# 屏幕适配


三星
其他
小米
联想
步步高
中兴
酷派
HTC
海信
天语
魅族
摩托罗拉
华为
索尼
朵唯
波导
康佳
欧博信
OPPO
海尔
尼彩
LG
金立
亿通
长虹
语信
TCL
昂达
夏新
欧奇

# 适配优化

## ☑ 像素（px）

- 含义：通常所说的像素，就是CCD/CMOS上光电感应元件的数量，一个感光元件经过感光，光电信号转换，A/D转换等步骤以后，在输出的照片上就形成一个点，我们如果把影像放大数倍，会发现这些连续色调其实是由许多色彩相近的小方点所组成，这些小方点就是构成影像的最小单位“像素”（Pixel）。简而言之，像素就是手机屏幕的最小构成单元。
- 单位：px（pixel），1px = 1像素点
- 一般情况下UI设计师的设计图会以px作为统一的计量单位。
- 绝对单位 1px 全天下所有的1px都是一样大的
- 相对：1%

# 适配优化

## ☑ 分辨率

- 含义：手机在横向、纵向上的像素点数总和
- 一般描述成 宽*高 ，即横向像素点个数 * 纵向像素点个数（如1080 x 1920）。
- 单位：px（pixel），1px = 1像素点

## ☑ 屏幕尺寸（in）

- 含义：手机对角线的物理尺寸
- 单位 英寸（inch），一英寸大约2.54cm
- 常见的尺寸有4.7寸、5寸、5.5寸、6寸

# 适配优化

## ☑ 屏幕像素密度（dpi） vs dip

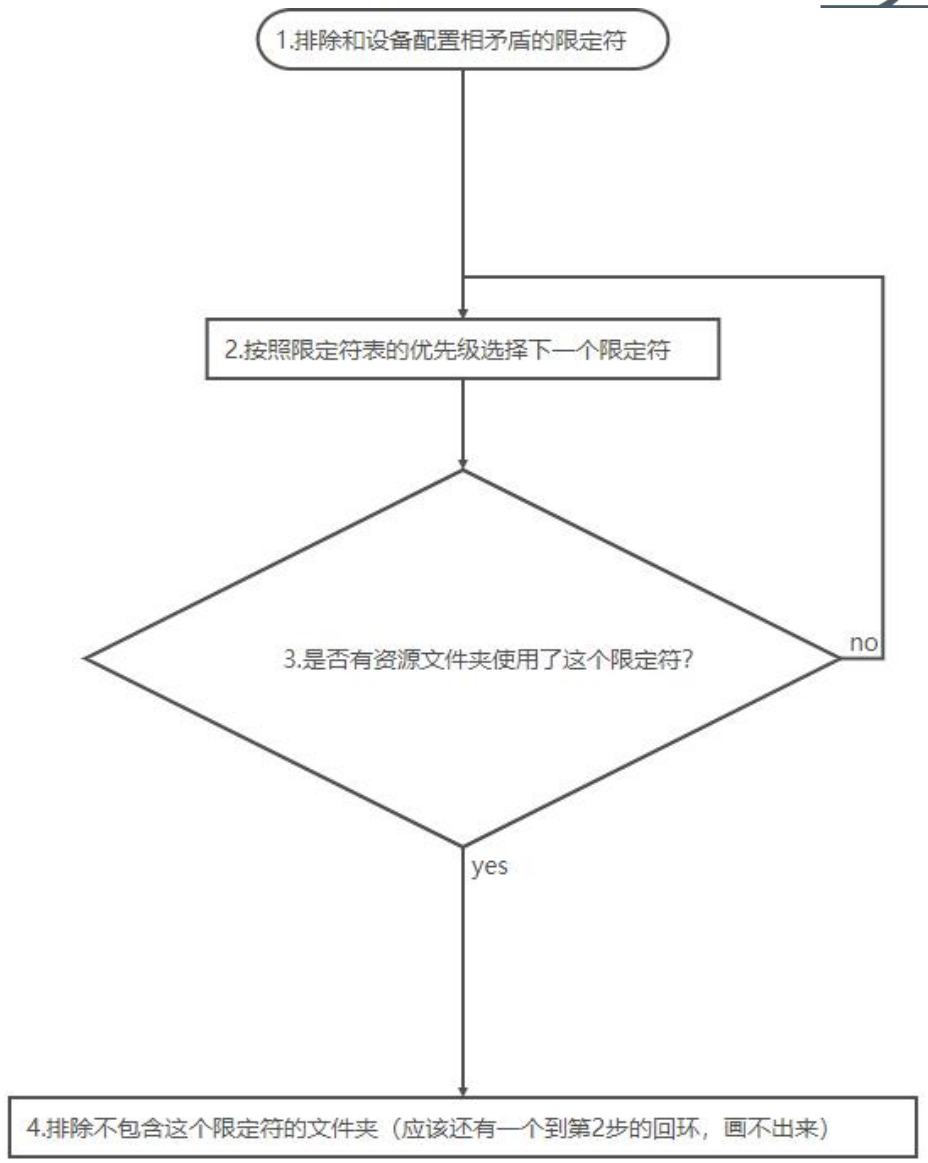
- 含义：每英寸的像素点数。
- 例如每英寸内有160个像素点，则其像素密度为160dpi。
- 单位：dpi（dots per inch）
- 计算公式： 像素密度 = 像素 / 尺寸 （dpi = px / in）
- 标准屏幕像素密度（mdpi）： 每英寸长度上还有160个像素点（160dpi），即称为标准屏幕像素密度（mdpi）。

