EasyView开发经验梳理

1. 前言

关于EasyView系统的介绍,此处不再赘述, 相关的EasyView使用说明文档请到知识库中检索.
本文专注于介绍使用EasyView开发过程中的经验.

2. EasyView开发和H5开发的区别

公司以前的产品是使用H5开发的, 安卓提供一个APP的壳子, 里面有一个Webview, 来打开公司的H5页面, 和在PC端使用浏览器打开页面没有太大区别.
EasyView分为两部分: 页面制作系统和解析系统.
EasyView页面制作系统, 就是我们在浏览器上使用的EasyView界面.
可以使用拖拽的形式制作页面, 并配置页面之间的跳转. 还可以加入js脚本来丰富页面的效果.
最终,制作出的页面会生成一个json文件, 一个js文件, 以及若干上传的图片保存于服务器上.
EasyView解析系统,就是我们的weex. 是一个安卓原生应用.
可以从服务器请求json文件和js文件并解析, 最终调用安卓底层方法来渲染出原生页面.
EasyView系统和H5开发有较大区别.
使用EasyView开发产品, 首先要解决的还是思想的转变.

2.1. 页面内容

H5开发产品时,可以天马行空的在任意位置添加任意标签, 页面内容可能非常多, 非常丰富. 而渲染速度可能差别不是很大.
EasyView开发时, 如果页面中拖拽的元素过多的话, 互相覆盖起来, 是非常不利于维护的.
而且, 页面元素少的话, EasyView打开页面基本是秒开, 速度非常快.
所以EasyView不怕页面多, 就怕同一个页面元素太多.
EasyView开发第一条原则: 页面元素越少, 渲染速度越快.

2.2. 页面层级

H5开发产品时, 我们几乎可以不受限制的使用div进行嵌套, 嵌套层级较多时对页面渲染速度影响也不是很大.
EasyView开发时, 遇到过这种情况:
首页有一个瀑布流, 瀑布流下有若干button, 其中一个button中有一个gallery, gallery中有三个galleryItem, galleryItem中有image.
最终的效果是, 明显的感受到其他的button先渲染出来, 而含有gallery的那个button后渲染出来.
EasyView开发第二条原则: 页面层级嵌套越少, 渲染速度越快.

2.3. 精灵图

H5开发产品时, 会将所有小图标做到一张图片中, 在使用的时候通过设置背景图片位置的方式来调用. 这样可以减少HTTP请求, 加快页面速度.
EasyView开发时, 由于不能像CSS一样设置背景图片位置, 所以是不能使用精灵图的. 但是, 我们仍然可以通过多种方式来达到和精灵图一样的效果.
比如, 可以将一个页面中相邻的图标都做到同一张图片中(当然位置都是在页面中的真实位置, 而不是像精灵图那样全部挤到一起).
甚至还可以将所有小图片都放到背景图中, 能实现一样的效果,只不过换背景图的时候需要重新制作一次.
这样都可以达到减少页面元素并且减少HTTP请求的目的.

2.4. 动态页面

H5开发产品时, 我们经常使用模板页面(同一个页面, 传入不同的参数, 请求到不同的数据, 然后渲染出不同的页面. 当然只是图片或者文字不一致, 而页面结构都是一样的).
EasyView渲染页面的流程是这样的:
第一步: 读取json文件, 解析, 传到安卓底层并开始渲染.
第二步: 读取js文件, 解析, 执行, 调用安卓底层方法修改页面内容.
好的, 现在有一个最简单的需求, 一个页面中有一个image组件, 使用三种方法来做.

1. 直接拖拽并指定image组件的src地址. 所有信息都保存在json文件中.
2. 直接拖拽image组件, 但是没有指定src地址, 需要在js脚本中请求接口后才调用setSrc方法来设置图片地址.
3. 没有拖拽image组件, json中空空如也, 需要在js脚本中请求接口, 然后new出来这个image组件并add到page中.

毫无疑问, 就渲染速度来说, 1>2>3

拖拽生成的页面,由于json中指定了图片地址和文字内容, 会在第一步就开始渲染, 渲染的更早, 当然页面渲染成功的时间也更早.
而如果使用模板页面, 最初的json里什么内容都没有, 第一步等于没有开始渲染. 第二步的时候才去请求接口, 得到数据之后才开始渲染. 渲染成功的时间更晚.
当然也不是完全不能使用动态页面. 这只是一个建议.

2.5. 页面自己跳转到自己并传参

以前Epg页面经常有这样的逻辑: series页面跳转到series页面, 但是传的参数变了, 通过js脚本请求到不同的数据然后加载出来.
EasyView解析系统中, 每个元素都是有一个唯一的id值的. 父页面和子页面中有相同的id是会产生DOM冲突的.
因此, 这种页面自己跳转自己的逻辑, 必须先将这个页面全部销毁掉, 然后再重新加载并渲染. 速度慢, 体验差.
其实这种需求完全可以不使用页面自己跳转自己并传参来实现, 完全可以使用js脚本局部刷新页面来实现.

3. EasyView中的特殊组件或特殊用法

3.1. page组件

page组件是EasyView中比较特殊的组件.
EasyView解析系统中, 有一个根元素, id是'_root'. 每一个page都是这个根元素的子元素.
页面跳转时, 会将页面栈内前一个页面hide或者unload掉, 然后将新页面load出来.
页面返回时, 会将当前页面unload掉, 然后将页面栈内前一个页面show出来.
hide时触发page的hide事件;
load时触发page的load事件;
show时触发page的show事件;
unload时触发page的unload事件;
要注意善用page的这些事件, 很多功能都是要靠这些事件来实现的.

