前言

> **ConstraintLayout** 是一个使用“相对定位”灵活地确定微件的位置和大小的一个布局，在 2016 年 **Google I/O** 中面世，它的出现是为了解决开发中过于复杂的页面层级嵌套过多的问题——层级过深会增加绘制界面需要的时间，影响用户体验，以灵活的方式定位和调整小部件。从 **Android Studio 2.3**起，创建**layout**文件就已经是默认**ConstraintLayout**了，但是尽管**Google**如此大力推这项技术，但在当时很少有人使用，近些年逐渐被大家拿起来，啊真香！（此处无图胜有图）。目前**ConstraintLayout**正式版已经更新至**2.0.4**，本文将带领大家熟悉**ConstraintLayout**全部内容。

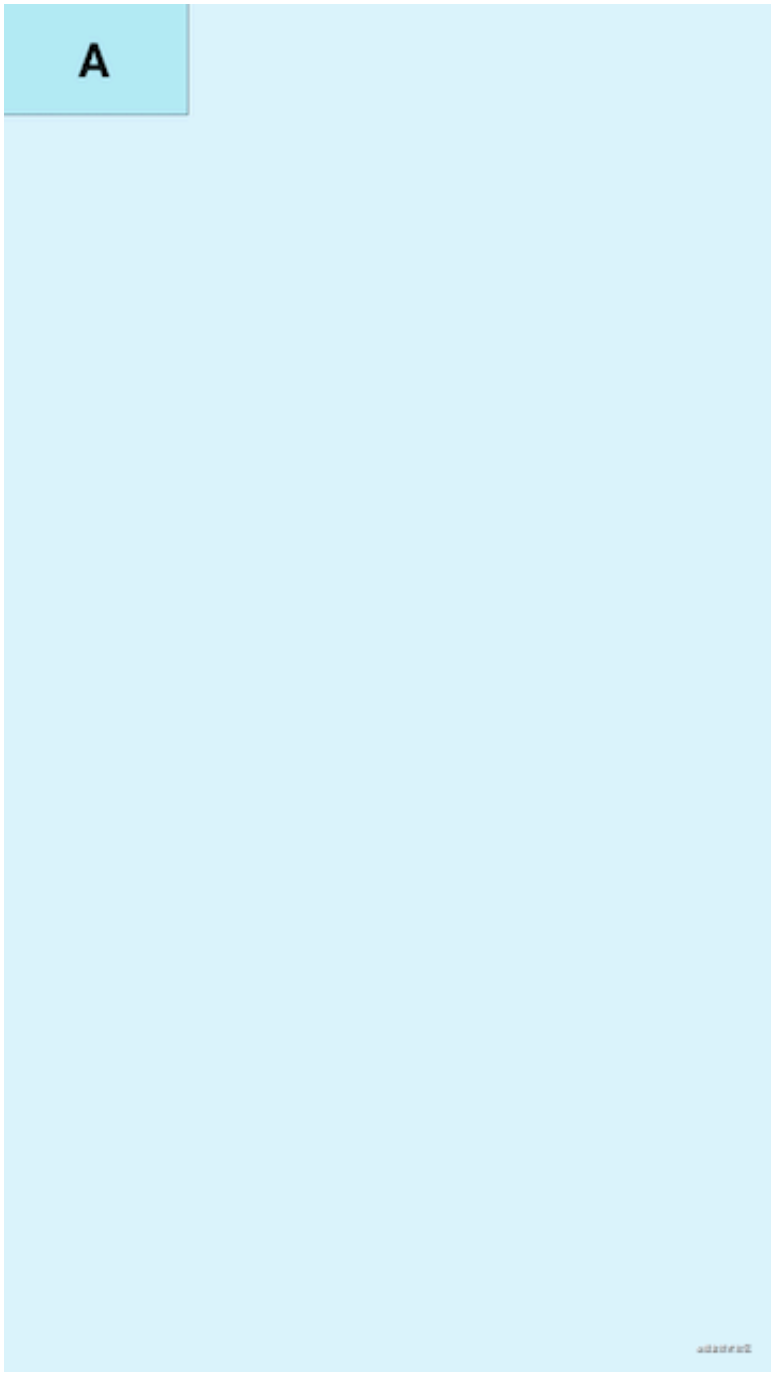
一. 布局的使用

1.1 位置约束

> `ConstraintLayout` 采用方向约束的方式对控件进行定位，至少要保证水平和垂直方向都至少有一个约束才能确定控件的位置

1.1.1 基本方向约束

> 比如我们想实现这个位置，顶部和界面顶部对齐，左部和界面左部对齐：



> 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity">
```

```
9
10      <TextView
11          android:layout_width="100dp"
12          android:layout_height="60dp"
13          android:background="@drawable/tv_bg"
14          android:gravity="center"
15          android:text="A"
16          android:textColor="@color/black"
17          android:textSize="25sp"
18          android:textStyle="bold"
19          app:layout_constraintStart_toStartOf="parent"
20          app:layout_constraintTop_toTopOf="parent"
21          tools:ignore="HardcodedText" />
22
23  </androidx.constraintlayout.widget.ConstraintLayout>
```

> 核心代码是这两行：

```
1  app:layout_constraintStart_toStartOf="parent"
2  app:layout_constraintTop_toTopOf="parent"
```

> 这两行代码的意思就是,控件的开始方向与父容器的开始方向对齐，控件的顶部方向与父容器的顶部方向对齐，其实 `layout_constraintStart_toStartOf` 也可以使用 `layout_constraintLeft_toLeftOf` ，但是使用 `start` 和 `end` 来表示左和右是为了考虑别的国家的习惯，有的国家开始方向是右，所以使用 `start` 和 `end` 可以兼容这种情况。到这里就可以看到该控件使用 `layout_constraintStart_toStartOf` 和 `layout_constraintTop_toTopOf` 两条约束确定了自己的位置，这里有一个使用技巧，就是，该控件的？？方向在哪个控件的？？方向，记住这一点就可以了。那么下面就介绍下全部的约束属性：

```
<!-- 基本方向约束 -->
<!-- 我的什么位置在谁的什么位置 -->
app:layout_constraintTop_toTopOf=""              我的顶部和谁的顶部对齐
app:layout_constraintBottom_toBottomOf=""        我的底部和谁的底部对齐
app:layout_constraintLeft_toLeftOf=""            我的左边和谁的左边对齐
app:layout_constraintRight_toRightOf=""          我的右边和谁的右边对齐
app:layout_constraintStart_toStartOf=""          我的开始位置和谁的开始位置对齐
app:layout_constraintEnd_toEndOf=""              我的结束位置和谁的结束位置对齐

app:layout_constraintTop_toBottomOf=""           我的顶部位置在谁的底部位置
app:layout_constraintStart_toEndOf=""            我的开始位置在谁的结束为止
<!-- ...以此类推 -->
```

> 那么 `ConstraintLayout` 就是使用这些属性来确定控件的位置，虽然比较多，但是有规律可循，没有任何记忆压力

1.1.2 基线对齐

> 我们看一个场景：

> 我们有时候需要写这样的需求：两个文本是基线对齐的，那就可以用到我们的一个属性 `layout_constraintBaseline_toBaselineOf` 来实现，它的意思就是这个控件的基线与谁的基线对齐，代码如下：

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="20"
        android:textColor="@color/black"
        android:textSize="50sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tv2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="¥"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintBaseline_toBaselineOf="@id/tv1"
        app:layout_constraintStart_toEndOf="@id/tv1" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

> 通过 `layout_constraintBaseline_toBaselineOf` 我们就可以让两个不同大小的文案基线对齐

1.1.3 角度约束

> 有些时候我们需要一个控件在某个控件的某个角度的位置，那么通过其他的布局其实是不太好实现的，但是 `ConstraintLayout` 为我们提供了角度位置相关的属性

```
app:layout_constraintCircle=""          目标控件id
app:layout_constraintCircleAngle=""     对于目标的角度(0-360)
app:layout_constraintCircleRadius=""    到目标中心的距离
```

> 我们来实现一下下图的UI，jetpack图标在android图标的45度方向，距离为60dp

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

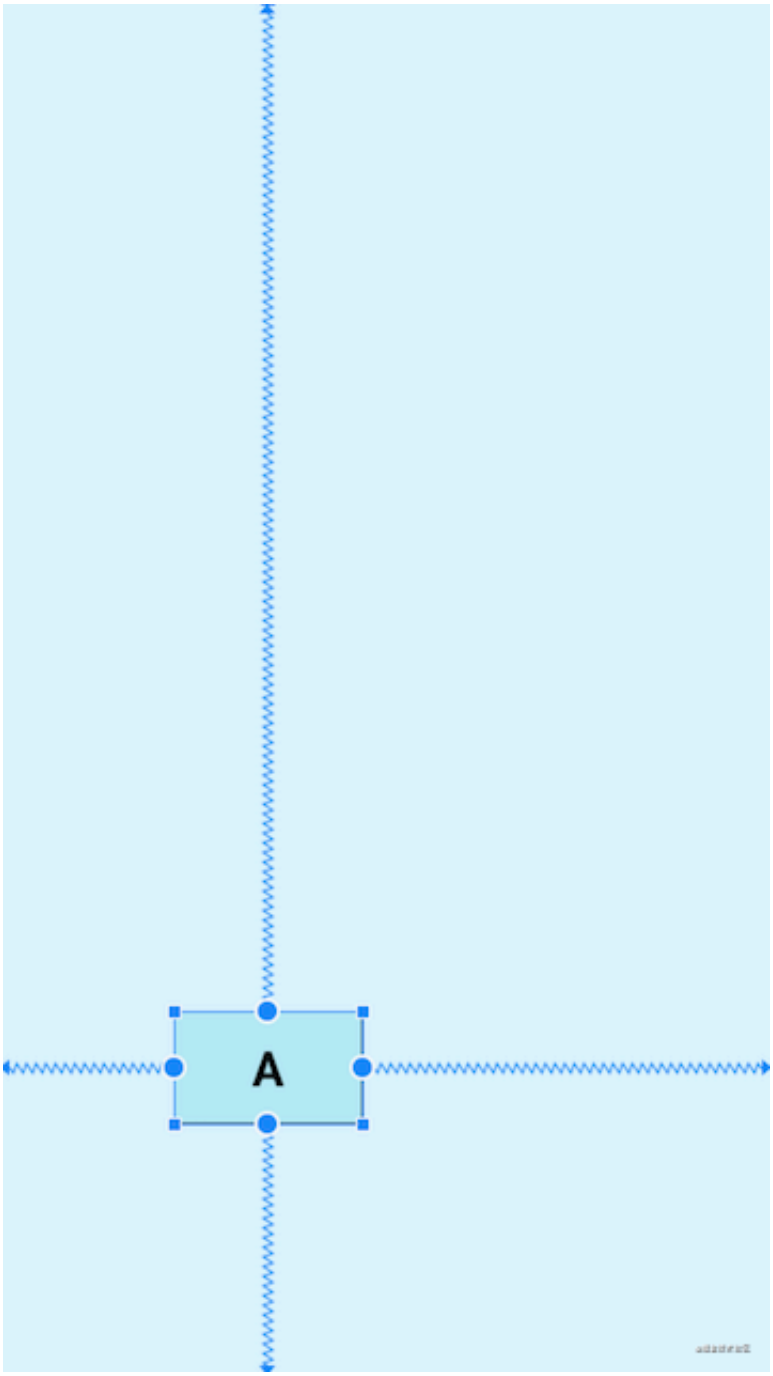
```
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <ImageView
        android:id="@+id/android"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/android"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/jetpack"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:src="@drawable/jetpack"
        app:layout_constraintCircle="@+id/android"
        app:layout_constraintCircleAngle="45"
        app:layout_constraintCircleRadius="70dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.1.4 百分比偏移

> 有的时候我们需要让控件在父布局的水平方向或垂直方向的百分之多少的位置，可以使用如下属性：

```
app:layout_constraintHorizontal_bias=""    水平偏移 取值范围是0-1的小数
app:layout_constraintVertical_bias=""      垂直偏移 取值范围是0-1的小数
```

> 示例：控件A在父布局水平方向偏移0.3（30%），垂直方向偏移0.8（80%）
> **注意**：在使用百分比偏移时，需要指定对应位置的约束条件




```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
```

```
        android:layout_width="100dp"
        android:layout_height="60dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.3"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.8" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.2 控件内边距、外边距、GONE Margin

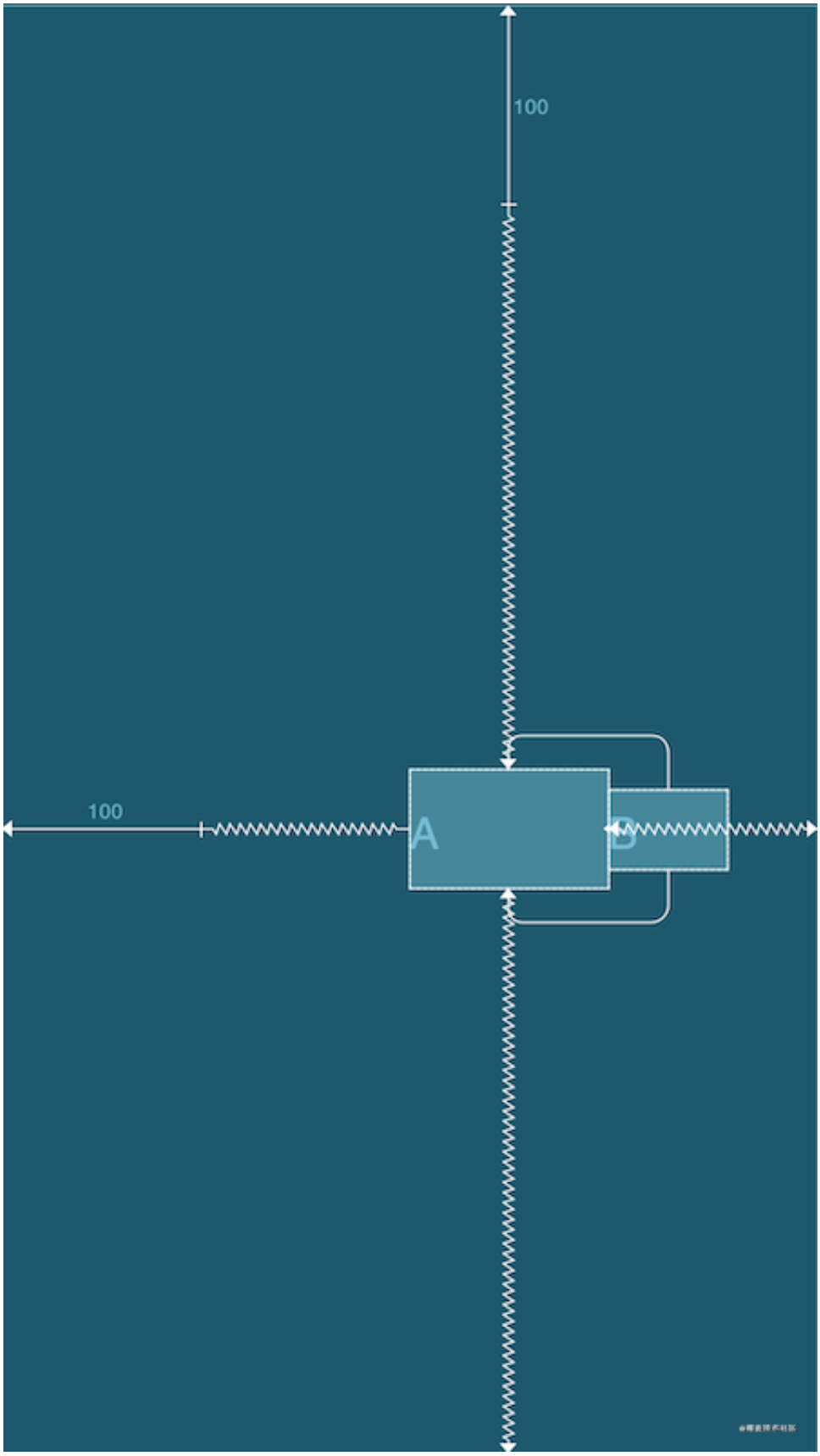
> `ConstraintLayout` 的内边距和外边距的使用方式其实是和其他布局一致的

