# **2019Android View总结**

### View的滑动方式

a.layout(left,top,right,bottom):通过修改View四个方向的属性值来修改View的坐标，从而滑动View  
b.offsetLeftAndRight() offsetTopAndBottom():指定偏移量滑动view  
c.LayoutParams,改变布局参数：layoutParams中保存了view的布局参数，可以通过修改布局参数的方式滑动view  
d.通过动画来移动view：注意安卓的平移动画不能改变view的位置参数，属性动画可以  
e.scrollTo/scrollBy:注意移动的是view的内容，scrollBy(50,50)你会看到屏幕上的内容向屏幕的左上角移动了，这是参考对象不同导致的，你可以看作是它移动的是手机屏幕，手机屏幕向右下角移动，那么屏幕上的内容就像左上角移动了  
f.scroller:scroller需要配置computeScroll方法实现view的滑动，scroller本身并不会滑动view，它的作用可以看作一个插值器，它会计算当前时间点view应该滑动到的距离，然后view不断的重绘，不断的调用computeScroll方法，这个方法是个空方法，所以我们重写这个方法，在这个方法中不断的从scroller中获取当前view的位置，调用scrollTo方法实现滑动的效果

### View的事件分发机制

点击事件产生后，首先传递给Activity的dispatchTouchEvent方法，通过PhoneWindow传递给DecorView,然后再传递给根ViewGroup,进入ViewGroup的dispatchTouchEvent方法，执行onInterceptTouchEvent方法判断是否拦截，再不拦截的情况下，此时会遍历ViewGroup的子元素，进入子View的dispatchToucnEvent方法，如果子view设置了onTouchListener,就执行onTouch方法，并根据onTouch的返回值为true还是false来决定是否执行onTouchEvent方法，如果是false则继续执行onTouchEvent，在onTouchEvent的Action Up事件中判断，如果设置了onClickListener ,就执行onClick方法。

### View的加载流程

View随着Activity的创建而加载，startActivity启动一个Activity时，在ActivityThread的handleLaunchActivity方法中会执行Activity的onCreate方法，这个时候会调用setContentView加载布局创建出DecorView并将我们的layout加载到DecorView中，当执行到handleResumeActivity时，Activity的onResume方法被调用，然后WindowManager会将DecorView设置给ViewRootImpl,这样，DecorView就被加载到Window中了，此时界面还没有显示出来，还需要经过View的measure，layout和draw方法，才能完成View的工作流程。我们需要知道View的绘制是由ViewRoot来负责的，每一个DecorView都有一个与之关联的ViewRoot,这种关联关系是由WindowManager维护的，将DecorView和ViewRoot关联之后，ViewRootImpl的requestLayout会被调用以完成初步布局，通过scheduleTraversals方法向主线程发送消息请求遍历，最终调用ViewRootImpl的performTraversals方法，这个方法会执行View的measure layout 和draw流程

### View的measure layout 和 draw流程

在上边的分析中我们知道，View绘制流程的入口在ViewRootImpl的performTraversals方法，在方法中首先调用performMeasure方法，传入一个childWidthMeasureSpec和childHeightMeasureSpec参数，这两个参数代表的是DecorView的MeasureSpec值，这个MeasureSpec值由窗口的尺寸和DecorView的LayoutParams决定，最终调用View的measure方法进入测量流程

##### measure：

View的measure过程由ViewGroup传递而来，在调用View.measure方法之前，会首先根据View自身的LayoutParams和父布局的MeasureSpec确定子view的MeasureSpec，然后将view宽高对应的measureSpec传递到measure方法中，那么子view的MeasureSpec获取规则是怎样的？分几种情况进行说明  
1.父布局是EXACTLY模式：  
 a.子view宽或高是个确定值，那么子view的size就是这个确定值，mode是EXACTLY（是不是说子view宽高可以超过父view？见下一个）  
 b.子view宽或高设置为match\_parent,那么子view的size就是占满父容器剩余空间，模式就是EXACTLY  
 c.子view宽或高设置为wrap\_content,那么子view的size就是占满父容器剩余空间，不能超过父容器大小，模式就是AT\_MOST

2.父布局是AT\_MOST模式：  
a.子view宽或高是个确定值，那么子view的size就是这个确定值，mode是EXACTLY  
 b.子view宽或高设置为match\_parent,那么子view的size就是占满父容器剩余空间,不能超过父容器大小，模式就是AT\_MOST  
 c.子view宽或高设置为wrap\_content,那么子view的size就是占满父容器剩余空间，不能超过父容器大小，模式就是AT\_MOST

3.父布局是UNSPECIFIED模式：  
a.子view宽或高是个确定值，那么子view的size就是这个确定值，mode是EXACTLY  
b.子view宽或高设置为match\_parent,那么子view的size就是0，模式就是UNSPECIFIED  
c.子view宽或高设置为wrap\_content,那么子view的size就是0，模式就是UNSPECIFIED