# 适配优化

| 密度类型 | 分辨率(px) | 屏幕像素密度(dpi) | density |
| --- | --- | --- | --- |
| 低密度（ldpi） | 240 x 320 | 0~120 | 0.75 |
| 中密度（mdpi） | 320 x 480 | 120~160 | 1 |
| 高密度（hdpi） | 480 x 800 | 160~240 | 1.5 |
| 超高密度（xhdpi） | 720 x 1280 | 240~320 | 2 |
| 超超高密度（xxhdpi） | 1080 x 1920 | 320~480 | 3 |

# 适配优化

- xlarge screens are at least 960dp x 720dp
- large screens are at least 640dp x 480dp
- normal screens are at least 470dp x 320dp
- small screens are at least 426dp x 320dp



# 适配优化

## 屏幕尺寸、分辨率、像素密度三者关系

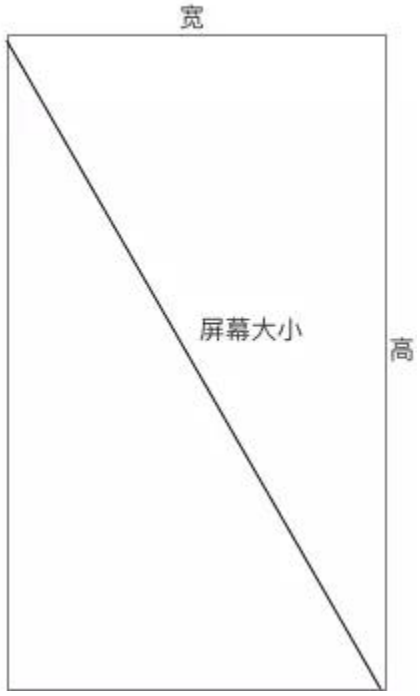
$$密度（单位/dpi）= \frac{\sqrt{（宽^2+高^2）}\ 单位（/px）}{屏幕大小\quad 单位（/inch）}$$