```
<!--  外边距  -->
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"

<!--  内边距  -->
android:padding="0dp"
android:paddingStart="0dp"
android:paddingLeft="0dp"
android:paddingTop="0dp"
android:paddingEnd="0dp"
android:paddingRight="0dp"
android:paddingBottom="0dp"
```

`ConstraintLayout`除此之外还有`GONE Margin`，当依赖的目标`view`隐藏时会生效的属性，例如B被A依赖约束，当B隐藏时B会缩成一个点，自身的`margin`效果失效，A设置的`GONE Margin`就会生效，属性如下：

```
<!--  GONE Margin  -->
app:layout_goneMarginBottom="0dp"
app:layout_goneMarginEnd="0dp"
app:layout_goneMarginLeft="0dp"
app:layout_goneMarginRight="0dp"
app:layout_goneMarginStart="0dp"
app:layout_goneMarginTop="0dp"
```

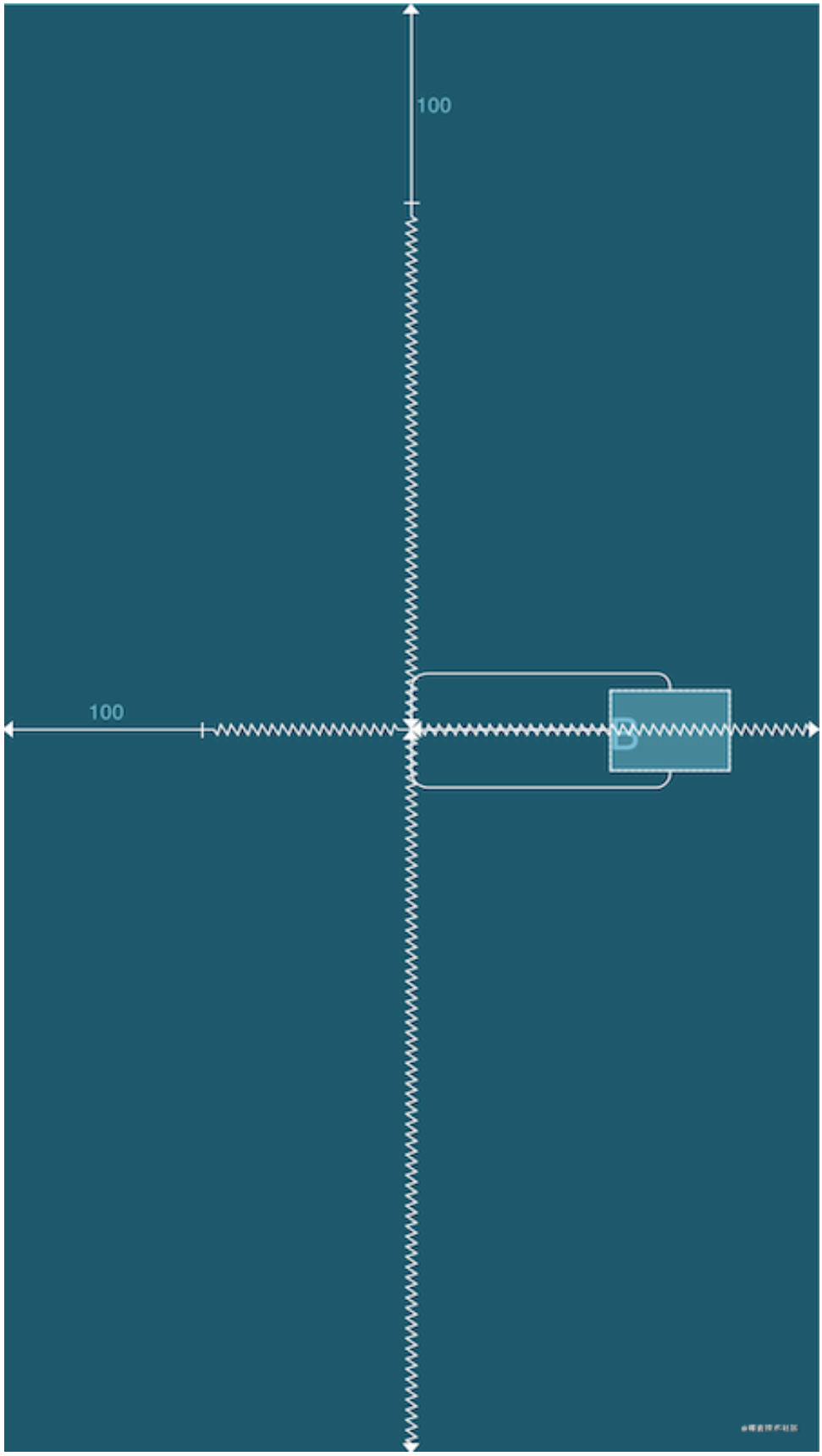
示例：当目标控件是显示的时候`GONE Margin`不会生效




```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="#DAF3FE"
8       tools:context=".MainActivity"
9       tools:ignore="HardcodedText">
10
11      <TextView
12          android:id="@+id/A"
13          android:layout_width="100dp"
14          android:layout_height="60dp"
15          android:layout_marginStart="100dp"
16          android:layout_marginTop="100dp"
17          android:background="@drawable/tv_bg"
18          android:gravity="center"
19          android:text="A"
20          android:textColor="@color/black"
21          android:textSize="25sp"
22          android:textStyle="bold"
23          app:layout_constraintBottom_toBottomOf="parent"
24          app:layout_constraintEnd_toEndOf="parent"
25          app:layout_constraintStart_toStartOf="parent"
26          app:layout_constraintTop_toTopOf="parent" />
27
28      <!--  该控件设置了 layout_goneMarginStart="100dp" 当A控件隐藏时才会生效  -->
29      <TextView
30          android:id="@+id/B"
31          android:layout_width="60dp"
32          android:layout_height="40dp"
33          android:background="@drawable/tv_bg"
34          android:gravity="center"
35          android:text="B"
36          android:textColor="@color/black"
37          android:textSize="25sp"
38          android:textStyle="bold"
39          app:layout_constraintBottom_toBottomOf="@id/A"
40          app:layout_constraintStart_toEndOf="@id/A"
41          app:layout_constraintTop_toTopOf="@id/A"
42          app:layout_goneMarginStart="100dp" />
43
44  </androidx.constraintlayout.widget.ConstraintLayout>
```

> 当目标A控件隐藏时，B的`GONE Margin`就会生效


100
100
B

1.3 控件尺寸

1.3.1 尺寸限制

> 在 `ConstraintLayout` 中提供了一些尺寸限制的属性，可以用来限制最大、最小宽高度，这些属性只有在给出的宽度或高度为 `wrap_content` 时才会生效，比如想给宽度设置最小或最大值，那宽度就必须设置为 `wrap_content`，这个比较简单就不放示例代码了，具体的属性如下：

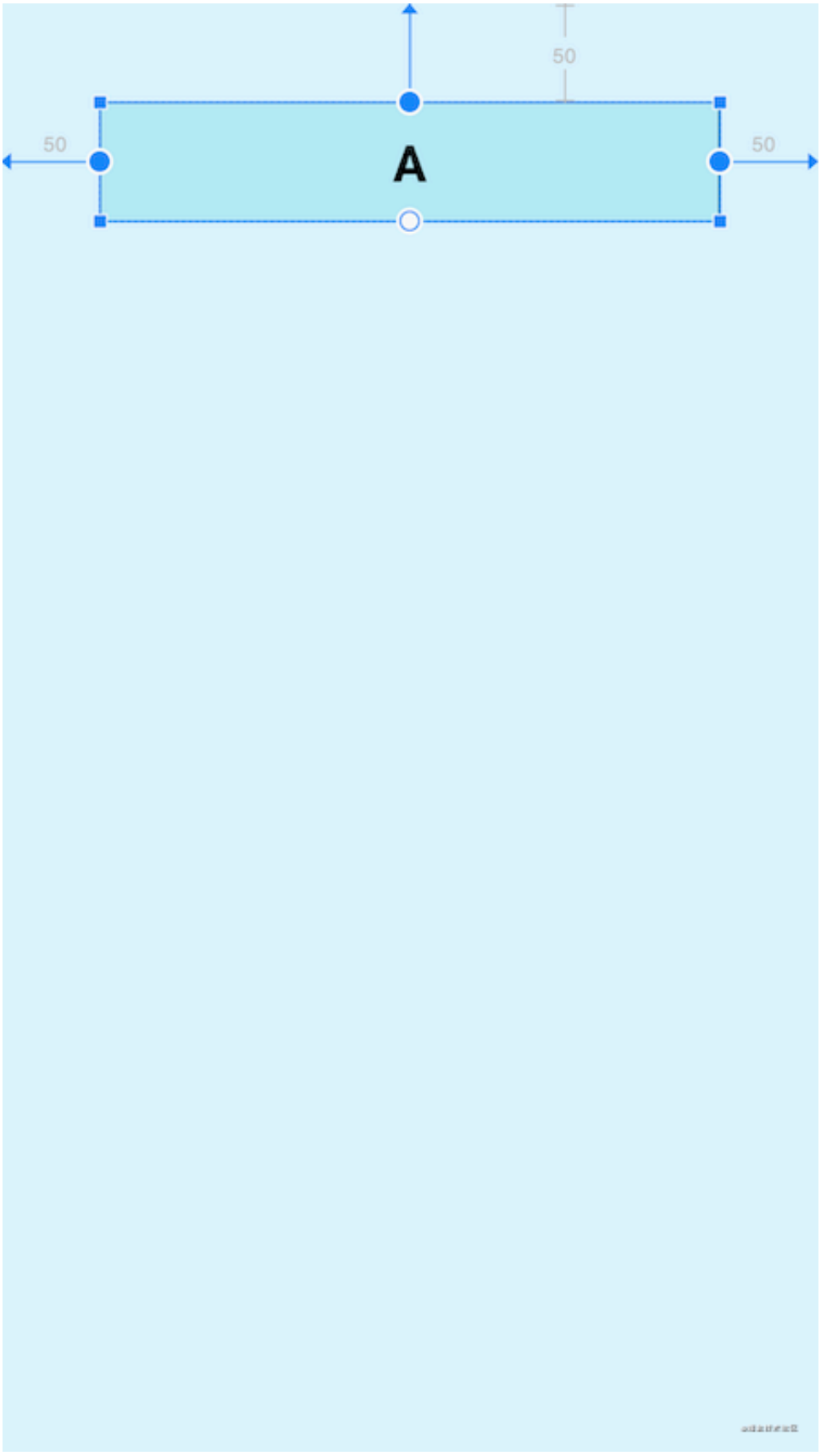
```
android:minWidth=""   设置view的最小宽度
android:minHeight=""  设置view的最小高度
android:maxWidth=""   设置view的最大宽度
android:maxHeight=""  设置view的最大高度
```

1.3.2 0dp(MATCH_CONSTRAINT)

> 设置 `view` 的大小除了传统的 `wrap_content`、`指定尺寸`、`match_parent` 外，`ConstraintLayout` 还可以设置为 `0dp（MATCH_CONSTRAINT）`，并且 `0dp` 的作用会根据设置的类型而产生不同的作用，进行设置类型的属性是 `layout_constraintWidth_default` 和 `layout_constraintHeight_default`，取值可为 `spread、percent、wrap`。具体的属性及示例如下：

```
app:layout_constraintWidth_default="spread|percent|wrap"
app:layout_constraintHeight_default="spread|percent|wrap"
```

- **spread（默认）**：占用所有的符合约束的空间

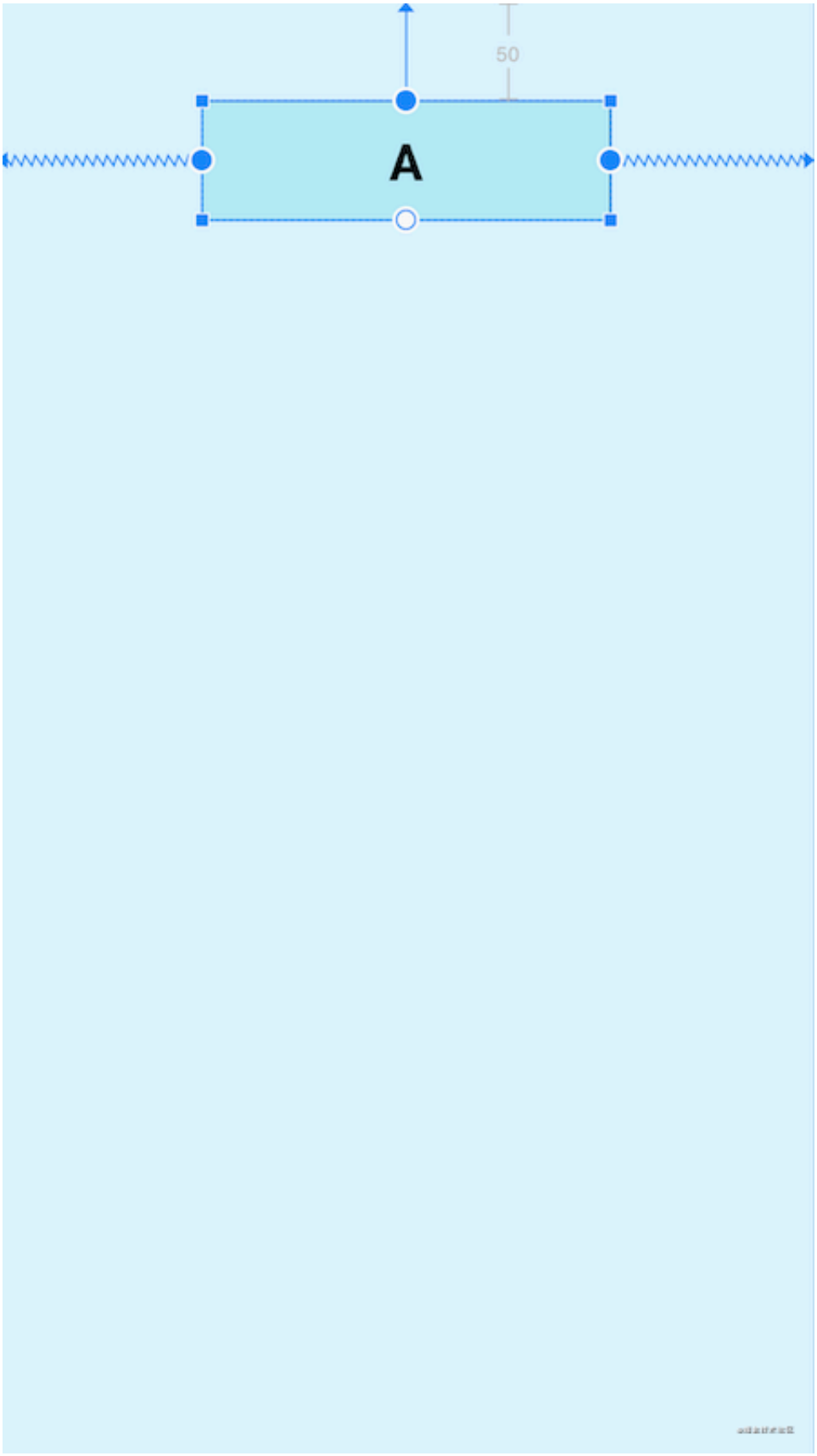