3.2. EasyView中的id

如前所述, EasyView解析系统中, 每个元素都是有一个唯一的id值.
但是, 拖拽产生的id都是这样的格式:`w1, w2, w3 ...`
为了避免DOM冲突, EasyView解析系统会自动将page的code加到每个严肃的id值前面. 比如:
该page的code是`home`, 那么该页面的所有元素的id会自动变成:`home_w1, home_w2, home_w3...`.
所以, js脚本中使用EV.WidgetManager.get()方法时, 要注意使用元素的真实id.
但是, 仍然有例外. 在js脚本中new出来并add到page中的元素, 解析系统不会去更改其id值. 所以如果你new的时候定义的是`'id':''w1'`, 那么其真实id就是w1.
当父页面和子页面都有new出来的元素的时候. 更应该注意DOM冲突.

3.3. List组件

5.1.3 版本的List组件是伪回收机制, 只能是请求数据源然后将ListItem一个一个的new出来然后add进List中.
在使用过程中会有以下问题:
1. 如果数据源数据较多, 将ListItem一个一个的new出来并add会耗费较多时间, 页面会有明显的卡顿.
2. 如果有这种需求: 切换导航栏时, 切换List的数据源,这种需求较难实现. 下面解释第二点为何难以实现.
一般遇到这种需求时, 思路是这样的: 既然需要切换数据源, 那么就将ListItem全部remove掉, 然后将新数据源的ListItem全部new出来并一个个add进List中.
但是,这种实现方式会遇到List错乱的问题.
因为js脚本发送remove命令给安卓底层,但是安卓底层响应是需要时间的. 在这个过程中, js又发送了add的命令. 安卓底层此时就会一边remove一边add, 就会导致List出现错乱.
针对这个需求, 实践过程中总结出了两种实现方式:
1. 游戏世界的实现方式: 既然List不能同时remove和add, 那么我在切换导航栏的时候直接将旧的List组件remove掉, 然后new出来一个新的List, 再将新的数据源中的ListItem都add到新List中.
2. 果果乐园的实现方式:复用同一个List, 请求到新的数据后, 刷新每个ListItem中的数据. 如果数据个数大于现有ListItem个数, 多的那些就需要new出来并add进List. 如果数据个数少于现有ListItem个数, 就需要remove掉多余的ListItem.
两种方法各有优劣, 并且都有成功的实现.
预计下个版本会对List做较大改动, 到时性能更好, 更易用. (本版本的List是基于阿里开源的weex框架老版本的List组件改造的. 目前weex更新了List组件, 改成了可回收机制的新List).

3.4. gallery组件

关于gallery组件, 切屏的时候有两种形式:
1. 左右两个箭头button, 切屏的时候光标先移动到这两个button上, 然后给这两个button设置focus事件, 在focus事件里调用gallery的next()方法或者prev()方法, 再根据gallery的index判断光标应该移动到哪个button上.
2. gallery每一屏最左边和最右边的button,注册beforeMove事件, 在beforeMove事件里调用gallery的next()方法或者prev()方法. 再调用setCurrentButtonId()方法将光标移动到指定button上.

需要注意的是, gallery的getIndex()方法, 是用来获取当前显示的是第几屏. 使用的时候容易出问题, 请注意.
有一个bug是这样的: 在调用完next()方法后立马调用getIndex()方法想去看看当前移动到了第几屏. 假设此时next()方法执行的是从第0屏到第1屏, 此时getIndex()方法获取的仍是0, 而不是1. 究其原因, 还是js执行的快, 而调用安卓底层方法慢.
当js执行完next()方法再执行getIndex()方法时, 安卓底层还未完成切屏的操作.

3.5. beforeMove事件

在以往的Epg页面中, beforeMove是个常用的事件.
EasyView解析系统中, 之前版本的光标移动逻辑是这样的,

1. 安卓端接收到按键监听后，传递给前端监听.
2. 前端接收到按键监听后, 判断上下左右, 触发beforeMove事件.
3. 如果beforeMove事件return false, 不执行第三步.
4. 找当前button对应方向上的下一个button, 光标设置过去。

5.1.3版本中的逻辑 是这样的：

1. 安卓端接收到按键监听后，判断上下左右, 直接查找当前button对应方向上的下个button, 将光标移动过去.
2. 将移动事件告诉前端(包括三个参数, 前一个buttonId, 移动方向, 新的buttonId).
3. 前端执行beforeMove事件.

这样的好处是显而易见的, 光标移动快, 非常快.
缺点: 由于光标时已经移动完了才通知前端的, 所以beforeMove事件里return false 不再能阻止光标移动.
此外还会导致开发人员有这样一个困惑: 既然这里是光标移动完毕之后才触发的事件, 为何不用afterMove来命名? 原因是为了兼容前面的代码.

5.1.3版中, button有beforeMove事件, Cursor也有beforeMove事件. 区别如下:

比如从A按钮移动到B按钮 如果A按钮注册了beforeMove事件, 那么执行A按钮的beforeMove事件, 不再执行Cursor的beforeMove事件.
如果A按钮没有注册beforeMove事件,那么执行Cursor的beforeMove事件.