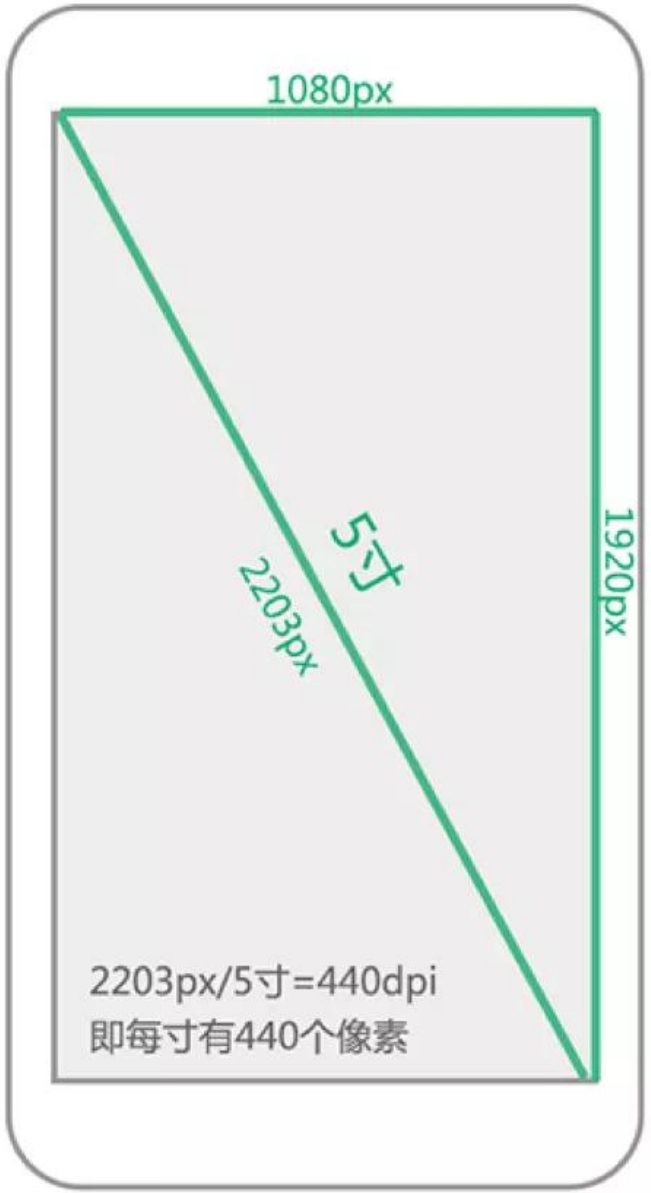
讲解

1. 密度即每英寸的像素点

2. 勾股定理求出手机的对角线物理尺寸

3. 再除以屏幕大小即可



# 适配优化

## ☑ 密度无关像素（dp） vs dpi

- 含义：density-independent pixel，叫dp或dip，与终端上的实际物理像素点无关
- 单位：dp，可以保证在不同屏幕像素密度的设备上显示相同的效果，是安卓特有的长度单位。
- 场景例子：假如同样都是画一条长度是屏幕一半的线，如果使用px作为计量单位，那么在480x800分辨率手机上设置应为240px；在320x480的手机上应设置为160px，二者设置就不同了；如果使用dp为单位，在这两种分辨率下，160dp都显示为屏幕一半的长度。
- dp与px的转换：px = density * dp
- px = dp * (dpi / 160) 宽高比 density =dpi / 160
- 320* 480 160dp =160px 1dp = 1px

# 适配优化

| 密度类型 | 代表的分辨率(px) | 屏幕密度（dpi） | 换算 |
| --- | --- | --- | --- |
| 低密度(ldpi) | 240 x 320 | 120 | 1dp = 0.75px |
| 中密度(mdpi) | 320 x 480 | 160 | 1dp=1px |
| 高密度(hdpi) | 480 x 800 | 240 | 1dp=1.5px |
| 超高密度(xhdpi) | 720 x 1280 | 320 | 1dp=2px |
| 超超高高密(xxhdpi) | 1080 x 1920 | 480 | 1dp=3px |

# 适配优化

## ☑ 独立比例像素（sp）

- 含义：scale-independent pixel，叫sp或sip
- 单位：sp，字体大小专用单位
- Android开发时用此单位设置文字大小，可根据字体大小首选项进行缩放；
- 推荐使用12sp、14sp、18sp、22sp作为字体大小，不推荐使用奇数和小数，容易造成精度丢失，12sp以下字体太小

# ☑sp 与 dp 的区别

- dp只跟屏幕的像素密度有关；
- sp和dp很类似但唯一的区别是，Android系统允许用户自定义文字尺寸大小（小、正常、大、超大等等），当文字尺寸是“正常”时1sp=1dp=0.00625英寸，而当文字尺寸是“大”或“超大”时，1sp>1dp=0.00625英寸。类似我们在windows里调整字体尺寸以后的效果——窗口大小不变，只有文字大小改变。

# 适配优化

```
//第一种
DisplayMetrics metrics = new DisplayMetrics();
Display display = getWindowManager().getDefaultDisplay();
display.getMetrics(metrics);

//第二种
DisplayMetrics metrics1 = getResources().getDisplayMetrics();

//第三种
DisplayMetrics metrics2 = Resources.getSystem().getDisplayMetrics();
```

# 适配优化

```
/**
 * Converts an unpacked complex data value holding a dimension to its final floating
 * point value. The two parameters <var>unit</var> and <var>value</var>
 * are as in {@link #TYPE_DIMENSION}.
 *
 * @param unit The unit to convert from.
 * @param value The value to apply the unit to.
 * @param metrics Current display metrics to use in the conversion --
 *                supplies display density and scaling information.
 *
 * @return The complex floating point value multiplied by the appropriate
 * metrics depending on its unit.
 */
public static float applyDimension(int unit, float value,
                                   DisplayMetrics metrics)
{
    switch (unit) {
    case COMPLEX_UNIT_PX:
        return value;
    case COMPLEX_UNIT_DIP:
        return value * metrics.density;
    case COMPLEX_UNIT_SP:
        return value * metrics.scaledDensity;
    case COMPLEX_UNIT_PT:
        return value * metrics.xdpi * (1.0f/72);
    case COMPLEX_UNIT_IN:
        return value * metrics.xdpi;
    case COMPLEX_UNIT_MM:
        return value * metrics.xdpi * (1.0f/25.4f);
    }
    return 0;
}
```

# 适配优化

## ☑ 适配方案

- 布局组件的适配
- 布局的适配
- 代码适配 接口适配: 加载图片的时候

# 适配优化

## ☑ 布局组件的适配

- 使用密度无关像素指定尺寸dp
- 使用相对布局或线性布局，不要使用绝对布局
- 使用wrap_content、match_parent、权重
- 使用minWidth、minHeight、lines等属性
- dimens使用