```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="0dp"
        android:layout_height="60dp"
        android:layout_marginStart="50dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="50dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintWidth_default="spread" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

> 可以看到，`view` 的宽度适应了所有有效的约束空间，左右留出了 `margin` 的设置值 `50dp`，这种效果就就是：自身 `view` 的大小充满可以配置的剩余空间，因为左右约束的都是父布局，所以 `view` 可配置的空间是整个父布局的宽度，又因为设置了 `margin`，所以会留出 `margin` 的大小，因为 `spread` 是默认值，所以可以不写 `app:layout_constraintWidth_default="spread"`。

- **percent**：按照父布局的百分比设置

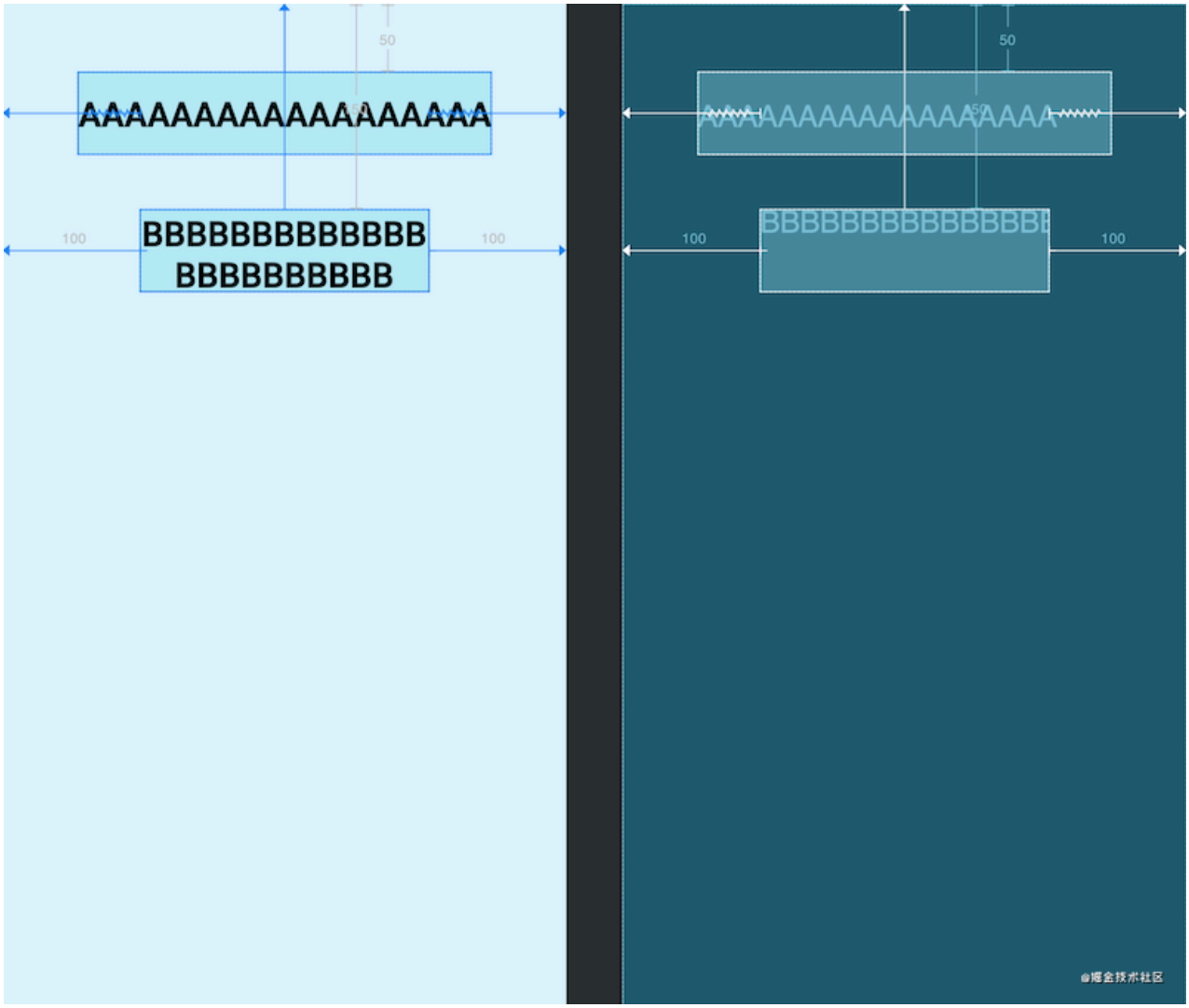



```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="0dp"
        android:layout_height="60dp"
        android:layout_marginTop="50dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintWidth_default="percent"
        app:layout_constraintWidth_percent="0.5" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

> `percent`模式的意思是自身`view`的尺寸是父布局尺寸的一定比例，上图所展示的是宽度是父布局宽度的0.5（50%，取值是0-1的小数），该模式需要配合`layout_constraintWidth_percent`使用，但是写了`layout_constraintWidth_percent`后，`layout_constraintWidth_default="percent"`其实就可以省略掉了。

- **wrap**：匹配内容大小但不超过约束限制



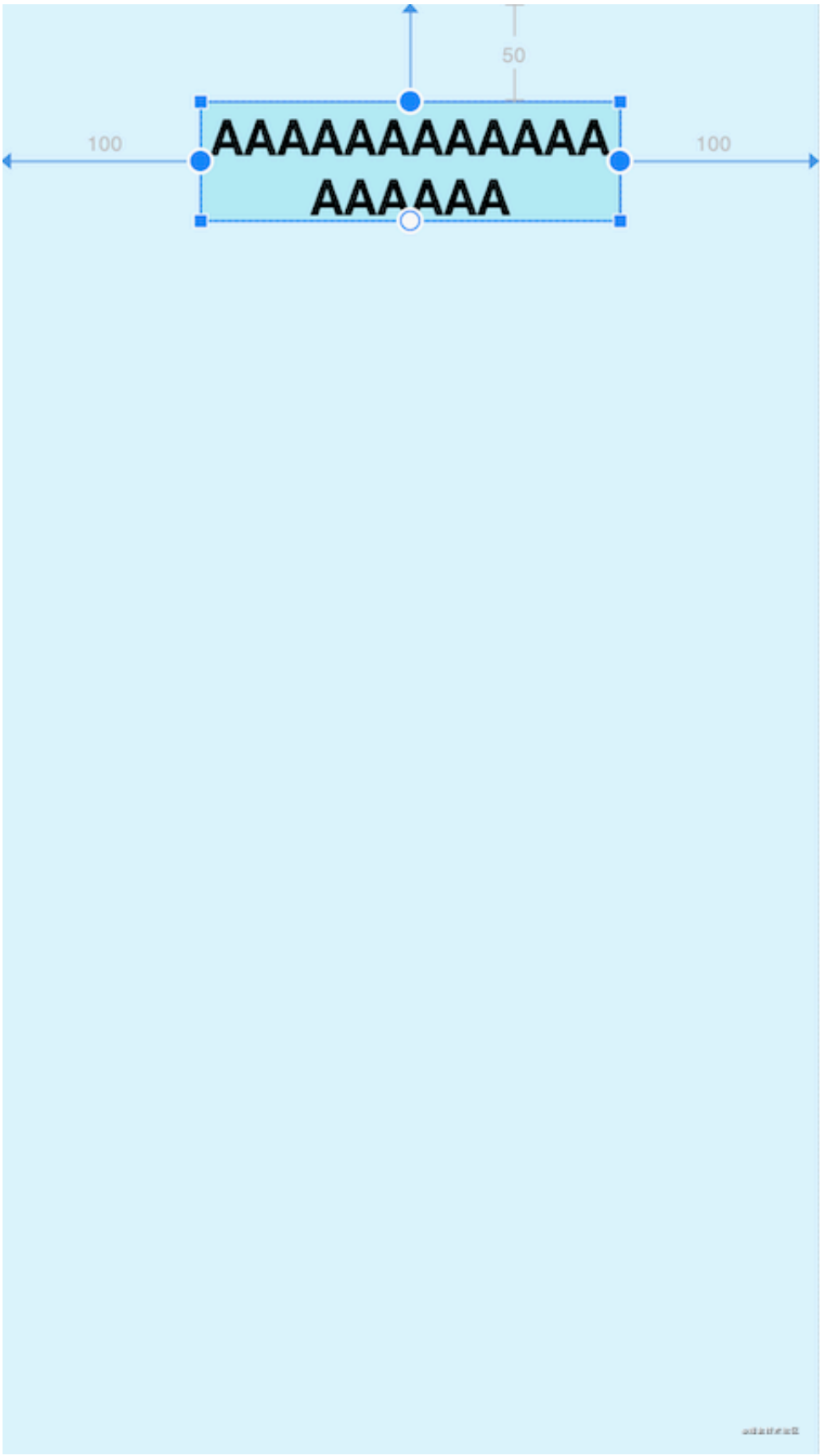

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <!--  宽度设置为wrap_content  -->
    <TextView
        android:id="@+id/A"
        android:layout_width="wrap_content"
        android:layout_height="60dp"
        android:layout_marginStart="100dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="100dp"
```

```
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="AAAAAAAAAAAAAAAAA"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintWidth_default="spread" />

    <!--  宽度设置为0dp wrap模式  -->
    <TextView
        android:id="@+id/B"
        android:layout_width="0dp"
        android:layout_height="60dp"
        android:layout_marginStart="100dp"
        android:layout_marginTop="150dp"
        android:layout_marginEnd="100dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="BBBBBBBBBBBBBBBBBBBBB"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintWidth_default="wrap" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

这里写了两个控件作为对比，控件A宽度设置为`wrap_content`，宽度适应内容大小，并且设置了`margin`，但是显然宽度已经超过`margin`的设置值了，而控件B宽度设置为`0dp wrap模式`，宽度适应内容大小，并且不会超过`margin`的设置值，也就是不会超过约束限制，这就是这两者的区别。Google还提供了两个属性用于强制约束：

```
<!--  当一个view的宽或高,设置成wrap_content时  -->
app:layout_constrainedWidth="true|false"
app:layout_constrainedHeight="true|false"
```

还是上一个例子，这里将控件A设置了强制约束，展示出的效果和控件B是一样的了：




```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="wrap_content"
        android:layout_height="60dp"
        android:layout_marginStart="100dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="100dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="AAAAAAAAAAAAAAAAA"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constrainedWidth="true"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintWidth_default="spread" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

除此之外，`0dp`还有一些其他的独特属性用于设置尺寸的大小限制

```
app:layout_constraintWidth_min=""    0dp下，宽度的最小值
app:layout_constraintHeight_min=""   0dp下，高度的最小值
app:layout_constraintWidth_max=""    0dp下，宽度的最大值
app:layout_constraintHeight_max=""   0dp下，高度的最大值
```

