 掌握ConstraintLayout

# Bias

对应的属性

app:layout\_constraintVertical\_bias="1"  
app:layout\_constraintHorizontal\_bias="1"

bias，决定的是在在两个view之间比例的位置

例如：

app:layout\_constraintLeft\_toLeftOf="parent"  
app:layout\_constraintRight\_toLeftOf="@+id/button1"

app:layout\_constraintHorizontal\_bias="0"



# chain

多个view使用以下属性相互关联就建立了链条

app:layout\_constraintEnd\_toStartOf="@+id/button11"

app:layout\_constraintStart\_toStartOf="parent"

连表头设置chainStyle:

app:layout\_constraintHorizontal\_chainStyle="spread"

1. spread - 元素将被展开（默认样式）



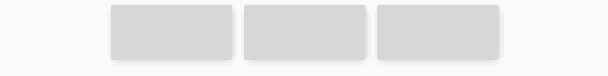
加权链 - 在spread模式下，如果某些小部件设置为MATCH\_CONSTRAINT(也就是0dp)，则它们将拆分可用空间



spread\_inside - 类似，但链的端点将不会扩展



packed - 链的元素将被打包在一起。 孩子的水平或垂直偏差属性将影响包装元素的定位



# 2.weight权重

在 XML 创建 Chain 链，然后实现如上的效果只需要对 textView3 修改属性 android:layout\_width="0dp" 并且设置新属性 app:layout\_constraintHorizontal\_weight="1"，如下：

如果链条模式是 spread 或者 spread inside ,我们也可以设置一个或多个view的大小为 0dp，从而让view占满余下的间距部分。默认情况下，如果有一个view的大小设置了0dp,那么该view会填满空余部分；如果多个view设置了0dp, 那么这几个view就会平均分配空余部分（空余部分会被囊括到View中），但是，我们可以通layout\_constraintHorizontal\_weight和 layout\_constraintVertical\_weight 属性调整这几个view的权重（也叫比重），就像在 LinearLayout中使用 layout\_weight 一样。设置权重之后，权重大的所占的区域就多，权重小的所占区域就小。

# 辅助线(Guideline)

<androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 />

Orientation:决定是垂直辅助线还是水平辅助线

app:layout\_constraintGuide\_percent="0.5"  
app:layout\_constraintGuide\_begin="20dp"  
app:layout\_constraintGuide\_end="20dp"

Guideline只能使用自己独有的属性设置位置，layout\_constraintGuide\_percent设置位置百分比，0最左边，1最右边，0.5居中

距离左边距离只能使用layout\_constraintGuide\_begin

# 障碍线(Barrier)

将多个元素放到这个障碍线里面使时，其中的任何元素的大小或位置变化时都会使它的位置进行改变。

<androidx.constraintlayout.widget.Barrier

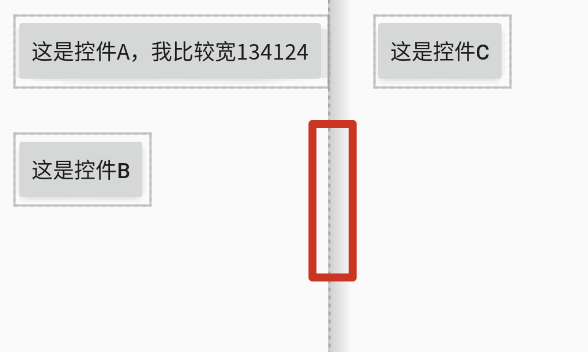
android:id="@+id/barrier"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:barrierDirection="end"