# 适配优化

## ☑ 布局的适配

- 使用Size限定符
- 最小宽度限定符
- 使用布局别名
- 使用屏幕方向限定符
- 多套layout适配

# 适配优化

## ☑ 图片的适配

- LOGO 图标

| 屏幕密度 | 对应的图片大小 | 图片资源目录 |
| --- | --- | --- |
| 120dip | 36px * 36px | mipmap-ldpi |
| 160dip(基准) | 48px * 48px | mipmap或者mipmap-mdpi |
| 240dip(1.5倍) | 72px * 72px | mipmap-hdpi |
| 320dip (2倍) | 96px * 96px | mipmap-xhdpi |
| 480dip (3倍) | 144px * 144px | mipmap-xxhdpi |
| 640dip (4倍) | 192px * 192px | mipmap-xxxhdpi |

# 适配优化

## ☑ 图片的适配

- 普通图片和图标
- 自动拉伸位图：Nine-Patch的图片类型
- 动画、自定义view、shape

# 适配优化

## ☑ ImageView的ScaleType适配

- android:scaleType=“center” 保持原图的大小，显示在ImageView的中心。当原图的size大于ImageView的size时，多出来的部分被截掉
- android:scaleType=“center_inside”
- 以原图正常显示为目的，如果原图大小大于ImageView的size，就按照比例缩小原图的宽高，居中显示在ImageView中。如果原图size小于ImageView的size，则不做处理居中显示图片
- android:scaleType=“center_crop”
- 以原图填满ImageView为目的，如果原图size大于ImageView的size，则与center_inside一样，按比例缩小，居中显示在ImageView上。如果原图size小于ImageView的size，则按比例拉升原图的宽和高，填充ImageView居中显示
- android:scaleType=“matrix” 不改变原图的大小，从ImageView的左上角开始绘制，超出部分做剪切处理
- androd:scaleType=“fit_xy” 把图片按照指定的大小在ImageView中显示，拉伸显示图片，不保持原比例，填满ImageView.

# 属性介绍

## ConstraintLayout本身的基本属性

```
android:minHeight="200dp"
android:minWidth="150dp"
android:maxHeight="500dp"
android:maxWidth="300dp"
```

控制 View 最大尺寸和最小尺寸的属性，可以设置到 ConstraintLayout 上来控制 ConstraintLayout 的尺寸信息

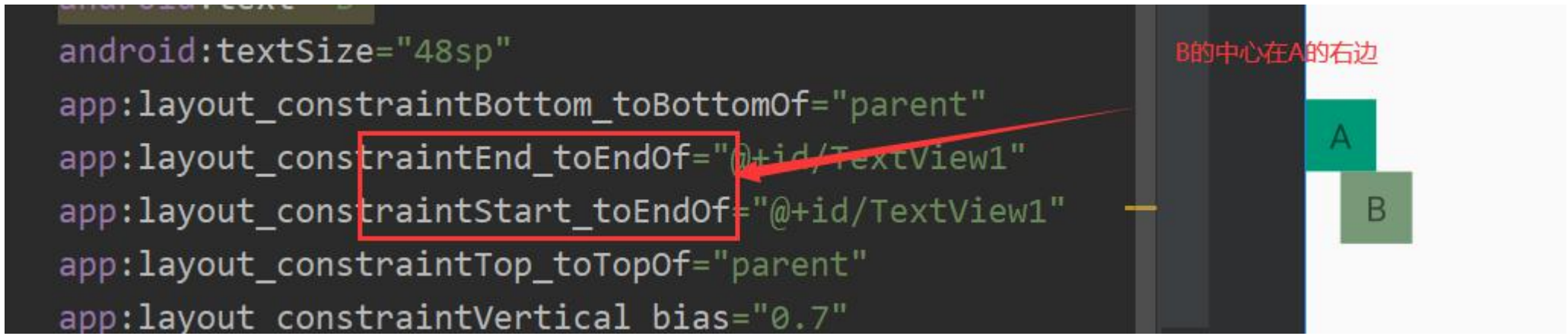
# 属性介绍

## ConstraintLayout相对定位属性

```
app:layout_constraintBaseline_toBaselineOf="parent"
```

# 属性介绍

```
app:layout_constraintStart_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintLeft_toRightOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintEnd_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintRight_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
```


android:textSize="48sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="@+id/TextView1"
app:layout_constraintStart_toEndOf="@+id/TextView1"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.7"
B的中心在A的右边
A
B

# 属性介绍

## ConstraintLayout的边距

```
android:layout_margin="0dp"
android:layout_marginStart="0dp"
android:layout_marginLeft="0dp"
android:layout_marginTop="0dp"
android:layout_marginEnd="0dp"
android:layout_marginRight="0dp"
android:layout_marginBottom="0dp"
```

```
app:layout_goneMarginStart="0dp"
app:layout_goneMarginLeft="0dp"
app:layout_goneMarginTop="0dp"
app:layout_goneMarginEnd="0dp"
app:layout_goneMarginRight="0dp"
app:layout_goneMarginBottom="0dp"
```