1.3.3 比例宽高（Ratio）

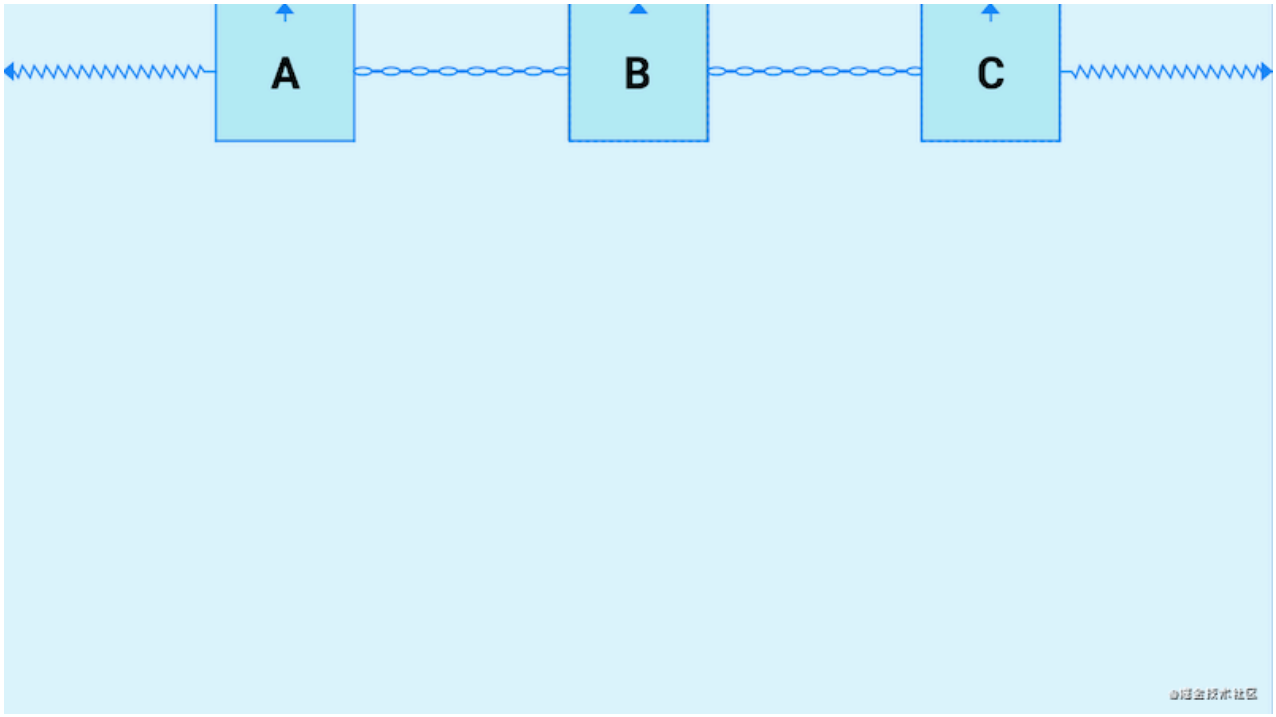
`ConstraintLayout`中可以对宽高设置比例，前提是至少有一个约束维度设置为`0dp`，这样比例才会生效，该属性可使用两种设置：
1.浮点值，表示宽度和高度之间的比率
2.宽度:高度，表示宽度和高度之间形式的比率

```
app:layout_constraintDimensionRatio=""  宽高比例
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="0dp"
        android:layout_height="100dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintDimensionRatio="1:1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.4 Chains(链)

> `Chains(链)`也是一个非常好用的特性，它是将许多个控件在水平或者垂直方向，形成一条链，用于平衡这些控件的位置，那么如何形成一条链呢？形成一条链要求链中的控件在水平或者垂直方向，首尾互相约束，这样就可以形成一条链，水平方向互相约束形成的就是一条水平链，反之则是垂直链，下面看示例：

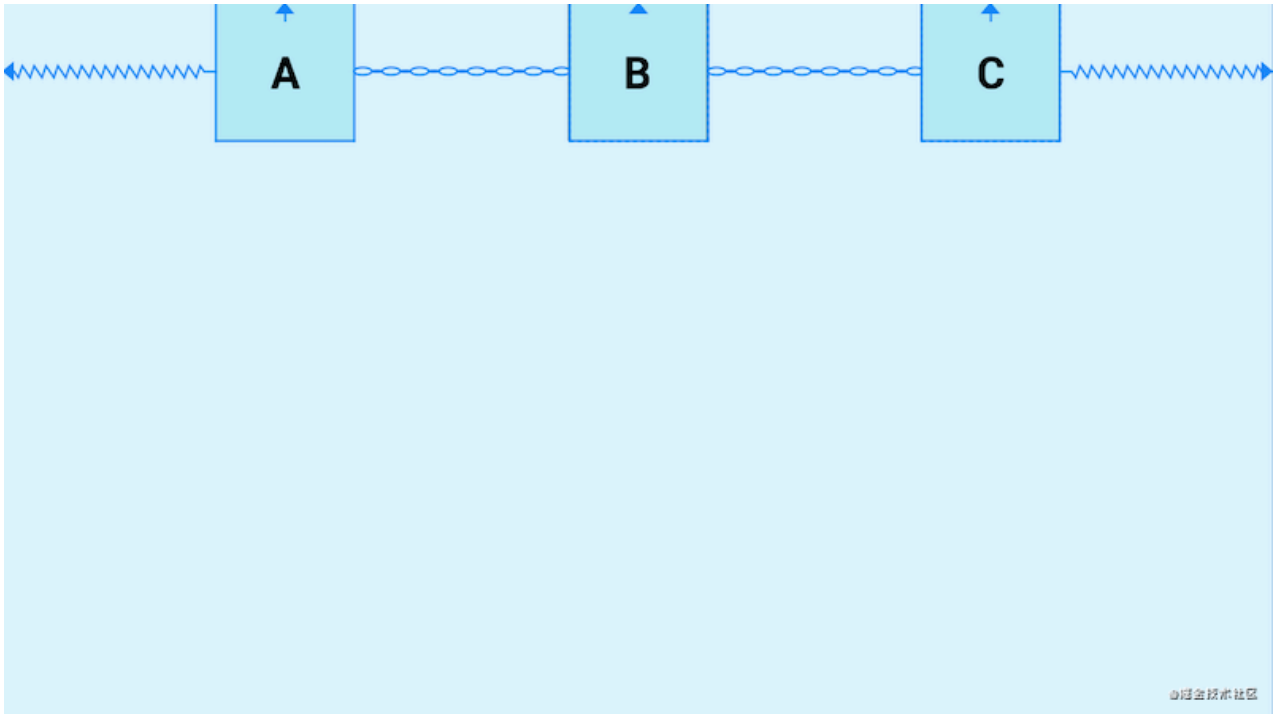



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toStartOf="@id/B"
        app:layout_constraintHorizontal_chainStyle="spread"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/B"
```

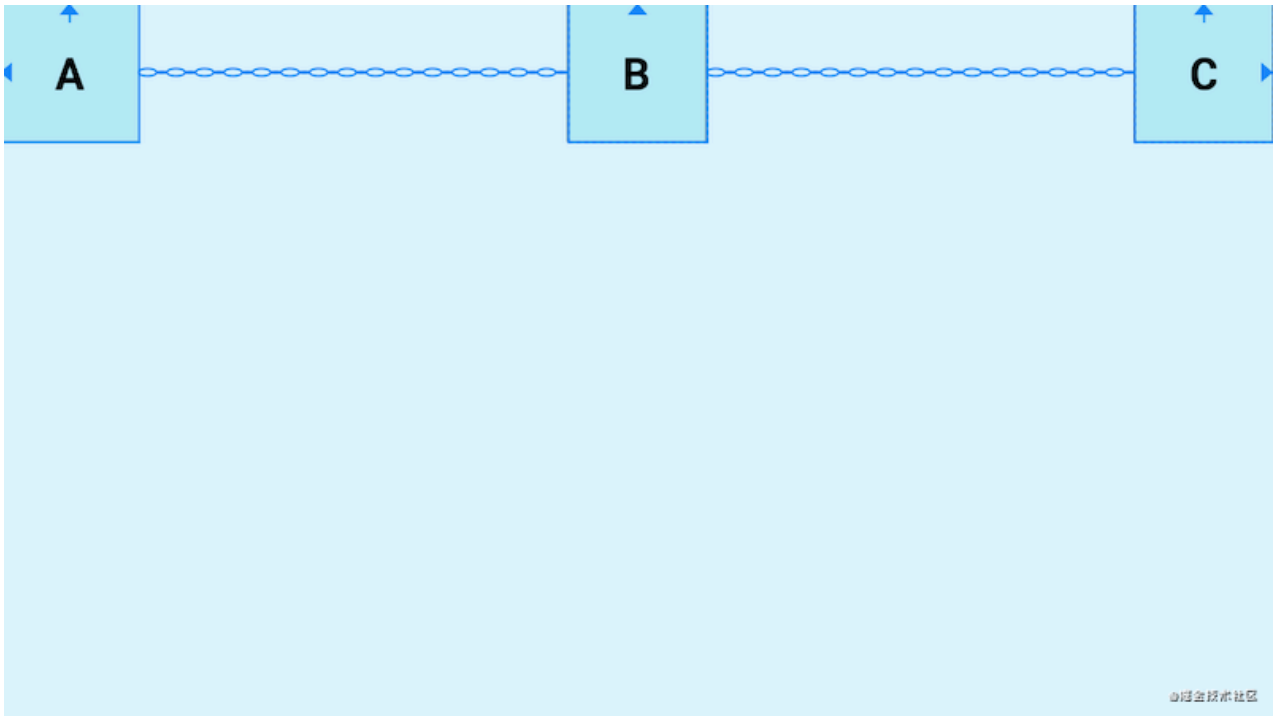
```
28          android:layout_width="80dp"
29          android:layout_height="80dp"
30          android:background="@drawable/tv_bg"
31          android:gravity="center"
32          android:text="B"
33          android:textColor="@color/black"
34          android:textSize="25sp"
35          android:textStyle="bold"
36          app:layout_constraintEnd_toStartOf="@id/C"
37          app:layout_constraintStart_toEndOf="@id/A"
38          app:layout_constraintTop_toTopOf="parent" />
39
40      <TextView
41          android:id="@+id/C"
42          android:layout_width="80dp"
43          android:layout_height="80dp"
44          android:background="@drawable/tv_bg"
45          android:gravity="center"
46          android:text="C"
47          android:textColor="@color/black"
48          android:textSize="25sp"
49          android:textStyle="bold"
50          app:layout_constraintEnd_toEndOf="parent"
51          app:layout_constraintStart_toEndOf="@id/B"
52          app:layout_constraintTop_toTopOf="parent" />
53
54  </androidx.constraintlayout.widget.ConstraintLayout>
```

> A、B、C，三个控件在水平方向上首尾互相约束，这样就形成了一条水平链，他们默认的模式是 `spread`，均分剩余空间，我们可以使用 `layout_constraintHorizontal_chainStyle` 和 `layout_constraintVertical_chainStyle` 分别对水平和垂直链设置模式，模式可选的值有：`spread、packed、spread_inside`

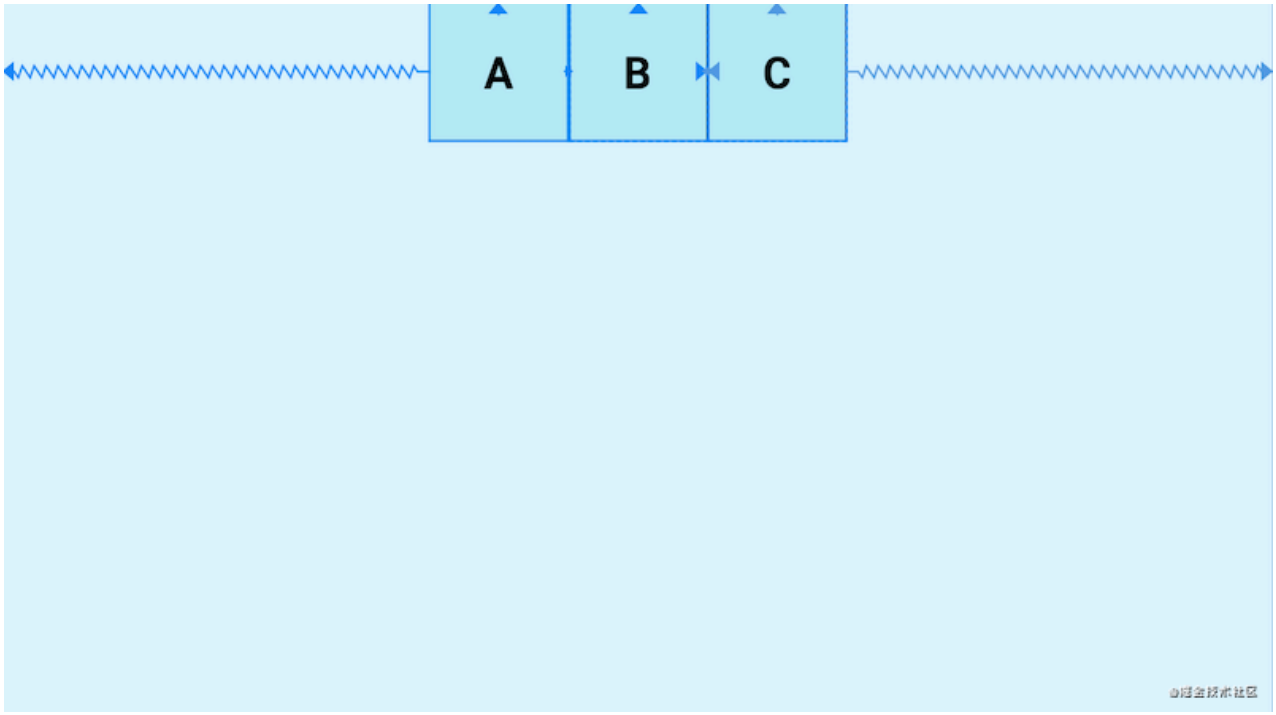
- **spread（默认）**：均分剩余空间



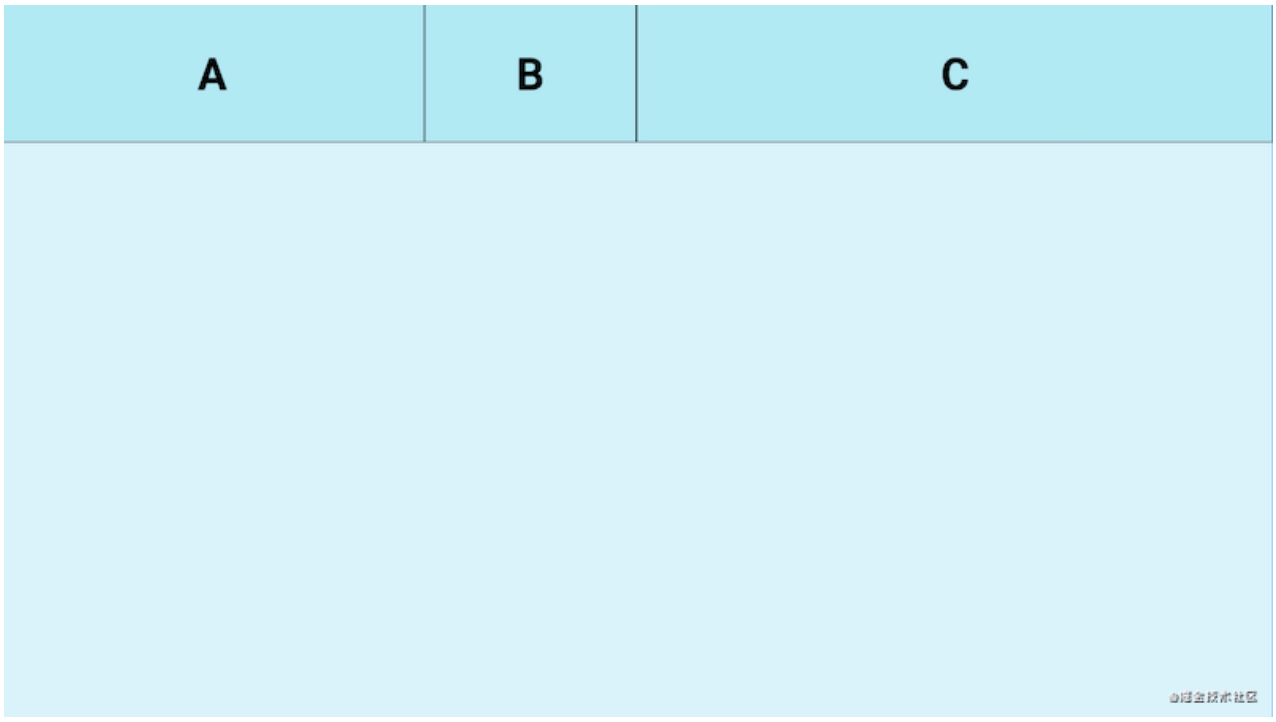
- **spread_inside**：两侧的控件贴近两边，剩余的控件均分剩余空间



