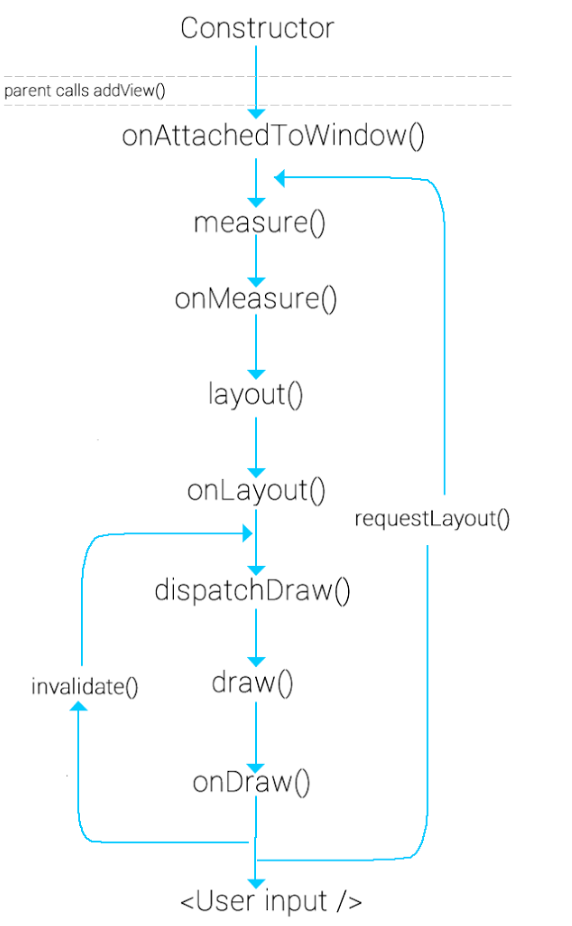
# View 绘制流程



#### Measure 过程

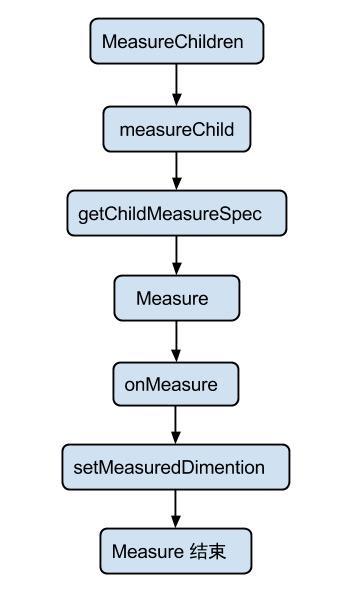
1. 测量过程由上至下，在measure过程的最后，每个视图将存储自己的尺寸大小和测量规格。

2. measure过程会为一个View及其所有子节点的mMeasureWidth和mMeasuredHeight变量赋值， 该值可以通过getMeasuredWidth和getMeasuredHeight方法获得。

3. measure过程的核心方法: **measure() - onMeasure() - setMeasuredDimension().**

**setMeasuredDimension**是测量阶段的终极方法，在onMeasure()方法中调用，将计算得到的尺寸，传递给该方法，测量阶段结束。在自定义     视图时，不需要关心系统复杂的Measure过程，只需要调用setMeasuredDimension()设置根据MeasureSpec计算得到的尺寸即可。同时,    onMeasure()方法也必须调用setMeasuredDimension()方法来设置重新测量之后的

    以measureChildren为例的调用链图:



#### Layout 过程

1. 子视图的具体位置都是相对于父视图而言的。View的onLayout()方法为空实现，而ViewGroup的onLayout为abstract，因此，自定义的View要继承ViewGroup时，必须实现onLayout函数。

2. 在Layout过程中，子视图会调用getMeasuredWidth()和getMeasuredHeight()方法获取到measure过程得到mMeasuredWidth和mMeasuredHeight，作为自己的width和height。然后调用每一个子视图的layout()，来确定每个子视图在父视图中的位置。

#### Draw 过程

1. 所有视图最终都是调用View的draw方法进行绘制。 在自定义视图中， 也不应该复写该方法， 而是复写onDraw（）方法进行绘制， 如果自定义的视图确实要复写该方法，先调用super.draw()完成系统的绘制，再进行自定义的绘制。

2.  onDraw()方法默认是空实现，自定义绘制过程需要复写方法，绘制自身的内容。

3. dispatchDraw()发起对子视图的绘制，在View中默认为空实现，ViewGroup复写了dispatchDraw()来对其子视图进行绘制。自定义的ViewGroup不应该对dispatchDraw()进行复写。

#### 如何对自定义View进行控制

1. 如果想控制View在屏幕上的渲染效果，就在重写onDraw()方法，在里面进行相应的处理。

2. 如果想要控制用户同View之间的交互操作，则在onTouchEvent()方法中对手势进行控制处理。

3. 如果想要控制View中内容在屏幕上显示的尺寸大小，就重写onMeasure()方法中进行处理。

4. 在 XML文件中设置自定义View的XML属性。

5. 如果想避免失去View的相关状态参数的话，就在onSaveInstanceState() 和 onRestoreInstanceState()方法中保存有关View的状态信息。