# 属性介绍

## ConstraintLayout的居中和偏移(bias)

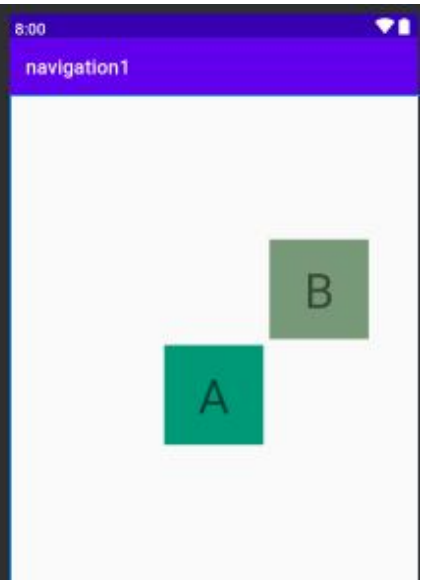
```
app:layout_constraintHorizontal_bias="0"
app:layout_constraintVertical_bias="0"
```

# 属性介绍

```
android:id="@+id/TextView2"
android:layout_width="100dp"
android:layout_height="100dp"
app:layout_constraintCircle="@+id/TextView1"
app:layout_constraintCircleAngle="45"
app:layout_constraintCircleRadius="150dp"
android:background="#797"
android:gravity="center"
android:text="B"
```

0-360度

半径



# 属性介绍

## ConstraintLayout的子View的尺寸控制


- 使用确定的尺寸，比如 48dp


- 使用 WRAP_CONTENT ，和其他地方的 WRAP_CONTENT 一样


- 使用 0dp，这个选项等于 “MATCH_CONSTRAINT”，也就是和约束规则指定的宽(高)度一样
  - MATCH_PARENT 属性无法在 ConstraintLayout 里面的 子 View 上使用

# 属性介绍

## ConstraintLayout控制子View的宽高比

```
app:layout_constraintDimensionRatio="5:1"
```

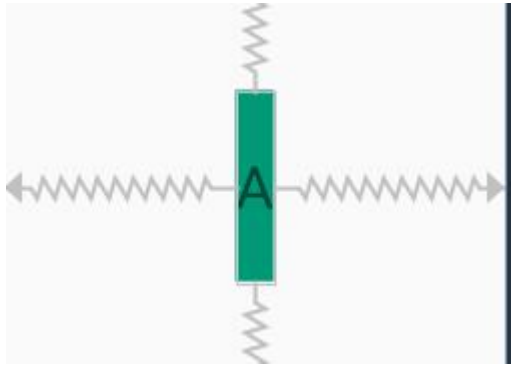
layout_constraintDimensionRatio 控制子View的宽高比
除了上面三种设置 子 View 的尺寸以外，还可以控制 子 View 的宽高比。如果要使用宽高比则需要至少设置一个尺寸约束为 0dp，然后设置 layout_constraintDimentionRatio 属性

- float 值，代表宽度/高度 的比率
- “宽度:高度”这种比率值

# 属性介绍

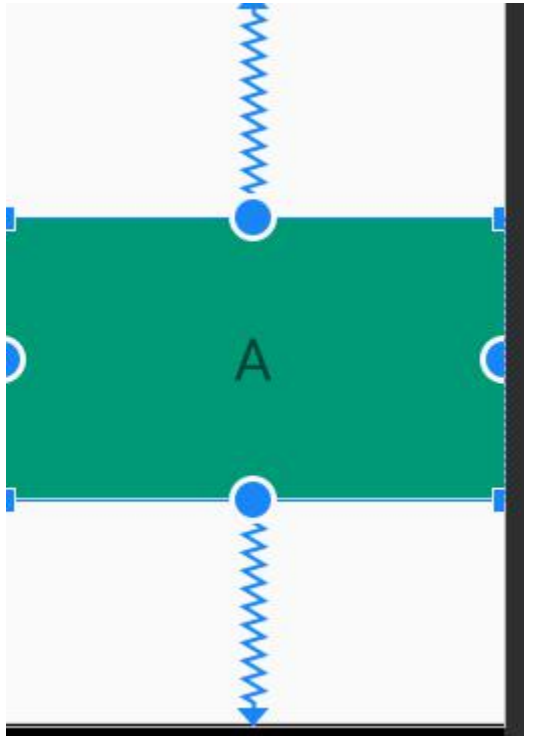
```
android:layout_width="wrap_content"
android:layout_height="0dp"
app:layout_constraintDimensionRatio="1:5"
```