- **packed**：所有控件贴紧居中



> Chains(链) 还支持 weight（权重） 的配置，使用 layout_constraintHorizontal_weight 和 layout_constraintVertical_weight 进行设置链元素的权重




```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
```

```
6       android:layout_height="match_parent"
7       android:background="#DAF3FE"
8       tools:context=".MainActivity"
9       tools:ignore="HardcodedText">
10
11      <TextView
12          android:id="@+id/A"
13          android:layout_width="0dp"
14          android:layout_height="80dp"
15          android:background="@drawable/tv_bg"
16          android:gravity="center"
17          android:text="A"
18          android:textColor="@color/black"
19          android:textSize="25sp"
20          android:textStyle="bold"
21          app:layout_constraintEnd_toStartOf="@id/B"
22          app:layout_constraintHorizontal_weight="2"
23          app:layout_constraintStart_toStartOf="parent"
24          app:layout_constraintTop_toTopOf="parent"
25          app:layout_constraintVertical_chainStyle="packed" />
26
27      <TextView
28          android:id="@+id/B"
29          android:layout_width="0dp"
30          android:layout_height="80dp"
31          android:background="@drawable/tv_bg"
32          android:gravity="center"
33          android:text="B"
34          android:textColor="@color/black"
35          android:textSize="25sp"
36          android:textStyle="bold"
37          app:layout_constraintEnd_toStartOf="@id/C"
38          app:layout_constraintHorizontal_weight="1"
39          app:layout_constraintStart_toEndOf="@id/A"
40          app:layout_constraintTop_toTopOf="parent" />
41
42      <TextView
43          android:id="@+id/C"
44          android:layout_width="0dp"
45          android:layout_height="80dp"
46          android:background="@drawable/tv_bg"
47          android:gravity="center"
48          android:text="C"
49          android:textColor="@color/black"
50          android:textSize="25sp"
51          android:textStyle="bold"
```

```
52          app:layout_constraintEnd_toEndOf="parent"
53          app:layout_constraintHorizontal_weight="3"
54          app:layout_constraintStart_toEndOf="@id/B"
55          app:layout_constraintTop_toTopOf="parent" />
56
57  </androidx.constraintlayout.widget.ConstraintLayout>
```

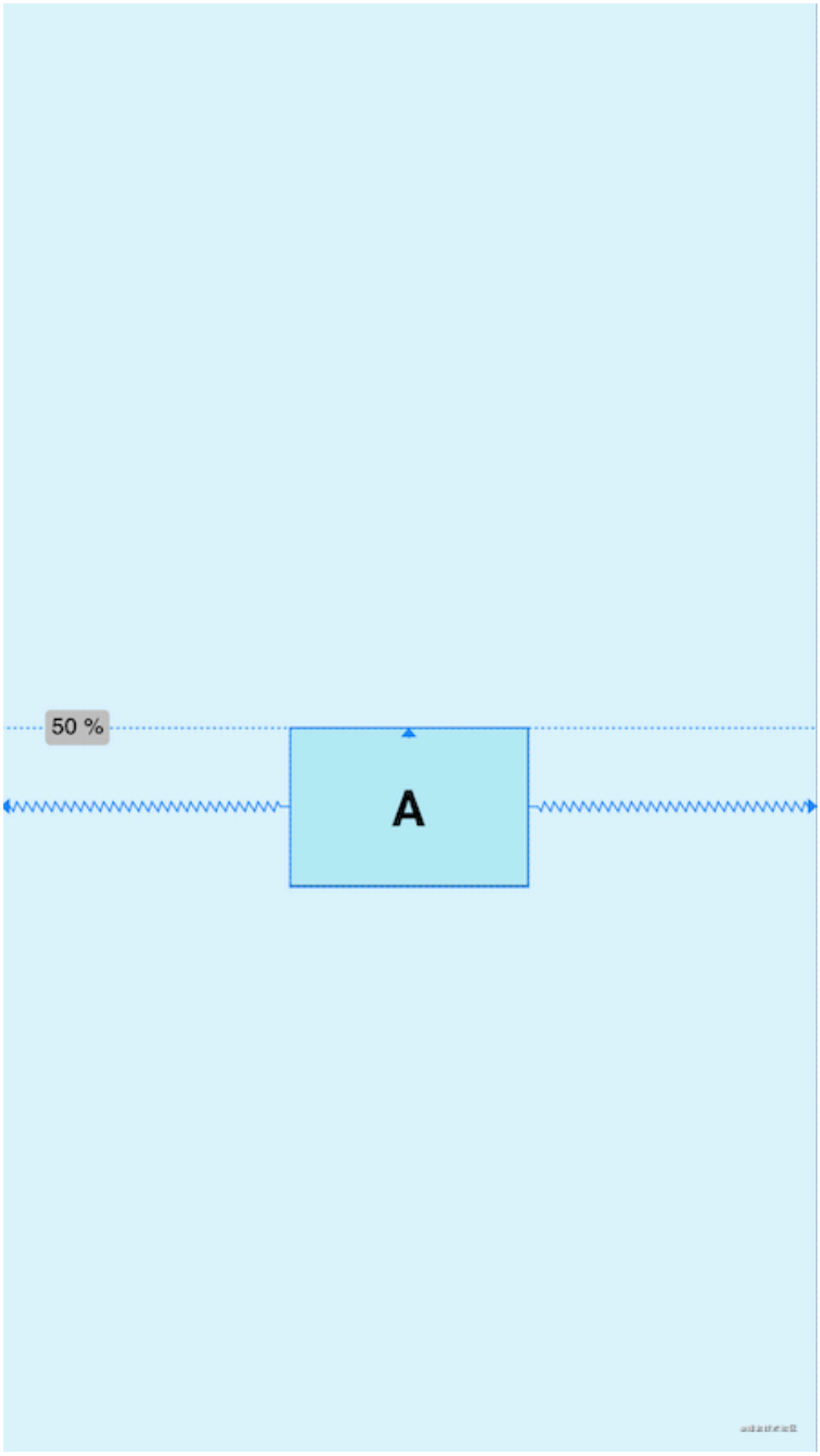
二. 辅助类

> `ConstraintLayout` 为了解决嵌套问题还提供了一系列的辅助控件帮助开发者布局，这些工具十分的方便，我在日常开发工作中也是使用的非常频繁

2.1 Guideline（参考线）

> `Guideline` 是一条参考线，可以帮助开发者进行辅助定位，并且实际上它并不会真正显示在布局中，像是数学几何中的辅助线一样，使用起来十分方便，出场率很高，`Guideline` 也可以用来做一些百分比分割之类的需求，有着很好的屏幕适配效果，`Guideline` 有水平和垂直方向之分，位置可以使用针对父级的百分比或者针对父级位置的距离

```
1  android:orientation="horizontal|vertical"   辅助线的对齐方式
2  app:layout_constraintGuide_percent="0-1"    距离父级宽度或高度的百分比(小数形式)
3  app:layout_constraintGuide_begin=""         距离父级起始位置的距离(左侧或顶部)
4  app:layout_constraintGuide_end=""           距离父级结束位置的距离(右侧或底部)
```



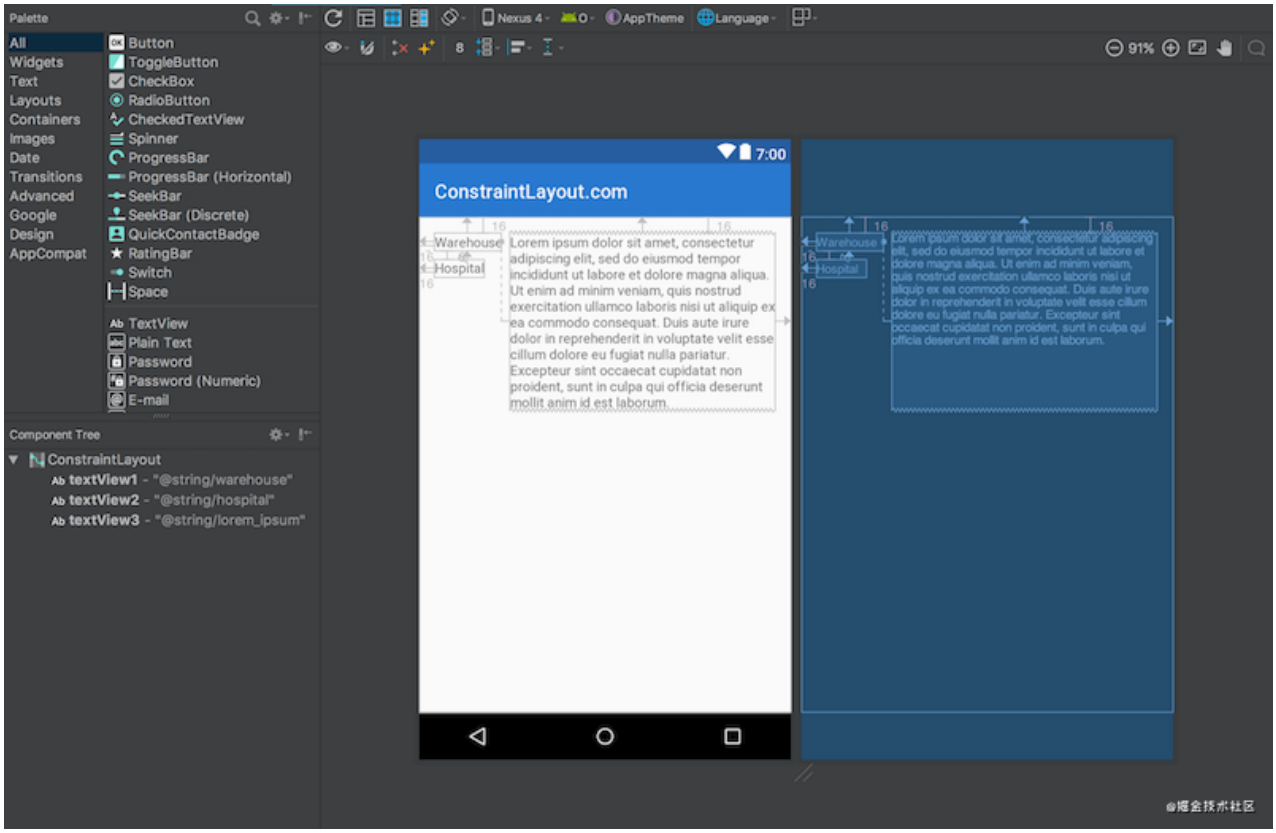

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/Guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />

    <TextView
        android:id="@+id/A"
        android:layout_width="120dp"
        android:layout_height="80dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@id/Guideline" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

> 上图中设置了一条水平方向位置在父级垂直方向0.5（50%）的`Guideline`，控件A的顶部依赖于`Guideline`，这样无论布局如何更改，`Guideline`的位置始终都会是父级垂直方向50%的位置，控件A的位置也不会偏离预设

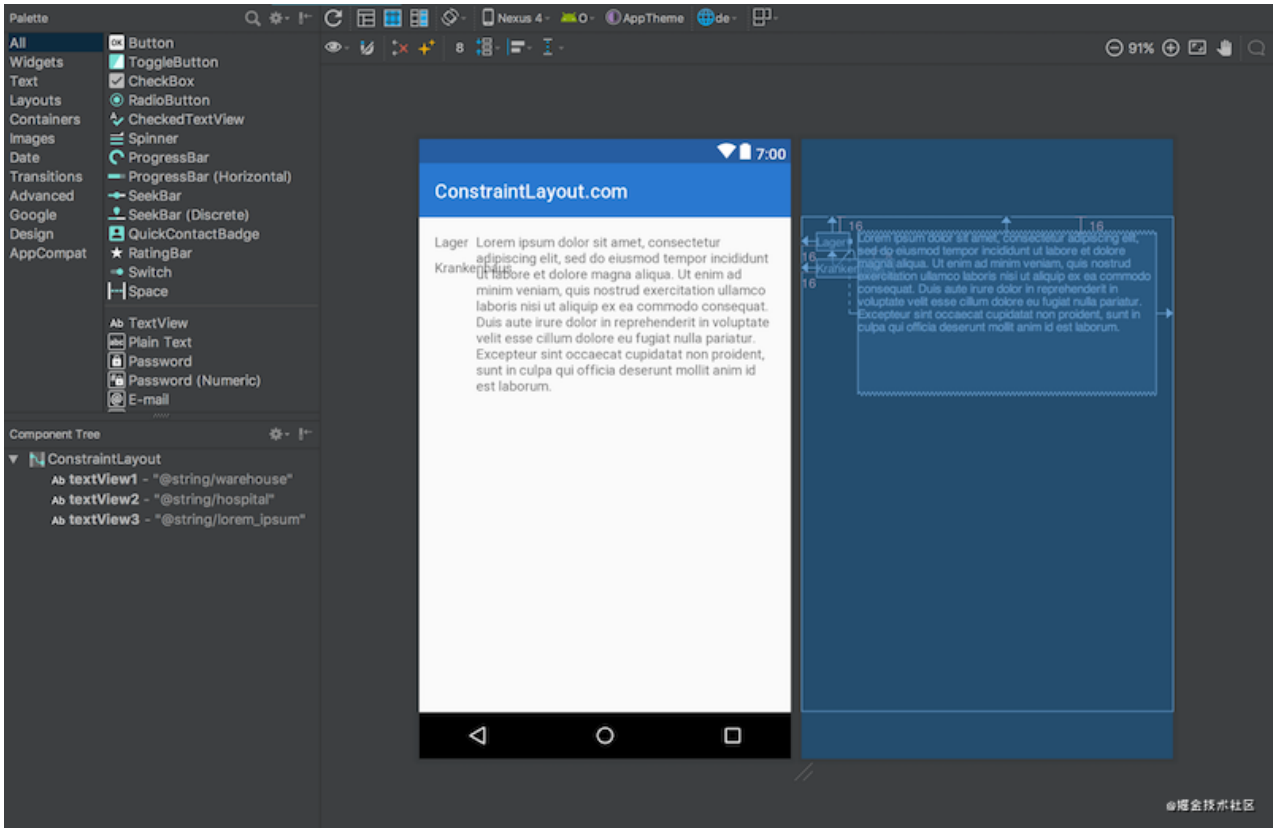
2.2 Barrier（屏障）

> 这个`Barrier`和`Guideline`一样，也不会实际出现在布局中，它的作用如同其名，形成一个屏障、障碍，使用也非常多。这里借助constraintlayout网站来讲解`Barrier`。
> 当我们创建布局时，有时会遇到布局可以根据本地化而更改的情况。这里借助有一个非常简单的例子：



这里有三个文本视图：左边的 `textView1` 和 `textView2` ；右边的 `textView3` 。 `textView3` 被限制在 `textView1` 的末尾，这工作得很好——它完全根据我们需要来定位和大小 `textView3` 。

然而，如果我们需要支持多种语言，事情会变得更加复杂。如果我们添加德语翻译，那么我们就会遇到一个问题，因为在英文版本中， `textView1` 中的文本比 `textView2` 中的文本长，而在德语中， `textView2` 中的文本比 `textView1` 长：



