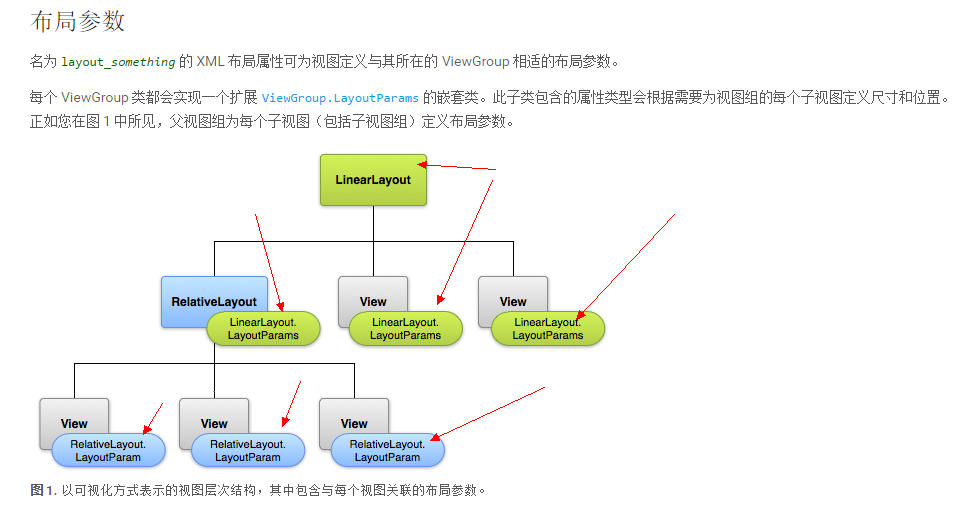
<https://developer.android.com/guide/topics/ui/declaring-layout.html>



## 布局位置

视图的几何形状就是矩形的几何形状。视图具有一个位置（以一对水平向左和垂直向上坐标表示）和两个尺寸（以宽度和高度表示）。 位置和尺寸的单位是像素。

可以通过调用方法 [getLeft()](https://developer.android.com/reference/android/view/View.html#getLeft()) 和方法 [getTop()](https://developer.android.com/reference/android/view/View.html#getTop()) 来检索视图的位置。前者会返回表示视图的矩形的水平向左（或称 X 轴） 坐标。后者会返回表示视图的矩形的垂直向上（或称 Y 轴）坐标。 这些方法都会返回视图相对于其父项的位置。 例如，如果 getLeft() 返回 20，则意味着视图位于其直接父项左边缘向右 20 个像素处。

此外，系统还提供了几种便捷方法来避免不必要的计算，即 [getRight()](https://developer.android.com/reference/android/view/View.html#getRight()) 和 [getBottom()](https://developer.android.com/reference/android/view/View.html#getBottom())。 这些方法会返回表示视图的矩形的右边缘和下边缘的坐标。 例如，调用 [getRight()](https://developer.android.com/reference/android/view/View.html#getRight()) 类似于进行以下计算：getLeft() + getWidth()。

## 尺寸、内边距和外边距

视图的尺寸通过宽度和高度表示。视图实际上具有两对宽度和高度值。

第一对称为测量宽度和测量高度。 这些尺寸定义视图想要在其父项内具有的大小。 这些测量尺寸可以通过调用 [getMeasuredWidth()](https://developer.android.com/reference/android/view/View.html#getMeasuredWidth()) 和 [getMeasuredHeight()](https://developer.android.com/reference/android/view/View.html#getMeasuredHeight()) 来获得。