以wrap_content的边为基准




```
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintDimensionRatio="h,16:9"
```

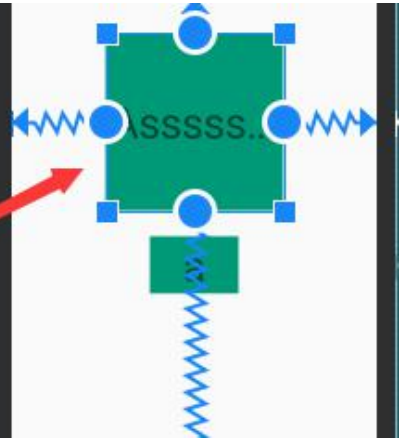
h表示约束高度
w表示约束宽度



# 属性介绍

```
android:id="@+id/TextView1"
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintWidth_max="200dp"
app:layout_constraintHeight_max="200dp"
android:text="Asssssssssssssss"
```

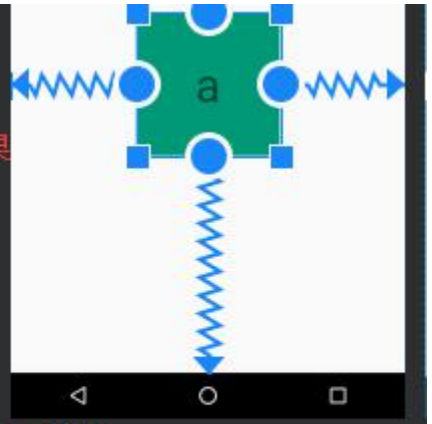
max约束只在0dp模式下有作用



```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:layout_constraintWidth_min="150dp"
app:layout_constraintHeight_min="150dp"
app:layout_constrainedWidth="true"
app:layout_constrainedHeight="true"
android:text="a"
```

min只在wrap_content模式下有效果

并且要求constraintedWidth/Height = true
后续版本可能不需要设置这个属性




如果改为这个时候layout_width=“wrap_content,这constraintedWidth必须设置为true”

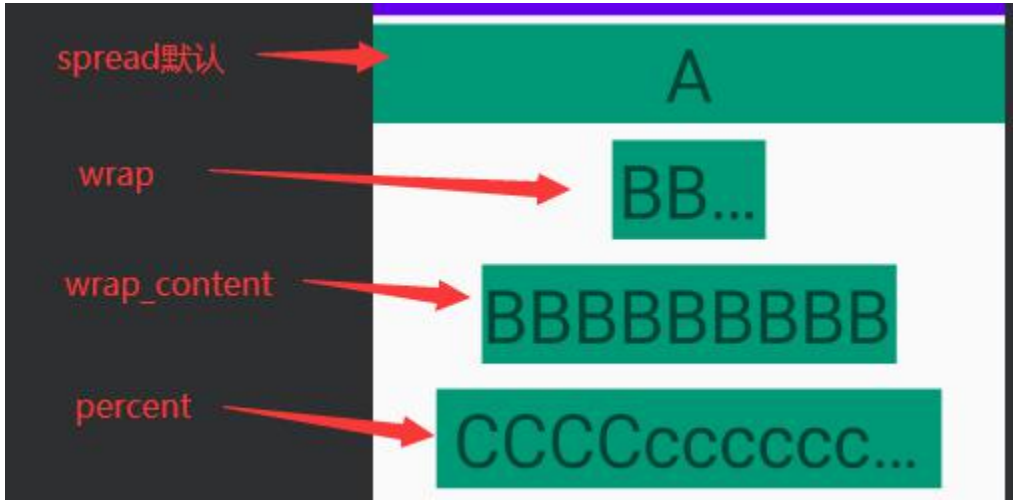
# 属性介绍

```
android:layout_width="0dp"
android:layout_height="wrap_content"
app:layout_constraintWidth_default="spread"
```

默认

```
android:layout_width="0dp"
android:layout_height="wrap_content"
app:layout_constraintWidth_default="percent"
app:layout_constraintWidth_percent="0.8"
```

配合使用




spread：尽可能扩展视图以满足每一方的约束。这是默认行为。
wrap：仅根据需要扩展视图以适应其内容，但仍允许视图小于约束所需的视图。因此，使用Wrap Content（上图）之间的区别在于，将宽度设置为Wrap Content会强制宽度始终与内容宽度完全匹配；而使用Match Constraints和layout_constraintWidth_default设置为wrap也允许视图小于内容宽度。
percent：设置View的宽度为parent的比例值，比例值默认是100%，即宽度是match_parent。这个比例值通过属性app:layout_constraintWidth_percent设置

## Chains

Chains provide group-like behavior in a single axis (horizontally or vertically). The other axis can be constrained independently.

### Creating a chain

A set of widgets are considered a chain if they are linked together via a bi-directional connection (see Fig. 9, showing a minimal chain, with two widgets).