这里的问题在于 `textView3` 仍然是相对于 `textView1` 的，所以 `textView2` 直接插入了 `textView3` 中。在设计视图里看起来更明显（白色背景的那个）。比较直接的解决办法是使用 `TableLayout` ，或者把 `textView1` & `textView2` 包裹在一个垂直的， `android:layout_width="wrap_content"` 的 `LinearLayout中` 。然后让 `textView3` 约束在这个 `LinearLayout` 的后面。但是我们有更好的办法： `Barriers` 。 `Barriers` 的配置属性如下：

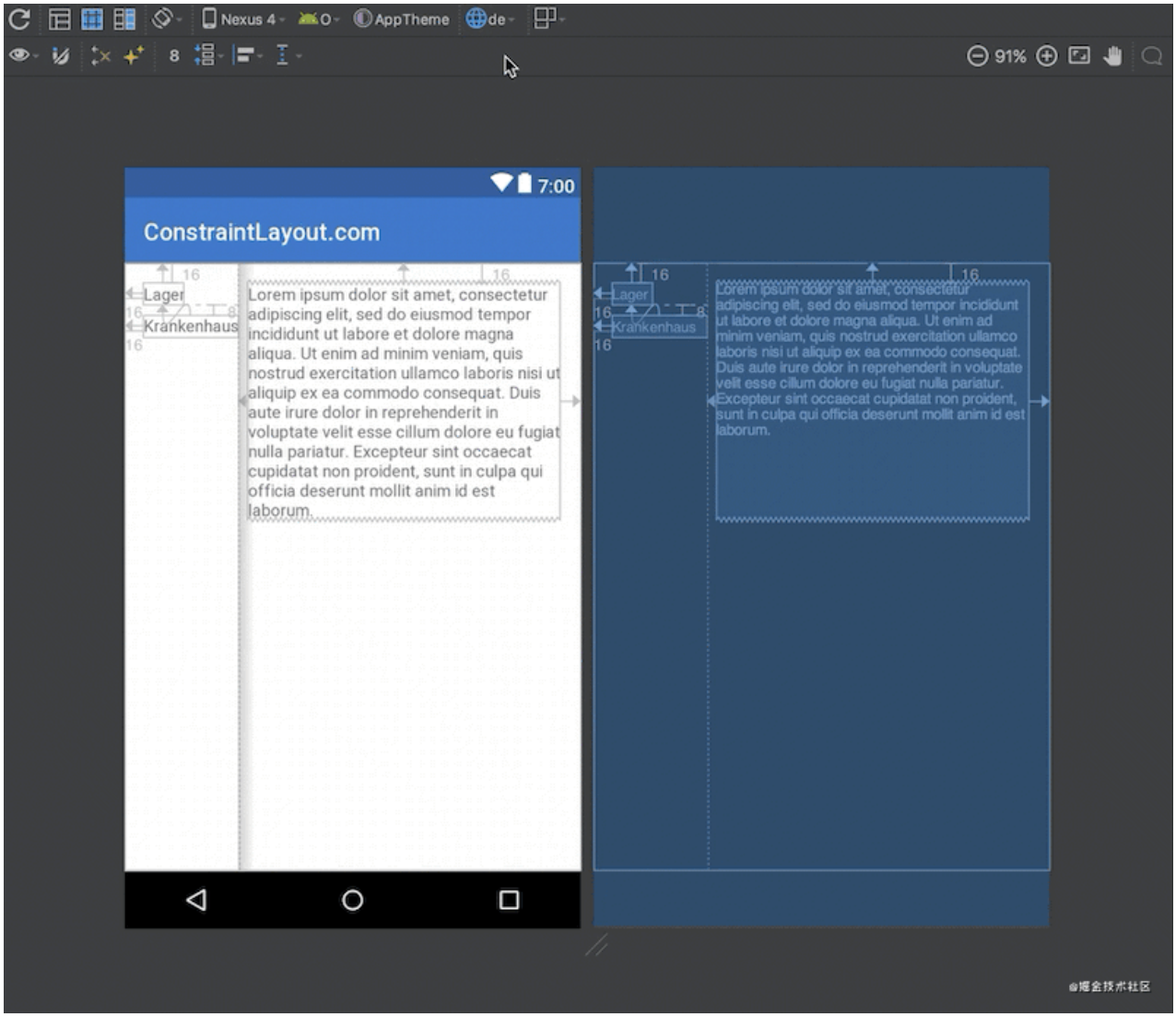
```
<!--  用于控制Barrier相对于给定的View的位置  -->
app:barrierDirection="top|bottom|left|right|start|end"

<!--  取值是要依赖的控件的id，Barrier将会使用ids中最大的一个的宽/高作为自己的位置  -->
app:constraint_referenced_ids="id,id"
```

修改过后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```

```
8       <TextView
9           android:id="@+id/textView1"
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content"
12          android:layout_marginStart="16dp"
13          android:layout_marginTop="16dp"
14          android:text="@string/warehouse"
15          app:layout_constraintStart_toStartOf="parent"
16          app:layout_constraintTop_toTopOf="parent" />
17
18      <TextView
19          android:id="@+id/textView2"
20          android:layout_width="wrap_content"
21          android:layout_height="wrap_content"
22          android:layout_marginStart="16dp"
23          android:layout_marginTop="8dp"
24          android:text="@string/hospital"
25          app:layout_constraintStart_toStartOf="parent"
26          app:layout_constraintTop_toBottomOf="@+id/textView1" />
27
28      <androidx.constraintlayout.widget.Barrier
29          android:id="@+id/barrier7"
30          android:layout_width="wrap_content"
31          android:layout_height="wrap_content"
32          app:barrierDirection="end"
33          app:constraint_referenced_ids="textView2,textView1" />
34
35      <TextView
36          android:id="@+id/textView3"
37          android:layout_width="0dp"
38          android:layout_height="wrap_content"
39          android:layout_marginStart="8dp"
40          android:text="@string/lorem_ipsum"
41          app:layout_constraintStart_toEndOf="@+id/barrier7"
42          app:layout_constraintTop_toTopOf="parent" />
43
44  </androidx.constraintlayout.widget.ConstraintLayout>
```

> 效果：



> 为了看到整体的效果，可以切换语言，此时你会看到 `Barrier` 会自动位于较宽的那个 `textView` 后面，也就间接让 `textView3` 也位于了正确的位置。

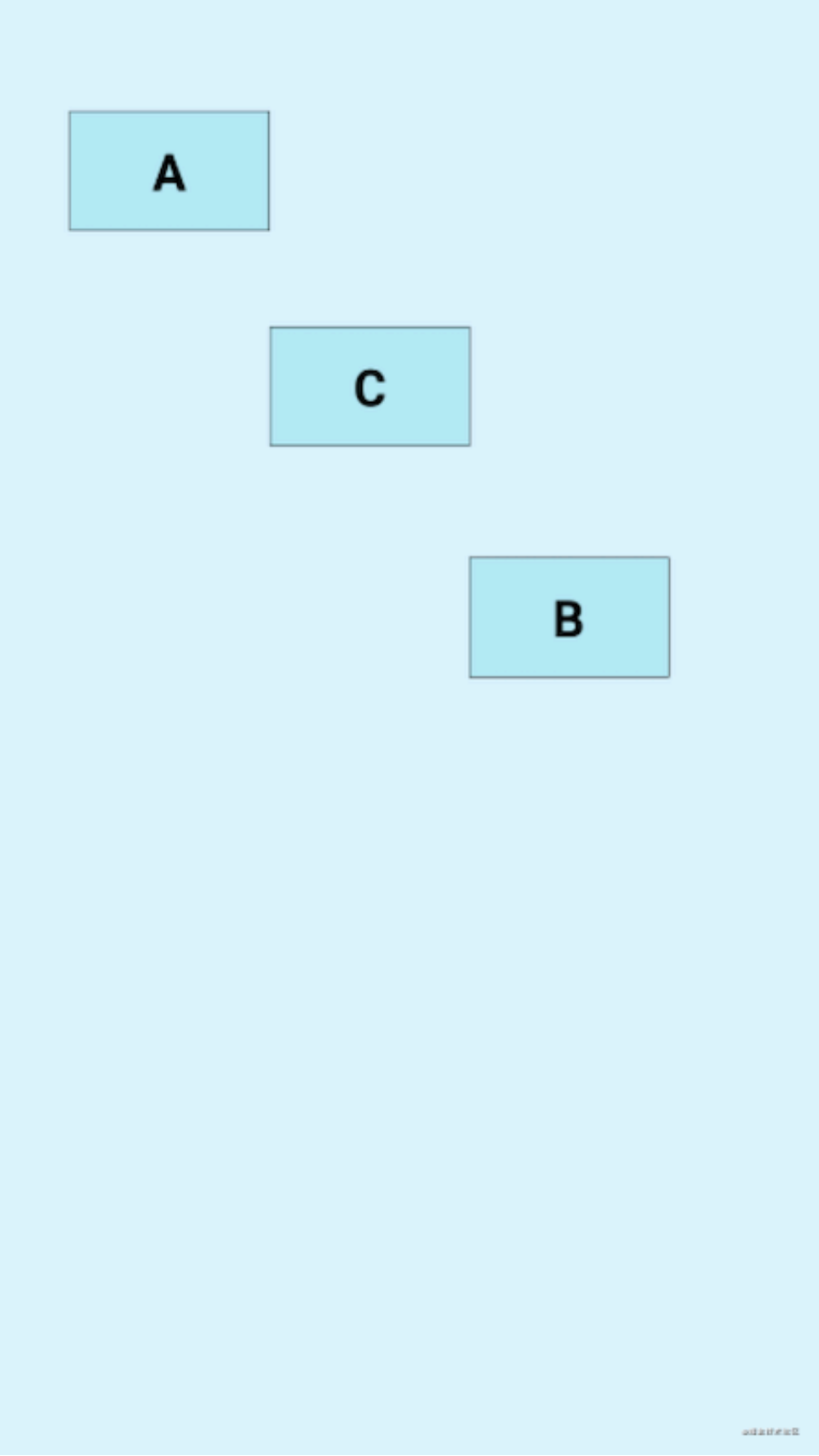
> 上述例子是直接使用的constraintlayout网站中的例子，可以直接访问链接进行查看。

2.3 Group（组）

> 工作当中常常会有很多个控件同时隐藏或者显示的场景，传统做法要么是进行嵌套，对父布局进行隐藏或显示，要么就是一个一个设置，这显然都不是很好的办法， `ConstraintLayout` 中的 `Group` 就是来解决这个问题的。 `Group` 的作用就是可以对一组控件同时隐藏或显示，没有其他的作用，它的属性如下：

```
1   app:constraint_referenced_ids="id,id"  加入组的控件id
```

示例：

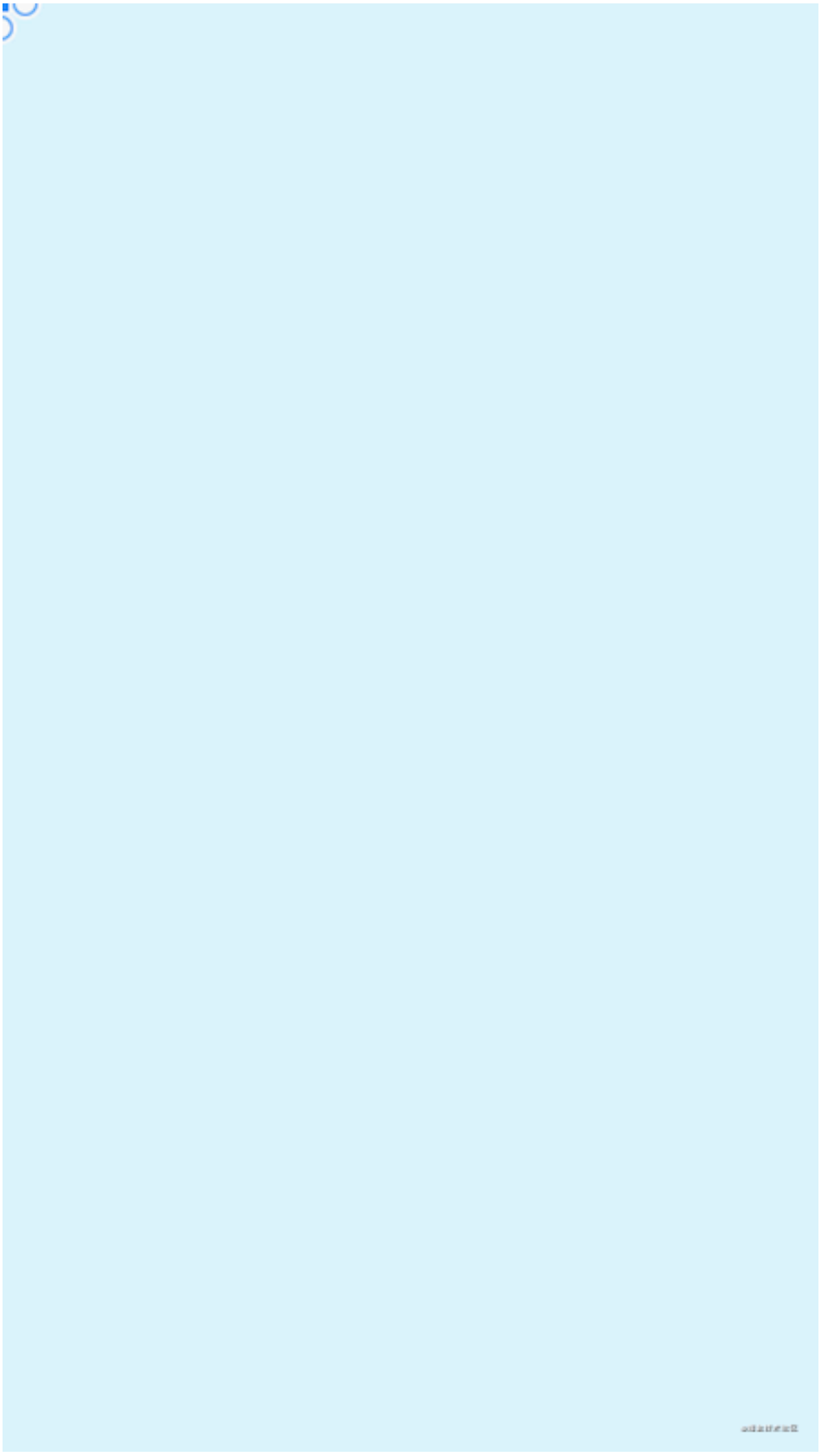



```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="100dp"
        android:layout_height="60dp"
        android:layout_marginTop="56dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.115"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/B"
        android:layout_width="100dp"
        android:layout_height="60dp"
        android:layout_marginTop="280dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="B"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.758"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/C"
        android:layout_width="100dp"
        android:layout_height="60dp"
```

```
47          android:layout_marginTop="164dp"
48          android:background="@drawable/tv_bg"
49          android:gravity="center"
50          android:text="C"
51          android:textColor="@color/black"
52          android:textSize="25sp"
53          android:textStyle="bold"
54          app:layout_constraintEnd_toEndOf="parent"
55          app:layout_constraintHorizontal_bias="0.437"
56          app:layout_constraintStart_toStartOf="parent"
57          app:layout_constraintTop_toTopOf="parent" />
58
59      <androidx.constraintlayout.widget.Group
60          android:layout_width="wrap_content"
61          android:layout_height="wrap_content"
62          android:visibility="visible"
63          app:constraint_referenced_ids="A,B,C" />
64
65  </androidx.constraintlayout.widget.ConstraintLayout>
```