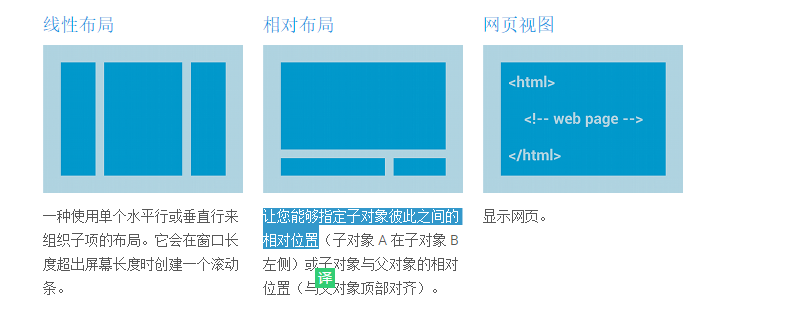
第二对简称为宽度和高度，有时称为绘制宽度和绘制高度。 这些尺寸定义视图在绘制时和布局后在屏幕上的实际尺寸。 这些值可以（但不必）与测量宽度和测量高度不同。 宽度和高度可以通过调用 [getWidth()](https://developer.android.com/reference/android/view/View.html#getWidth()) 和 [getHeight()](https://developer.android.com/reference/android/view/View.html#getHeight()) 来获得。

要想测量其尺寸，视图需要将其内边距考虑在内。内边距以视图左侧、顶部、右侧和底部各部分的像素数表示。 内边距可用于以特定数量的像素弥补视图的内容。 例如，左侧内边距为 2，会将视图的内容从左边缘向右推 2 个像素。 可以使用 [setPadding(int, int, int, int)](https://developer.android.com/reference/android/view/View.html#setPadding(int, int, int, int)) 方法设置内边距，并通过调用 [getPaddingLeft()](https://developer.android.com/reference/android/view/View.html#getPaddingLeft())、[getPaddingTop()](https://developer.android.com/reference/android/view/View.html#getPaddingTop())、[getPaddingRight()](https://developer.android.com/reference/android/view/View.html#getPaddingRight()) 和 [getPaddingBottom()](https://developer.android.com/reference/android/view/View.html#getPaddingBottom()) 进行查询。

尽管视图可以定义内边距，但它并不支持外边距。 不过，视图组可以提供此类支持。如需了解更多信息，请参阅 [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) 和 [ViewGroup.MarginLayoutParams](https://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html)。

如需了解有关尺寸的详细信息，请参阅[尺寸值](https://developer.android.com/guide/topics/resources/more-resources.html#Dimension)。

## 常见布局



**线性布局:**

[**http://www.runoob.com/w3cnote/android-tutorial-linearlayout.html**](http://www.runoob.com/w3cnote/android-tutorial-linearlayout.html)

**我们屏幕适配的使用 用的比较多的就是LinearLayout的weight(权重属性)**

**③Java代码中设置weight属性：**

**setLayoutParams(new LayoutParams(LayoutParams.FILL\_PARENT,**

**LayoutParams.WRAP\_CONTENT, 1));**





<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/LinearLayout1"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:divider="@drawable/ktv\_line\_div"

android:orientation="vertical"

android:showDividers="middle"

android:dividerPadding="10dp"

tools:context="com.jay.example.linearlayoutdemo.MainActivity" >

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="按钮1" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="按钮2" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="按钮3" />

</LinearLayout>

## 5.注意事项:

使用Layout\_gravity的一个很重要的问题!!! 问题内容: 在一个LinearLayout的水平方向中布置两个TextView,想让一个左,一个右,怎么搞? 或许你会脱口而出:"gravity设置一个left,一个right就可以啦!" 真的这么简单?你试过吗?写个简单的Layout你就会发现,事与愿违了: 代码如下：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal"

tools:context="com.jay.example.getscreendemo.MainActivity" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="200dp"

android:layout\_gravity="left"

android:background="#FF7878"

android:gravity="center"

android:text="O(∩\_∩)O哈哈~" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="200dp"