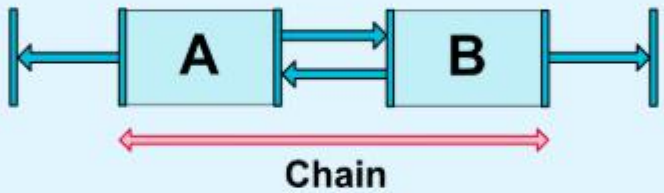



*Fig. 9 - Chain*

### Chain heads

Chains are controlled by attributes set on the first element of the chain (the "head" of the chain):

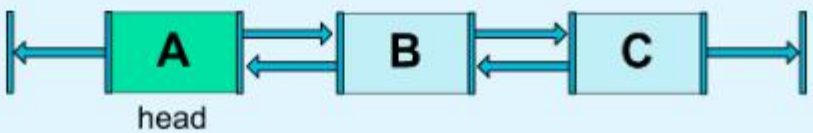


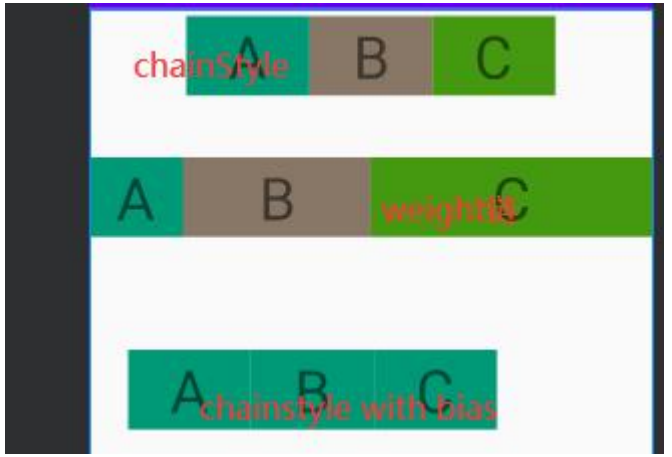

*Fig. 10 - Chain Head*

# 属性介绍

## ConstraintLayout的链条布局

```
android:layout_width="100dp"
android:layout_height="wrap_content"
app:layout_constraintHorizontal_chainStyle="packed"
app:layout_constraintHorizontal_weight="1"
```

```
app:layout_constraintHorizontal_chainStyle="packed"
app:layout_constraintHorizontal_bias="0.1"
```




chainStyle:

spread

spread_inside

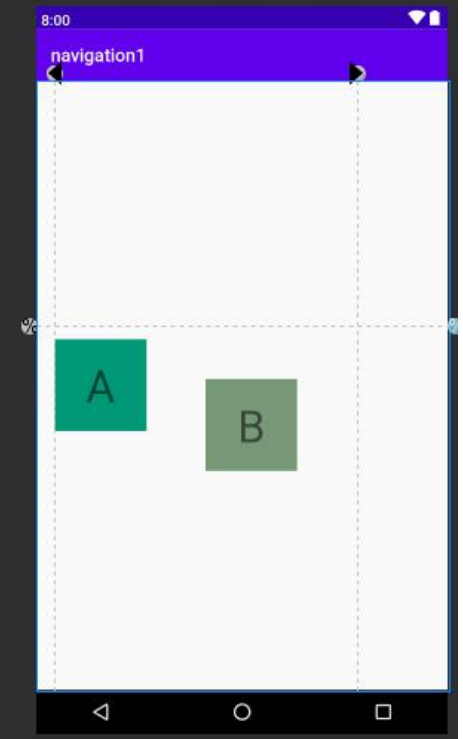
packed

# 属性介绍

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline1"
    android:orientation="vertical"
    app:layout_constraintGuide_begin="20dp"
    android:layout_width="0dp"
    android:layout_height="0dp"/>
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline2"
    android:orientation="vertical"
    app:layout_constraintGuide_end="100dp"
    android:layout_width="0dp"
    android:layout_height="0dp"/>

