畅的原生化体验和更加丰富的控件功能，但是同时也带来了新的难题。如前所述，原生控件是插入到webview控件上（实际实现时是插入到WKWebView下的WKScrollView下），如图7，网页元素总是绘制在WKContentView控件上——WKContentView负责绘制网页中的全部HTML元素，视频控件插入后将覆盖网页中的所有HTML元素：

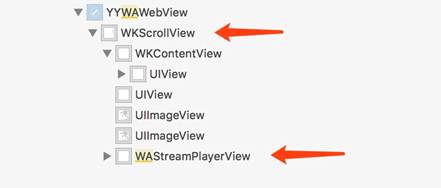


图7. 原生控件插入到WKWebView后将覆盖控件树中的HTML节点

如上图，插入的原生控件必然总是盖住网页（节点树中越靠下的节点，显示层级越高），这样就会导致：

a、如果开发者期望在原生控件上覆盖一些自定义HTML元素，将无法被支持到。

b、所有的H5弹出元素都会被原生控件遮挡，比如alert对话框。这一问题可以通过将H5的弹出组件都原生化得以解决，如上节提到的Toast、Alert、Picker、ActionSheet的原生化；

c、如果开发者在div滚动条中插入原生控件作为div的子节点，预期原生控件应该随着父节点div滚动条的滚动而移动，并且超出div区域的内容应该被裁掉，但是由于原生控件是直接插入到webview下，与div之间没有关联，所以不会跟随移动也不会被裁减，在表现上会出现与开发者预期不一致的情况，影响用户体验。

为了解决这一问题，客户端尝试对WKWebView解析HTML元素的原理进行分析，WKWebView在进行HTML解析时，会根据页面DOM元素在WKWebView控件下生成对应的iOS原生控件，通过分析，普通情况下生成的原生控件与HTML节点无对应关系，但是在某些特殊情况下，一些特殊DOM元素会在WebView的对应位置生成位置、大小完全一致的原生控件，如包含overflow属性的DIV标签，如下图所示：

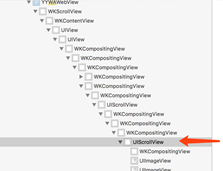
  

图6. WKWebView解析HTML在客户端生成对应的原生控件示例

如上图所示，WKWebView将在解析HTML时将该标签位置生成一个对应的UIScrollView控件。利用这个属性，我们可以在开发者期望插入原生控件的位置，预生成一个包含overflow标签的DIV节点，然后在插入原生控件时，将原生控件插入到该标签对应的UIScrollView上，就可以做到“原生控件不遮挡HTML元素”。例如将一个视频播放器插入到DOM节点以后，节点树如下：



图9. 将视频控件插入到网页DOM节点后的节点树

客户端采用的“原生控件插入到网页DOM节点”方案，具体实现原理如下：

a、WEB端预先在需要插入原生控件的预留位置插入一个具有overflow属性的DIV标签，并通过“组件API”insertContainer通知客户端该滚动条的位置、大小；

b、客户端根据insertContainer传入的位置和大小，在WKWebView下遍历找到这个DIV标签对应的UIScrollView（大小位置均一致），保存其对象指针，并分配一个id返回给WEB端；

c、当WEB端插入原生控件时，通过接口传入id通知客户端：该原生控件属于哪个div滚动条，客户端找到该滚动条对应的原生UIScrollView，并将控件插入到该UIScrollView下；

d、当页面的DOM元素发生变化时，需要通过updateContainer告诉客户端调整指定的原生控件的大小，客户端根据参数调整原生控件的大小（位置不需要调整，因为总是在相对于父控件的原点位置）。

插入DOM节点后原生控件事件处理。由于WKWebView会接管用户的所有操作事件，因此按照上述方案插入后，原生控件是无法响应用户事件的。因此需要对事件做特殊处理：通过重载WKWebView的hitTest方法，在该方法的处理逻辑中优先处理网页上的事件，如果网页未处理，再传递给原生控件。

**8.**   **总结**

微信客户端为小程序提供了整套运行环境：包括js脚本的运行时支持、小程序任务管理、service中的js脚本与webview之间的通信桥接机制，以及对复杂控件进行了原生化。从而为开发者及用户提供了良好的小程序体验。