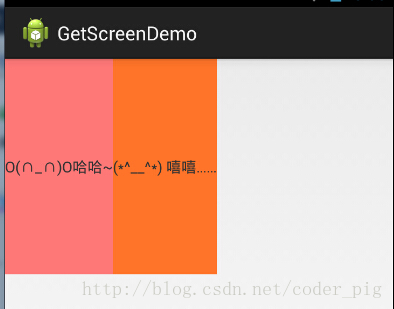
android:layout\_gravity="right"

android:background="#FF7428"

android:gravity="center"

android:text="(\*^\_\_^\*) 嘻嘻……" />

</LinearLayout>

看到这里你会说:哎呀,真的不行耶,要不在外层LinearLayout加个gravity=left的属性,然后设置第二个 TextView的layout\_gravity为right,恩,好我们试一下:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal"

android:gravity="left"

tools:context="com.jay.example.getscreendemo.MainActivity" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="200dp"

android:background="#FF7878"

android:gravity="center"

android:text="O(∩\_∩)O哈哈~" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="200dp"

android:layout\_gravity="right"

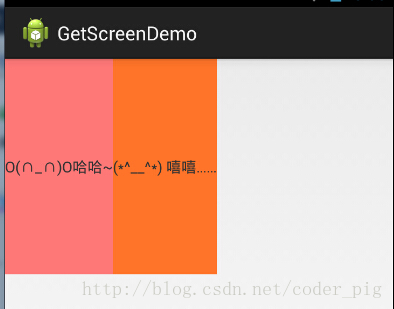
android:background="#FF7428"

android:gravity="center"

android:text="(\*^\_\_^\*) 嘻嘻……" />

</LinearLayout>

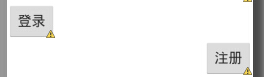
结果还是一样：



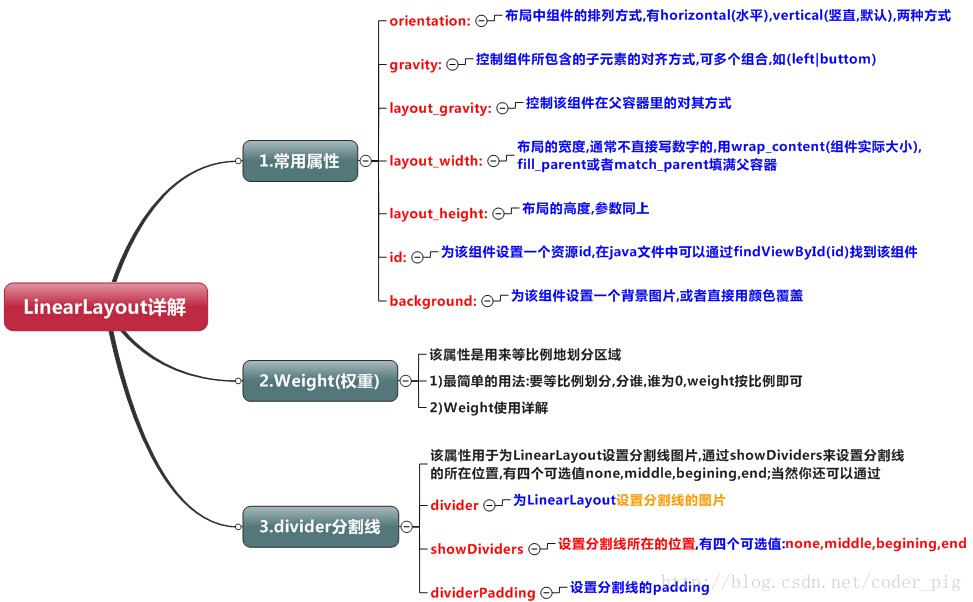
好吧,没辙了,怎么办好?

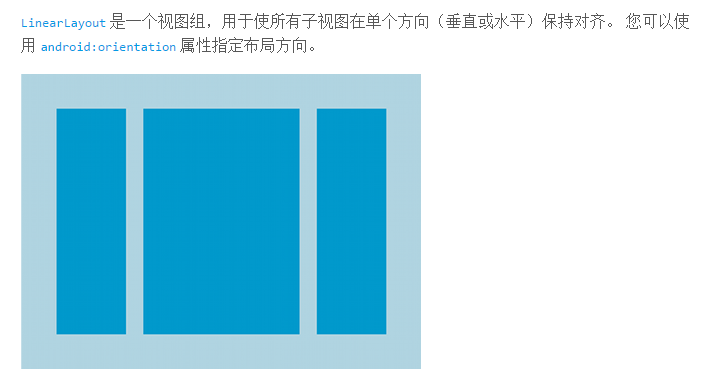
***当 android:orientation="vertical" 时， 只有水平方向的设置才起作用，垂直方向的设置不起作用。 即：left，right，center\_horizontal 是生效的。 当 android:orientation="horizontal" 时， 只有垂直方向的设置才起作用，水平方向的设置不起作用。 即：top，bottom，center\_vertical 是生效的。***