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline3"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.4"
    android:layout_width="0dp"
```

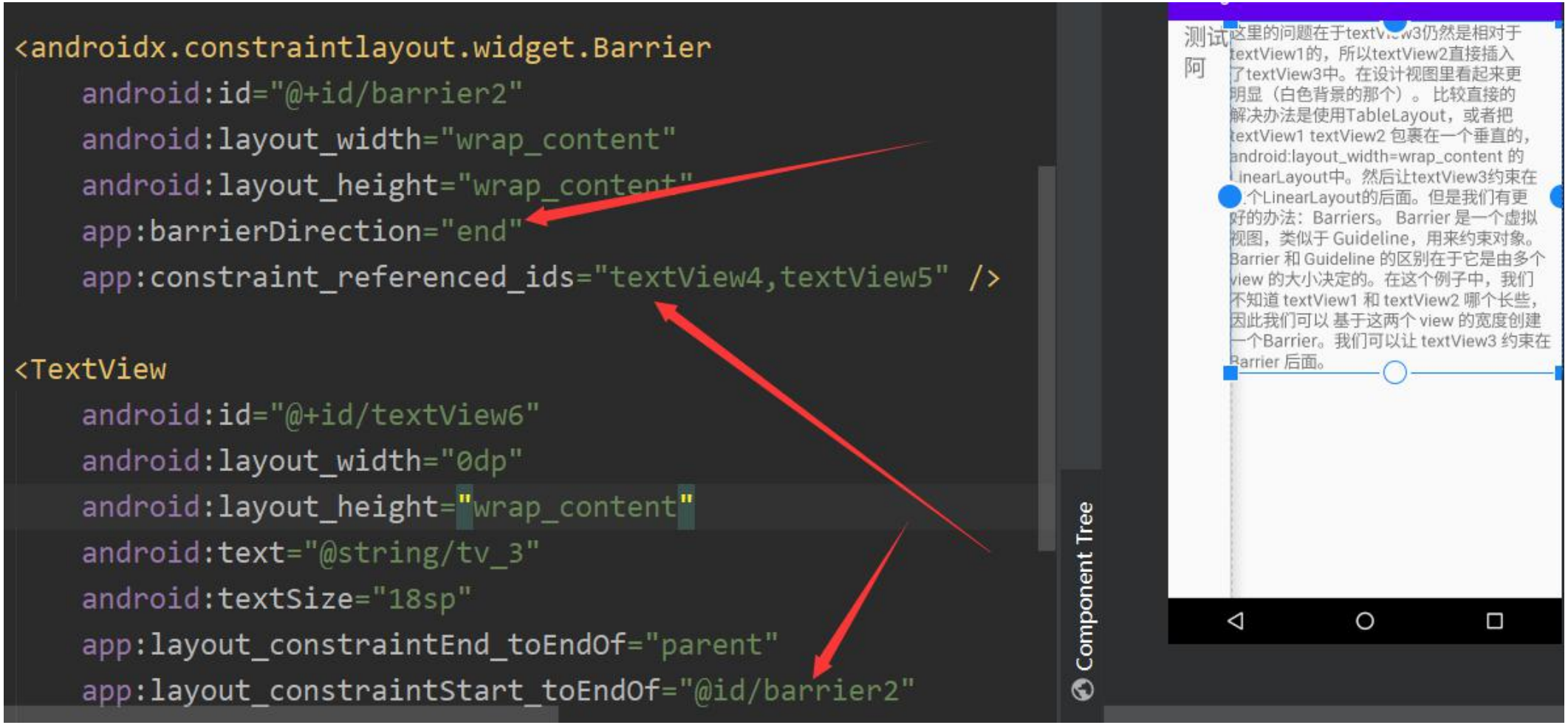
百分比，可以用来划分布局区域



# 属性介绍

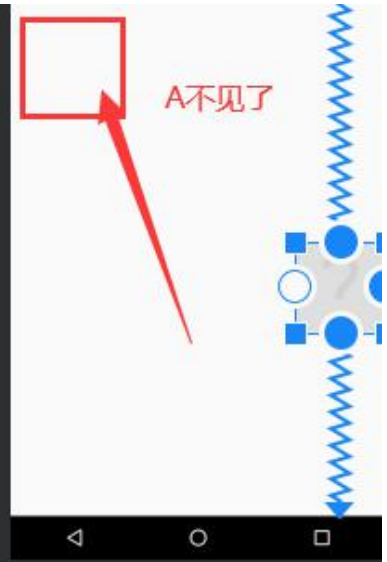
```
<androidx.constraintlayout.widget.Barrier
    android:id="@+id/barrier2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:barrierDirection="end"
    app:constraint_referenced_ids="textView4,textView5" />

<TextView
    android:id="@+id/textView6"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="@string/tv_3"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@id/barrier2"
```



```
<androidx.constraintlayout.widget.Group
    android:id="@+id/group1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:visibility="gone"
    app:constraint_referenced_ids="textView4,textView5"
```

```
<androidx.constraintlayout.widget.Placeholder
    android:id="@+id/placehorder"
    android:layout_width="100dp"
    app:content="@+id/TextView1"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.6"
    android:layout_height="100dp"/>
```



# 属性介绍

## UI编辑器所使用的属性

```
app:layout_optimizationLevel="none"
app:layout_editor_absoluteX="0dp"
app:layout_editor_absoluteY="0dp"
app:layout_constraintBaseline_creator="0"
app:layout_constraintLeft_creator="0"
app:layout_constraintTop_creator="0"
app:layout_constraintRight_creator="0"
app:layout_constraintBottom_creator="0"
```