app:constraint\_referenced\_ids="btn\_a,btn\_b"/>



障碍线会根据左边内容改变而移动，这是和辅助线最大的区别

barrierDirection决定在constraint\_referenced\_ids中所有view的位置，可以是上下左右

# 分组(Group)

可以关联多个View, 通过设置group决定多个View的显示与否

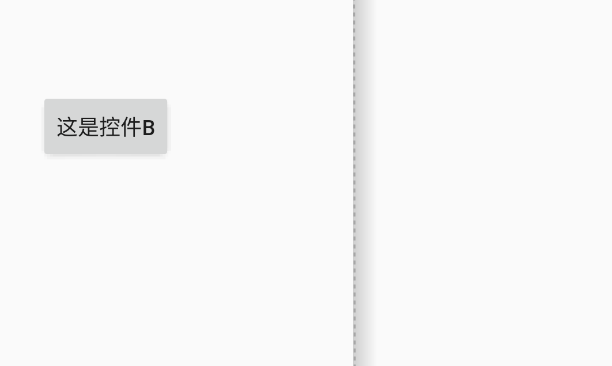
<androidx.constraintlayout.widget.Group

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:visibility="invisible"

app:constraint\_referenced\_ids="btn\_a,btn\_c"/>



# 按比例设置视图大小

View或者某张图片按比例显示，以适应不同的屏幕分辨率

<Button  
 android:id="@+id/btn\_c"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginTop="200dp"  
 android:text="这是控件C"  
 app:layout\_constraintLeft\_toLeftOf="@id/barrier"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintDimensionRatio="h,3:5"/>

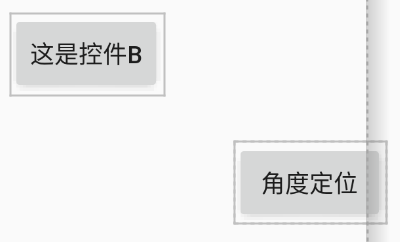


如果宽高都有确定值就以实际值为准，如果为match\_constrain(0dp)则是根据宽高比，layout\_constraintDimensionRatio="h,3:5"代表宽:高=3:5，同理w,3:5 意思是 高:宽=3:5

# 6.角度定位

用一个角度和一个距离来约束两个空间的中心

<Button  
 android:id="@+id/TextView2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 android:text="角度定位"  
 app:layout\_constraintCircle="@+id/btn\_b"  
 app:layout\_constraintCircleAngle="120"  
 app:layout\_constraintCircleRadius="150dp" />



# 7. goneMargin

goneMargin主要用于约束的控件可见性被设置为gone的时候使用的margin值

app:layout\_constraintLeft\_toRightOf="@+id/TextView1"

app:layout\_goneMarginLeft="10dp"

# 8 尺寸约束

当控件的高度或宽度为wrap\_content时，可以使用下列属性来控制最大、最小的高度或宽度：

android:minWidth 最小的宽度

android:minHeight 最小的高度

android:maxWidth 最大的宽度

android:maxHeight 最大的高度

注意！当ConstraintLayout为1.1版本以下时，使用这些属性需要加上强制约束，如下所示：

app:constrainedWidth=”true”

app:constrainedHeight=”true”

使用 0dp (MATCH\_CONSTRAINT)

官方不推荐在ConstraintLayout中使用match\_parent，可以设置 0dp (MATCH\_CONSTRAINT) 配合约束代替match\_parent，举个例子：

#### 4.1 Optimizer

当我们使用 MATCH\_CONSTRAINT 时，ConstraintLayout 将对控件进行 2 次测量，ConstraintLayout在1.1中可以通过设置 **layout\_optimizationLevel** 进行优化，可设置的值有：

none：无优化

standard：仅优化直接约束和屏障约束（默认）

direct：优化直接约束

barrier：优化屏障约束

chain：优化链约束

dimensions：优化尺寸测量

#### 4.4 Placeholder

Placeholder指的是占位符。在Placeholder中可使用setContent()设置另一个控件的id，使这个控件移动到占位符的位置。举个例子：

<android.support.constraint.Placeholder

android:id="@+id/placeholder"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:content="@+id/textview"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<TextView

android:id="@+id/textview"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="#cccccc"

android:padding="16dp"

android:text="TextView"

android:textColor="#000000"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

新建一个Placeholder约束在屏幕的左上角，新建一个TextView约束在屏幕的右上角，在Placeholder中设置 app:content="@+id/textview"，这时TextView会跑到屏幕的左上角。效果如下：