然而，这方法好像并没有什么卵用。比如： 如果只能竖直方向设置左右对齐的话,就会出现下面的效果:



这显然不是我们要的结果把! 综上,要么按照上述给出的规则来布局,不过对于这种情况还是使用相对布局RelativeLayout把！ 网上没给出具体的原因,都是说这样改有人说这个和orientation的优先级有关 ,暂且先mark下来吧,后续如果知道原因的话再解释!前面屏幕适配也说过了,**布局还是建议使用 RelativeLayout!**



[LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) 的所有子视图依次堆叠，因此无论子视图有多宽，垂直列表每行均只有一个子视图，水平列表将只有一行高（最高子视图的高度加上内边距）。 [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) 遵守子视图之间的“边距”以及每个子视图的“重力”（右对齐、居中对齐、左对齐）。

**什么是LinearLayout**

作为最基础的布局，所以从事过Android开发的同学都应该非常了解   
中文解释应该叫做线性布局，相比如RelativeLayout，LinearLayout更简单，在没有weight的情况也每次只要测量一次就够，而RelativeLayout每次都需要测量两次

<http://www.cnblogs.com/zhangs1986/archive/2013/01/17/2864237.html>

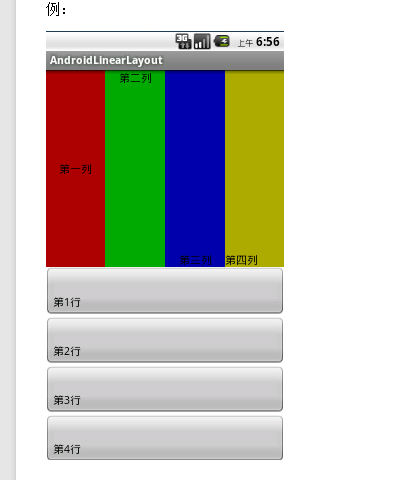
要点：

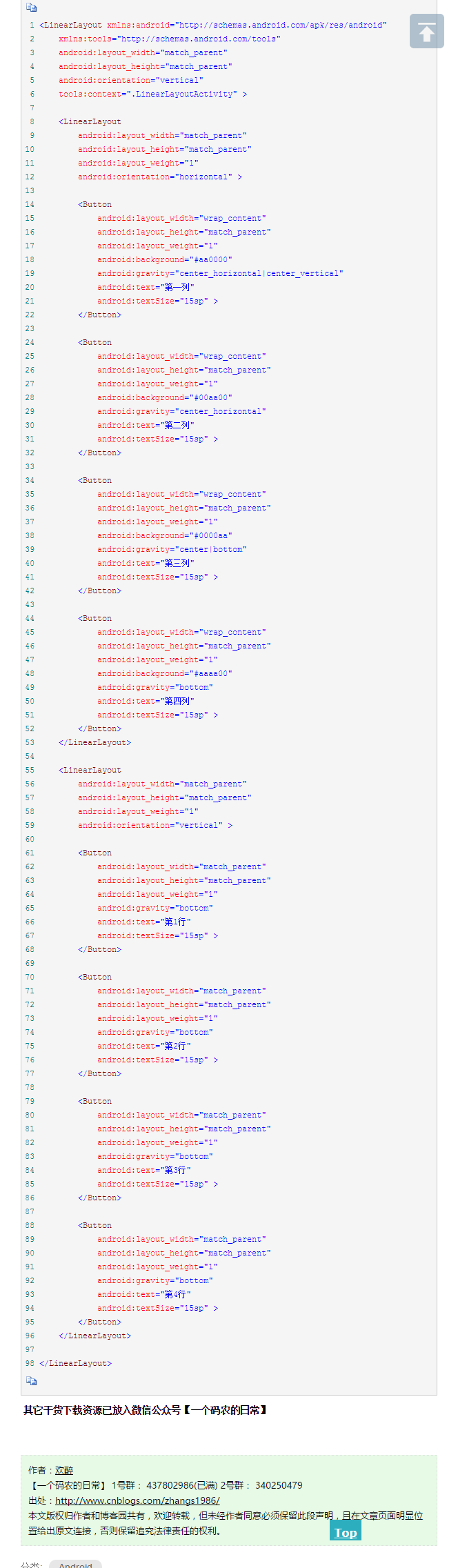
android:orientation="**vertical**"垂直线性布局,"**horizontal**"水平线性布局