获取到宽高的MeasureSpec后，传入view的measure方法中来确定view的宽高，这个时候还要分情况

1.当MeasureSpec的mode是UNSPECIFIED,此时view的宽或者高要看view有没有设置背景，如果没有设置背景，就返回设置的minWidth或minHeight,这两个值如果没有设置默认就是0，如果view设置了背景，就取minWidth或minHeight和背景这个drawable固有宽或者高中的最大值返回  
2.当MeasureSpec的mode是AT\_MOST和EXACTLY，此时view的宽高都返回从MeasureSpec中获取到的size值，这个值的确定见上边的分析。因此如果要通过继承view实现自定义view，一定要重写onMeasure方法对wrap\_conten属性做处理，否则，他的match\_parent和wrap\_content属性效果就是一样的

**layout:**

layout方法的作用是用来确定view本身的位置，onLayout方法用来确定所有子元素的位置，当ViewGroup的位置确定之后，它在onLayout中会遍历所有的子元素并调用其layout方法，在子元素的layout方法中onLayout方法又会被调用。layout方法的流程是，首先通过setFrame方法确定view四个顶点的位置，然后view在父容器中的位置也就确定了，接着会调用onLayout方法，确定子元素的位置，onLayout是个空方法，需要继承者去实现。

getMeasuredHeight和getHeight方法有什么区别？getMeasuredHeight（测量高度）形成于view的measure过程，getHeight（最终高度）形成于layout过程，在有些情况下，view需要measure多次才能确定测量宽高，在前几次的测量过程中，得出的测量宽高有可能和最终宽高不一致，但是最终来说，还是会相同，有一种情况会导致两者值不一样，如下，此代码会导致view的最终宽高比测量宽高大100px

public void layout(int l,int t,int r, int b){

super.layout(l,t,r+100,b+100);}

draw:

View的绘制过程遵循如下几步：  
a.绘制背景 background.draw(canvas)  
b.绘制自己（onDraw）  
c.绘制children（dispatchDraw）  
d.绘制装饰（onDrawScrollBars）

View绘制过程的传递是通过dispatchDraw来实现的，它会遍历所有的子元素的draw方法，如此draw事件就一层一层的传递下去了

ps：view有一个特殊的方法setWillNotDraw，如果一个view不需要绘制内容，即不需要重写onDraw方法绘制，可以开启这个标记，系统会进行相应的优化。默认情况下，View没有开启这个标记，默认认为需要实现onDraw方法绘制，当我们继承ViewGroup实现自定义控件，并且明确知道不需要具备绘制功能时，可以开启这个标记，如果我们重写了onDraw,那么要显示的关闭这个标记

子view宽高可以超过父view？能

1.android:clipChildren = "false" 这个属性要设置在父 view 上。代表其中的子View 可以超出屏幕。  
2.子view 要有具体的大小，一定要比父view 大 才能超出。比如 父view 高度 100px 子view 设置高度150px。子view 比父view大，这样超出的属性才有意义。（高度可以在代码中动态赋值，但不能用wrap\_content / match\_partent）。  
3.对父布局还有要求，要求使用linearLayout(反正我用RelativeLayout 是不行)。你如果必须用其他布局可以在需要超出的view上面套一个linearLayout 外面再套其他的布局。  
4.最外面的布局如果设置的padding 不能超出

### 自定义view需要注意的几点

1.让view支持wrap\_content属性，在onMeasure方法中针对AT\_MOST模式做专门处理，否则wrap\_content会和match\_parent效果一样（继承ViewGroup也同样要在onMeasure中做这个判断处理）

if(widthMeasureSpec == MeasureSpec.AT\_MOST && heightMeasureSpec == MeasureSpec.AT\_MOST){

setMeasuredDimension(200,200); // wrap\_content情况下要设置一个默认值，200只是举个例子，最终的值需要计算得到刚好包裹内容的宽高值

}else if(widthMeasureSpec == MeasureSpec.AT\_MOST){

setMeasuredDimension(200,heightMeasureSpec );

}else if(heightMeasureSpec == MeasureSpec.AT\_MOST){

setMeasuredDimension(heightMeasureSpec ,200);

}

2.让view支持padding（onDraw的时候，宽高减去padding值，margin由父布局控制，不需要view考虑），自定义ViewGroup需要考虑自身的padding和子view的margin造成的影响  
3.在view中尽量不要使用handler，使用view本身的post方法  
4.在onDetachedFromWindow中及时停止线程或动画  
5.view带有滑动嵌套情形时，处理好滑动冲突

### ACTION\_DOWN没有拦截，ACTION\_MOVE ACTION\_UP还会拦截吗