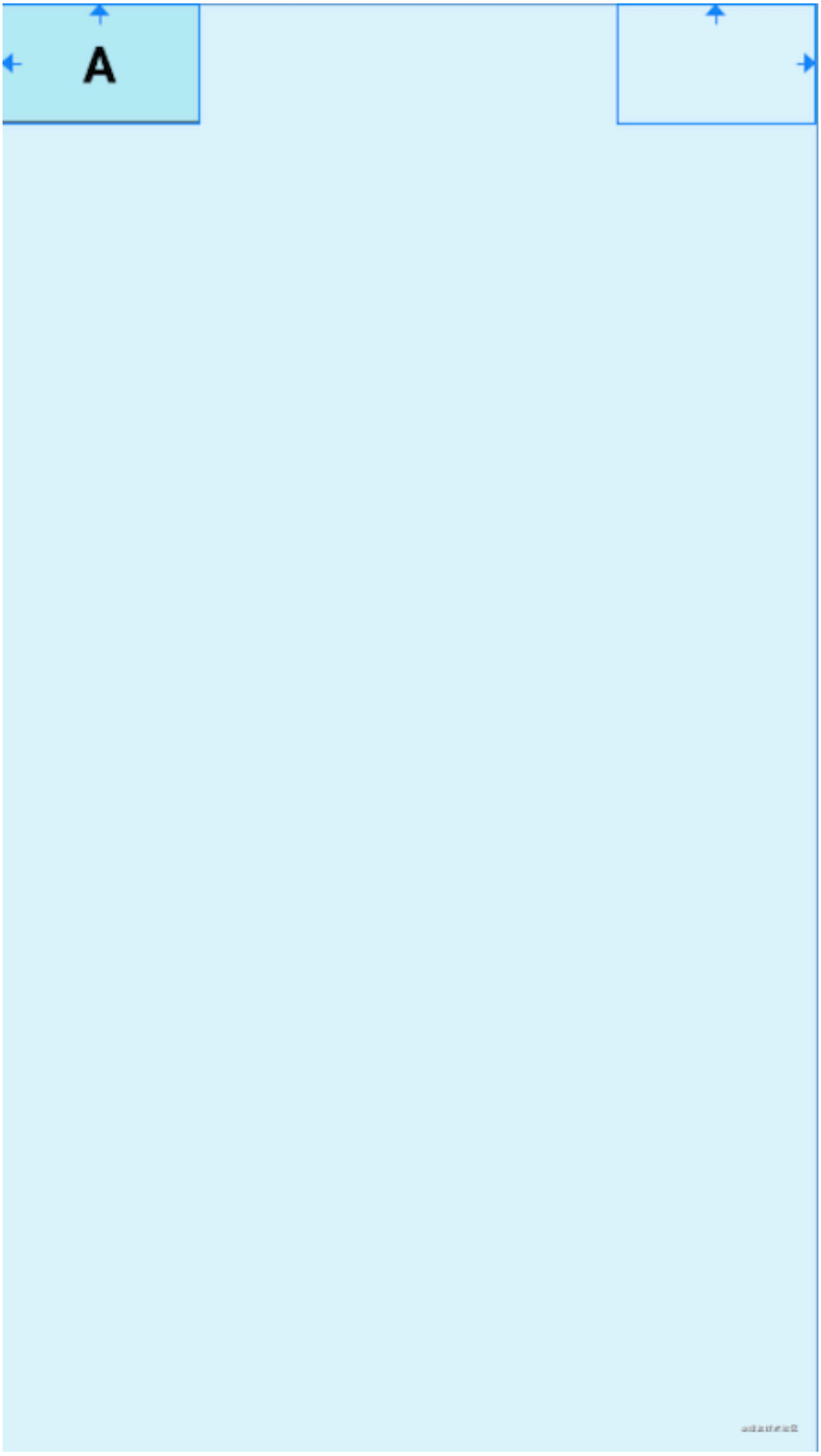
A、B、C三个 `view` ，受 `Group` 控制，当 `Group` 的 `visibility` 为 `visible` 时，它们都是正常显示的，设置为 `gone` 时，它们都会隐藏：



2.4 Placeholder（占位符）

> `Placeholder`的作用就是占位，它可以在布局中占好位置，通过`app:content=""`属性，或者动态调用`setContent()`设置内容，来让某个控件移动到此占位符中

> 示例：




```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="100dp"
        android:layout_height="60dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.constraintlayout.widget.Placeholder
        android:layout_width="100dp"
        android:layout_height="60dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

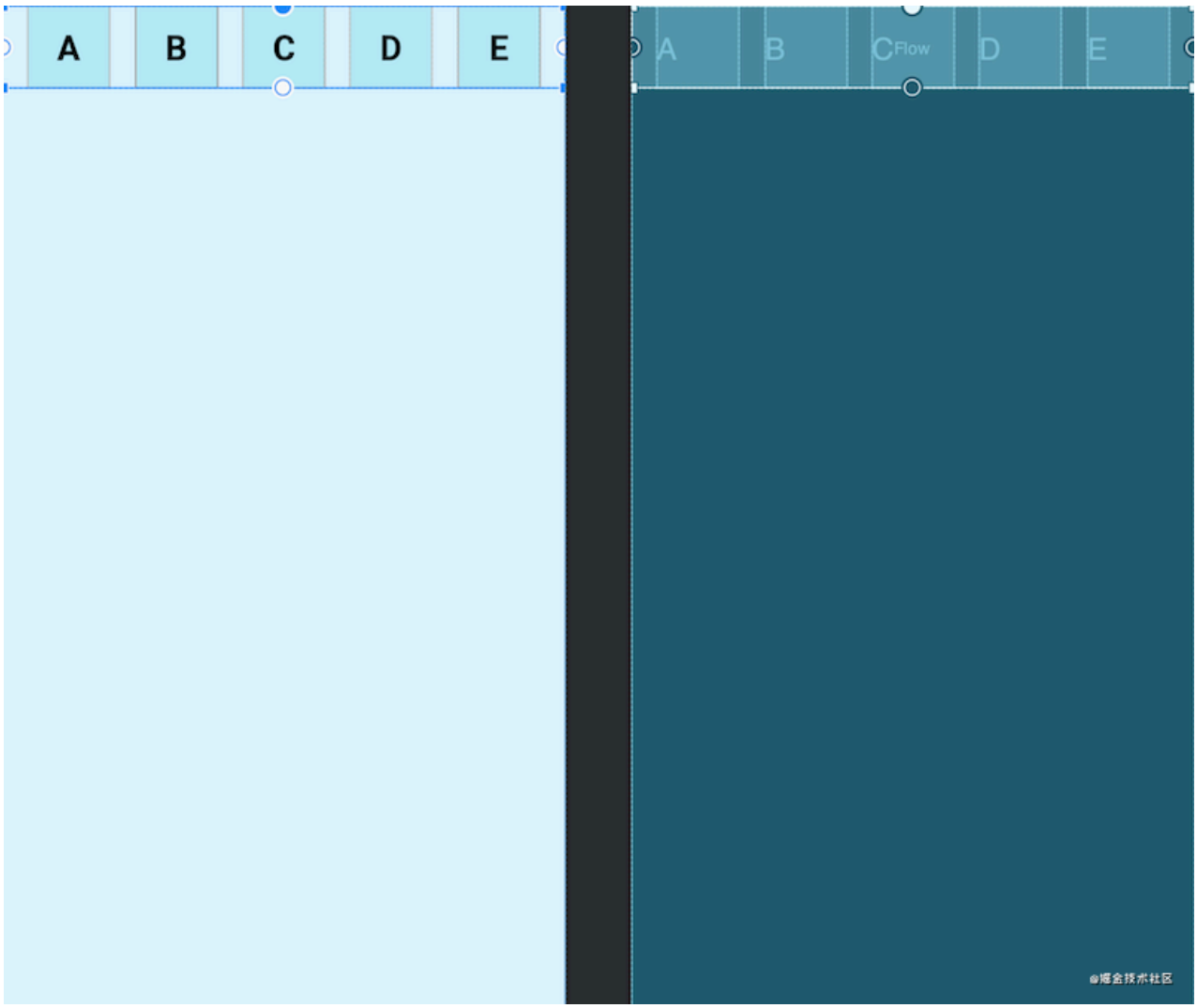
> 当我们设置 `app:content="@+id/A"` 或者调用 `setContent()` 时，控件A就会被移动到 `Placeholder` 中，当然在布局中使用 `app:content=""` 显然就失去了它的作用。

2.5 Flow（流式虚拟布局）

> `Flow` 是用于构建链的新虚拟布局，当链用完时可以缠绕到下一行甚至屏幕的另一部分。当您在一个链中布置多个项目时，这很有用，但是您不确定容器在运行时的大小。您可以使用它来根据应用程序中的动态尺寸（例如旋转时的屏幕宽度）构建布局。 `Flow` 是一种虚拟布局。在 `ConstraintLayout` 中，虚拟布局(`Virtual layouts`)作为 `virtual view group` 的角色参与约束和布局中，但是它们并不会作为视图添加到视图层级结构中，而是仅仅引用其它视图来辅助它们在布局系统中完成各自的布局功能。

> 下面使用动画来展示Flow创建多个链将布局元素充裕地填充一整行：

> 我们来看具体的例子：




```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

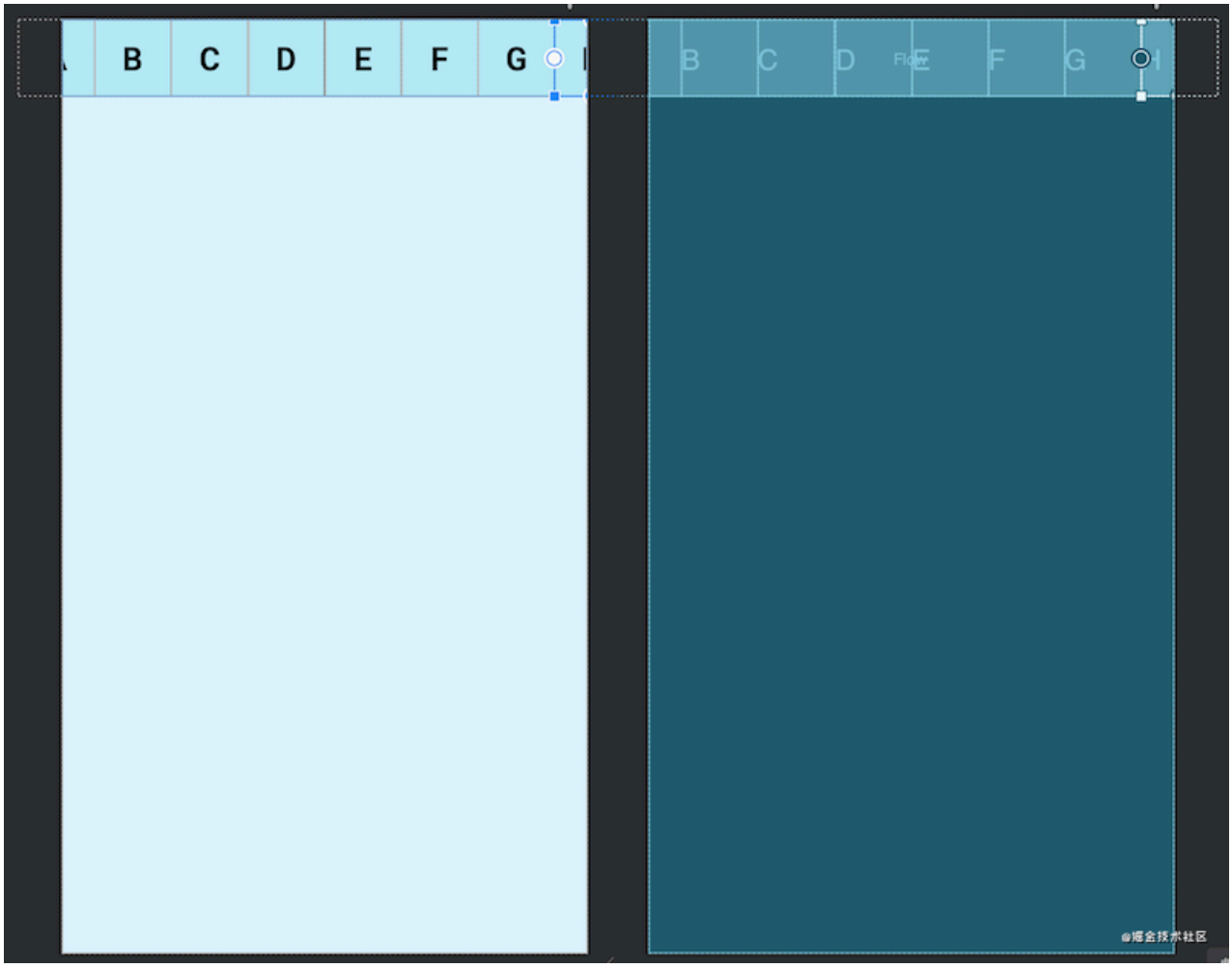
```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/A"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="A"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/B"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="B"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/C"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="C"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/D"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="@drawable/tv_bg"
        android:gravity="center"
        android:text="D"
```

```
51          android:textColor="@color/black"
52          android:textSize="25sp"
53          android:textStyle="bold" />
54
55      <TextView
56          android:id="@+id/E"
57          android:layout_width="60dp"
58          android:layout_height="60dp"
59          android:background="@drawable/tv_bg"
60          android:gravity="center"
61          android:text="E"
62          android:textColor="@color/black"
63          android:textSize="25sp"
64          android:textStyle="bold" />
65
66      <androidx.constraintlayout.helper.widget.Flow
67          android:layout_width="match_parent"
68          android:layout_height="wrap_content"
69          app:constraint_referenced_ids="A,B,C,D,E"
70          app:layout_constraintTop_toTopOf="parent" />
71
72  </androidx.constraintlayout.widget.ConstraintLayout>
```