android:gravity**="top"(buttom、left、right、center\_vertical、fill\_vertical、center\_horizontal、fill\_horizontal、center、fill、clip\_vertical、clip\_horizontal)**控制布局中控件的对齐方式。如果是没有子控件的控件设置此属性，表示其内容的对齐方式，比如说TextView里面文字的对齐方式；若是有子控件的控件设置此属性，则表示其子控件的对齐方式，**gravity如果需要设置多个属性值，需要使用“|”进行组合**

**android:gravity 与 android:layout\_gravity的区别  
android:gravity是指定本元素的子元素相对它的对齐方式。  
android:layout\_gravity是指定本元素相对它的父元素的对齐方式。**

 android:layout\_weight**="1"**通过设置控件的layout\_weight属性以控制各个控件在布局中的**相对大小**,线性布局会根据该控件layout\_weight值与其所处布局中所有控件layout\_weight值之和的比值为该控件分配占用的区域。在水平布局的LinearLayout中有两个Button，这两个Button的layout\_weight属性值**都为1**，那么这两个按钮**都会被拉伸到整个屏幕宽度的一半**。如果layout\_weight指**为0**，控件会按原大小显示，**不会被拉伸**；对于其余layout\_weight属性值大于0的控件，系统将会**减去layout\_weight属性值为0的控件的宽度或者高度**，再用剩余的宽度或高度按相应的比例来分配每一个控件显示的宽度或高度。





**从上面可以看出啊，layout\_XXX是相对于同级的视图起作用的，**

**而没有layout前缀的则:①没有子视图的情况，则是对自己的内容起作用②有子视图的情况，则是对其子视图起作用。**

orientation="**vertical**" 这个是其子视图是垂直放置的，垂直的网上码

"**horizontal呢，则是全都是水平的码，垂直方向不管的**

https://developer.android.com/guide/topics/ui/layout/relative.html

# Relative Layout

[RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) area (such as aligned to the bottom, left or center).

[RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) lets child views specify their position relative to the parent view or to each other (specified by ID). So you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on. **By default, all child views are drawn at the top-left of the layout,** so you must define the position of each view using the various layout properties available from [RelativeLayout.LayoutParams](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html).

[android:layout\_alignParentTop](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_alignParentTop)

If "true", makes the top edge of this view match the top edge of the parent.

