**Android UI布局优化**

参考：<https://www.jianshu.com/p/2cc6d5842986>

<http://blog.csdn.net/harvic880925/article/details/45155965>

<http://blog.csdn.net/lmj623565791/article/details/45556391>

1. <include />
2. <merge />减少布局深度。
3. <ViewStub />高效占位符，是一个看不见的，不占布局位置，占用资源非常小的控件。为ViewStub指定一个布局，在Inflate布局时，只有ViewStub会被初始化，然后当ViewStub被设置为可见时，或调用了ViewStub.inflate()时，ViewStub所指向的布局就会被Inflate和实例化，并将ViewStub的布局属性传给它所指向的布局。
4. 合理选择控件容器
5. LinearLayout易用，效率高，表达能力有限；只能描述一个方向上连续排列的控件，容易导致布局嵌套太深，不符合布局扁平化的设计原理。
6. RelativeLayout复杂，表达能力强，但是效率稍逊；RelativeLayout主要在onMeasure和onLayout阶段会耗费更多时间。
7. 去掉window的默认背景和其他不必要的背景
8. ClipRect & QuickReject：
9. 自定义view的onDraw()方法，通过canvas.clipRect()来帮助系统识别那些可见的区域，只有在这个区域内才会被绘制，其他的区域会被忽视。
10. 自定义view的onDraw()方法，还可使用canvas.quickreject()来判断指定区域是否在裁剪范围外，若要绘制的元素在裁剪范围外，就可直接跳过绘制步骤。
11. 慎用Alpha：对一个View做Alpha转化，需先将View绘制出来，然后做Alpha转化，最后将转换后的效果绘制在界面上。即做Alpha转化需对当前View绘制两遍。

可以使用硬件加速，利用GPU的缓存机制提高效率，https://hencoder.com/ui-1-8/