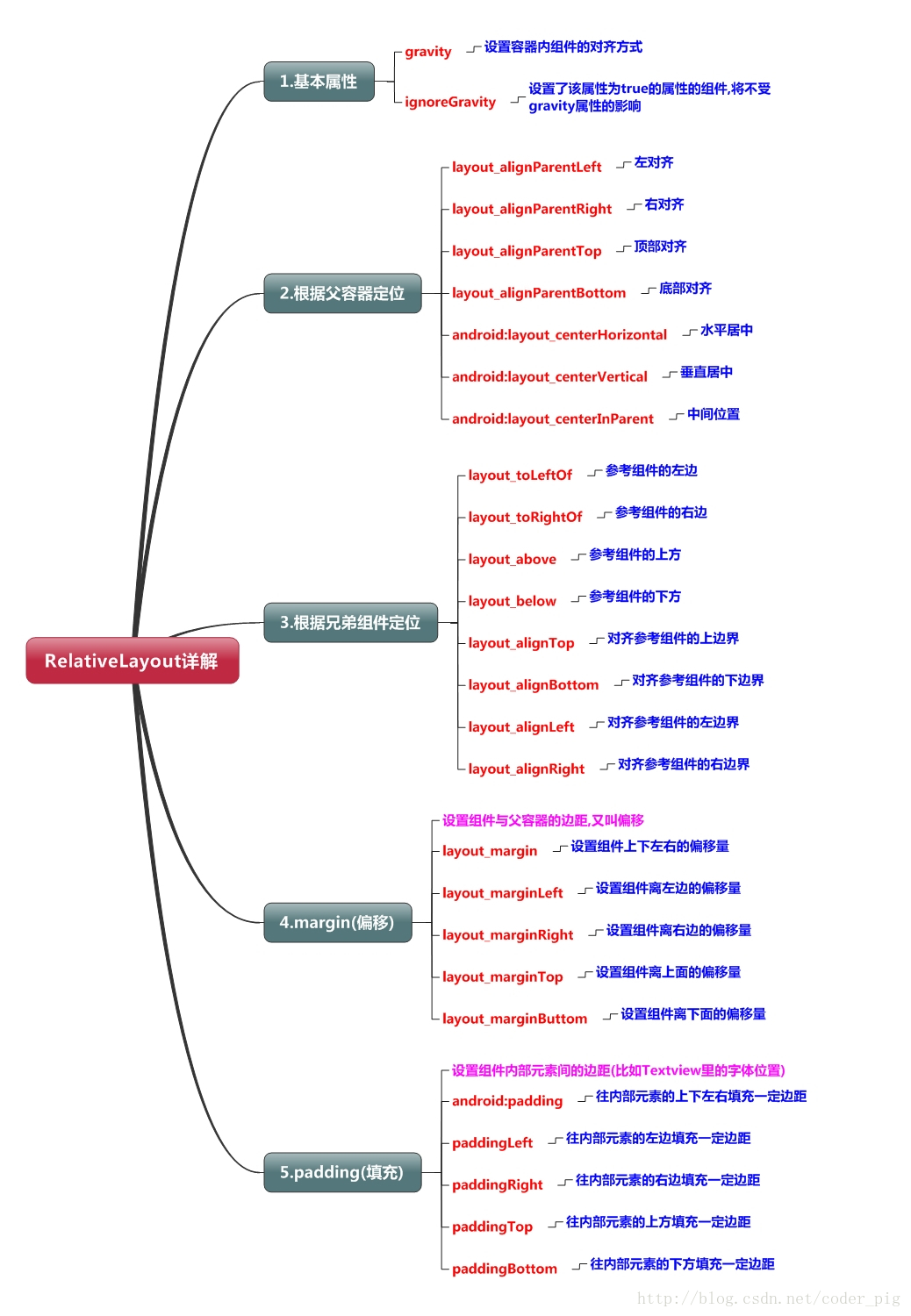
[android:layout\_centerVertical](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_centerVertical)

If "true", centers this child vertically within its parent.

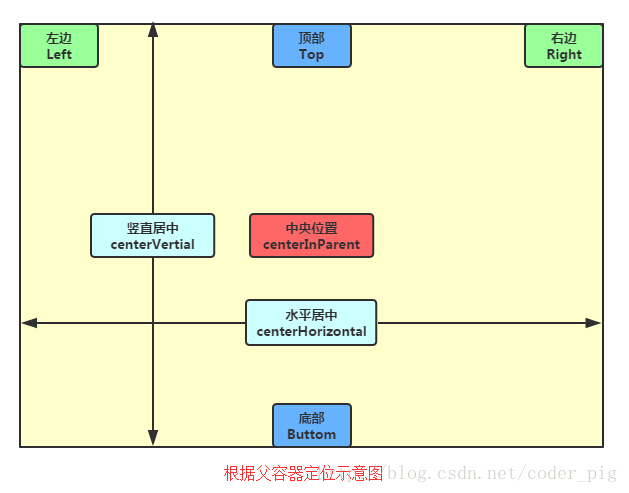
**这里的layout\_前缀却和上面的作用不太一样了，这里主要是用于和父视图的对齐，也可以指定某个兄弟视图进行对齐**

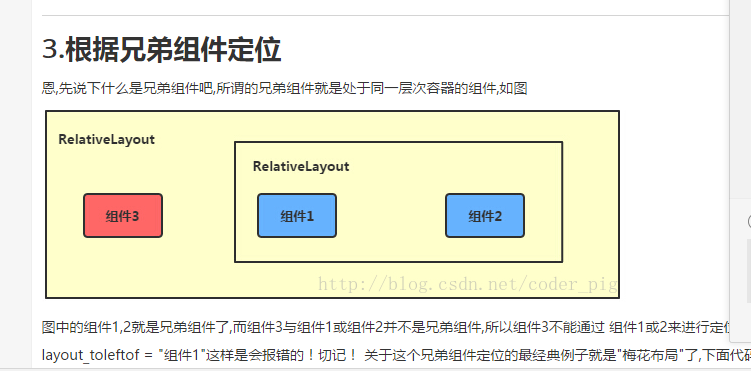
**The value for each layout property is either a boolean to enable a layout position relative to the parent**[**RelativeLayout**](https://developer.android.com/reference/android/widget/RelativeLayout.html)**or an ID that references another view in the layout against which the view should be positioned.**

**http://www.runoob.com/w3cnote/android-tutorial-relativelayout.html**



## 父容器定位属性示意图





## margin与padding的区别

首先margin代表的是偏移,比如marginleft = "5dp"表示组件离容器左边缘偏移5dp; 而padding代表的则是填充,而填充的对象针对的是组件中的元素,比如TextView中的文字 比如为TextView设置paddingleft = "5dp",则是在组件里的元素的左边填充5dp的空间！ **margin针对的是容器中的组件，而padding针对的是组件中的元素，**要区分开来！ 下面通过简单的代码演示两者的区别:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity" >

<Button

android:id="@+id/btn1"

android:layout\_height="wrap\_content"

android:layout\_width="wrap\_content"

android:text="Button"/>

<Button

android:paddingLeft="100dp"

android:layout\_height="wrap\_content"

android:layout\_width="wrap\_content"

android:text="Button"

android:layout\_toRightOf="@id/btn1"/>

<Button

android:id="@+id/btn2"

android:layout\_height="wrap\_content"

android:layout\_width="wrap\_content"

android:text="Button"

android:layout\_alignParentBottom="true"/>

<Button

android:layout\_marginLeft="100dp"

android:layout\_height="wrap\_content"

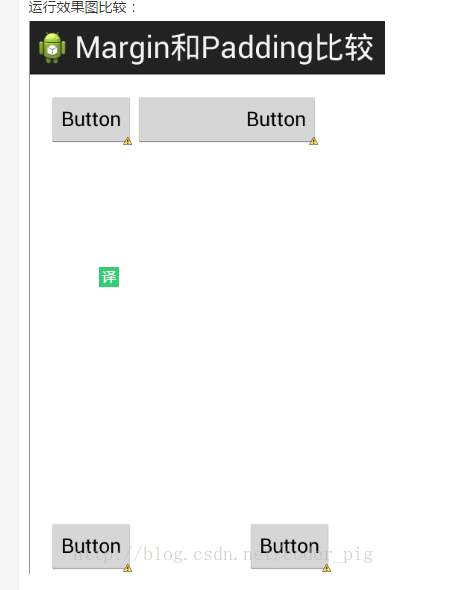
android:layout\_width="wrap\_content"

android:text="Button"

android:layout\_toRightOf="@id/btn2"

android:layout\_alignParentBottom="true"/>

</RelativeLayout>



**从上面可以看出，margin就是和相对定位的组件，可能是他爹，也可能是兄弟去争地方，padding就是和自己老婆争地方。**