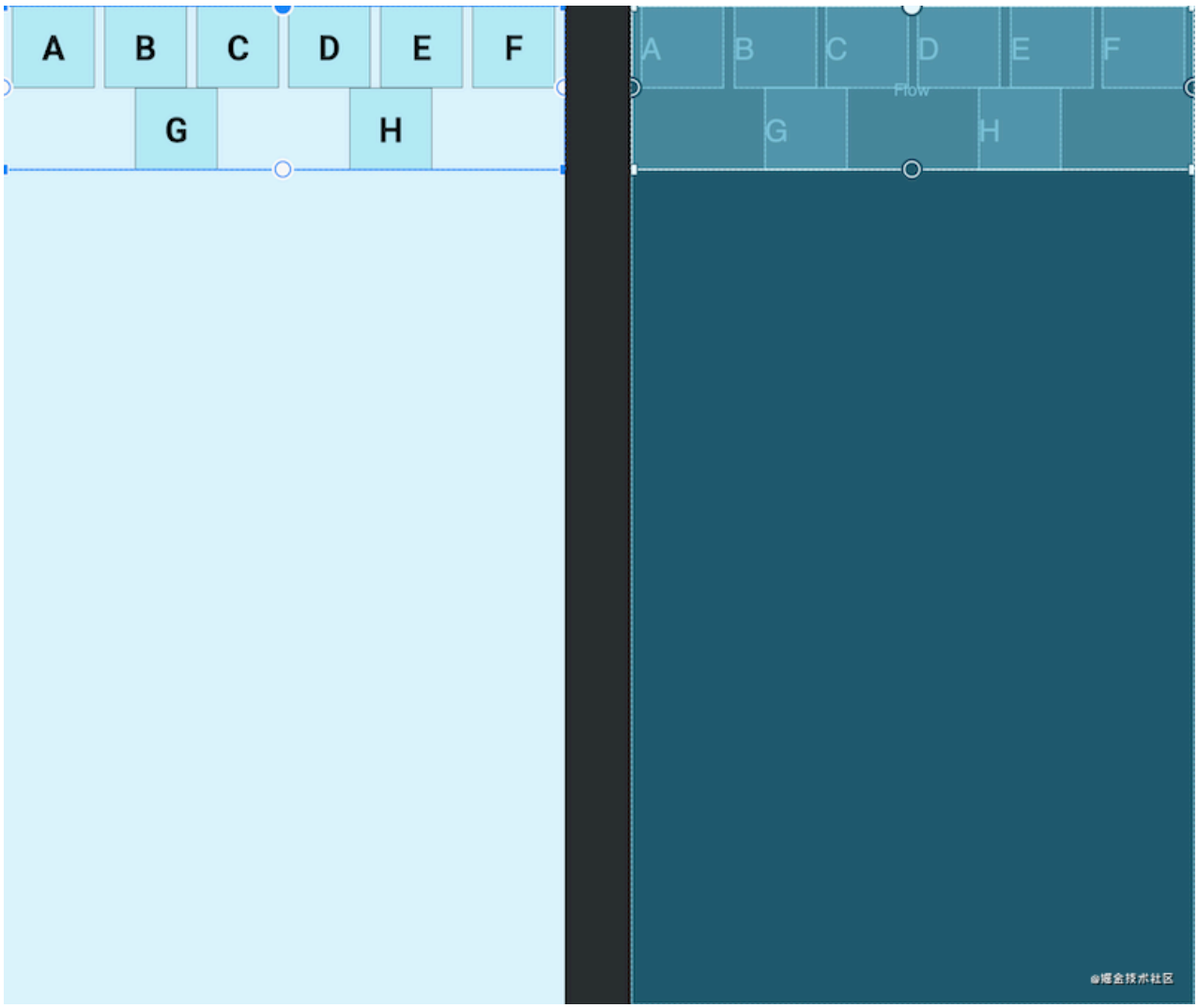
2.5.1 链约束

> `Flow` 的 `constraint_referenced_ids` 关联的控件是没有设置约束的，这一点和普通的链是不一样的，这种排列方式是 `Flow` 的默认方式 `none` ，我们可以使用 `app:flow_wrapMode=""` 属性来设置排列方式，并且我们还可以使用 `flow_horizontalGap` 和 `flow_verticalGap` 分别设置两个 `view` 在水平和垂直方向的间隔，下面我们再添加几个控件来展示三种排列方式：

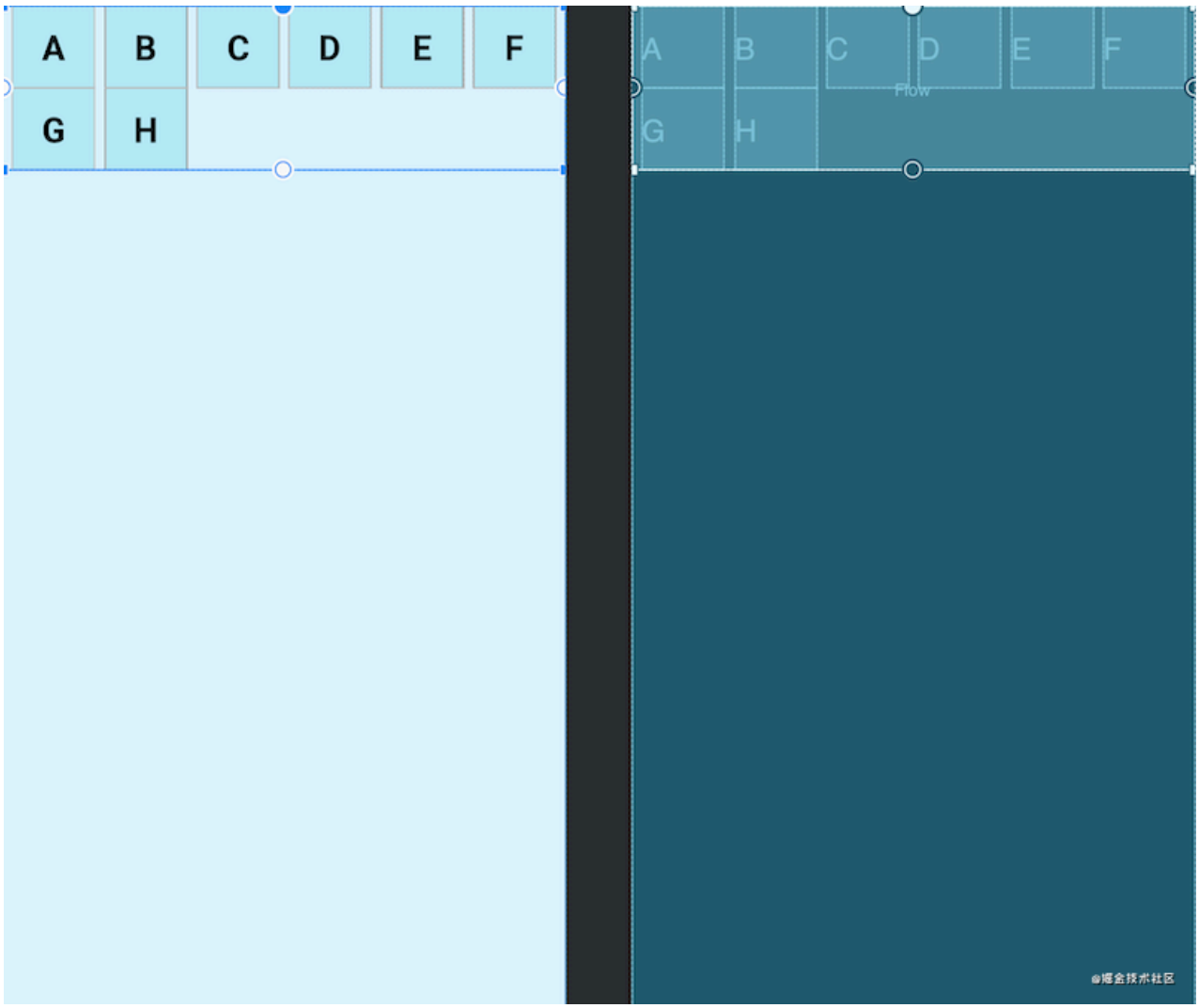
- **none（默认值）**：所有引用的 `view` 形成一条链，水平居中，超出屏幕两侧的 `view` 不可见



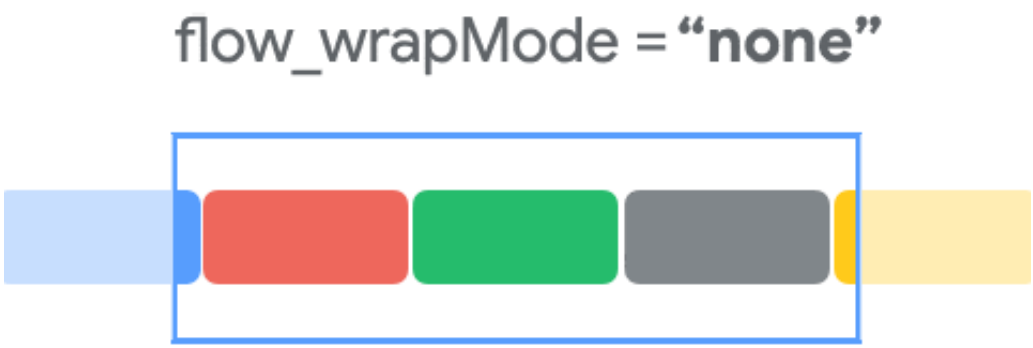
- **chian**：所引用的`view`形成一条链，超出部分会自动换行，同行的`view`会平分宽度。



- **aligned**：所引用的 `view` 形成一条链，但 `view` 会在同行同列。



> 下面使用动画来展示三种效果的变化



当 `flow_wrapMode` 的值是 `chian` 或 `aligned` 时，我们还可以针对不同的链进行配置，这里就不一一展示效果了，具体的属性如下：

```
app:flow_horizontalStyle="packed | spread | spread_inside"   所有水平链的配置
app:flow_verticalStyle="packed | spread | spread_inside"     所有垂直链的配置

app:flow_firstHorizontalStyle="packed | spread | spread_inside" 第一条水平链的配置，其他条不生效
app:flow_firstVerticalStyle="packed | spread | spread_inside"    第一条垂直链的配置，其他条不生效
app:flow_lastHorizontalStyle="packed | spread | spread_inside"   最后一条水平链的配置，其他条不生效
app:flow_lastVerticalStyle="packed | spread | spread_inside"     最后一条垂直链的配置，其他条不生效
```

2.5.2 对齐约束

上面展示的都是相同大小的 `view` ，那么不同大小 `view` 的对齐方式， `Flow` 也提供了相应的属性进行配置(`flow_wrapMode="aligned"` 时，我试着没有效果)

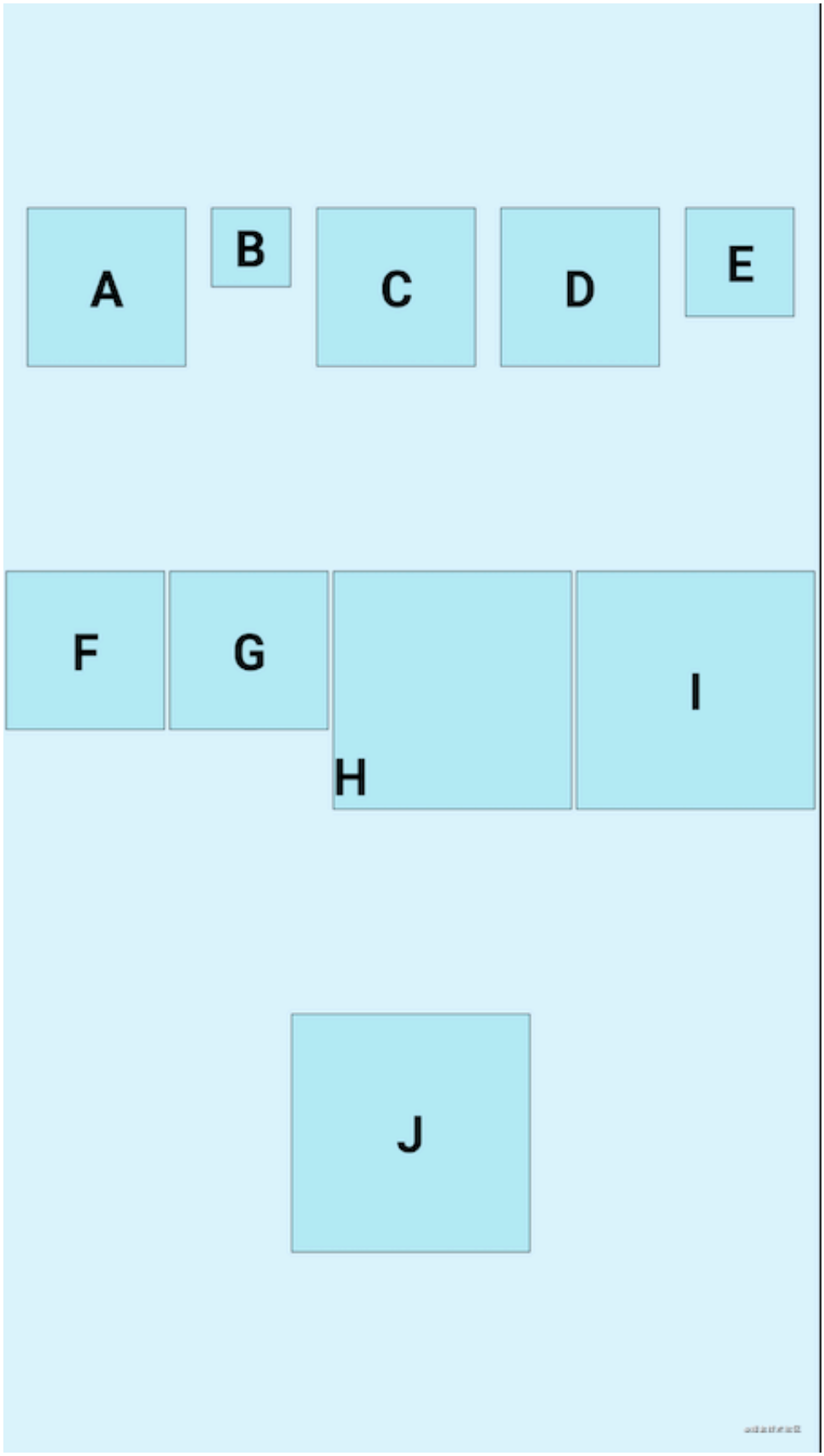
```
<!--  top:顶对齐、bottom:底对齐、center:中心对齐、baseline:基线对齐  -->
app:flow_verticalAlign="top | bottom | center | baseline"

<!--  start:开始对齐、end:结尾对齐、center:中心对齐  -->
app:flow_horizontalAlign="start|end|center"
```

使用 `flow_verticalAlign` 时，要求 `orientation` 的方向是 `horizontal` ，而使用 `flow_horizontalAlign` 时，要求 `orientation` 的方向是 `vertical`

下面展示下各个效果：

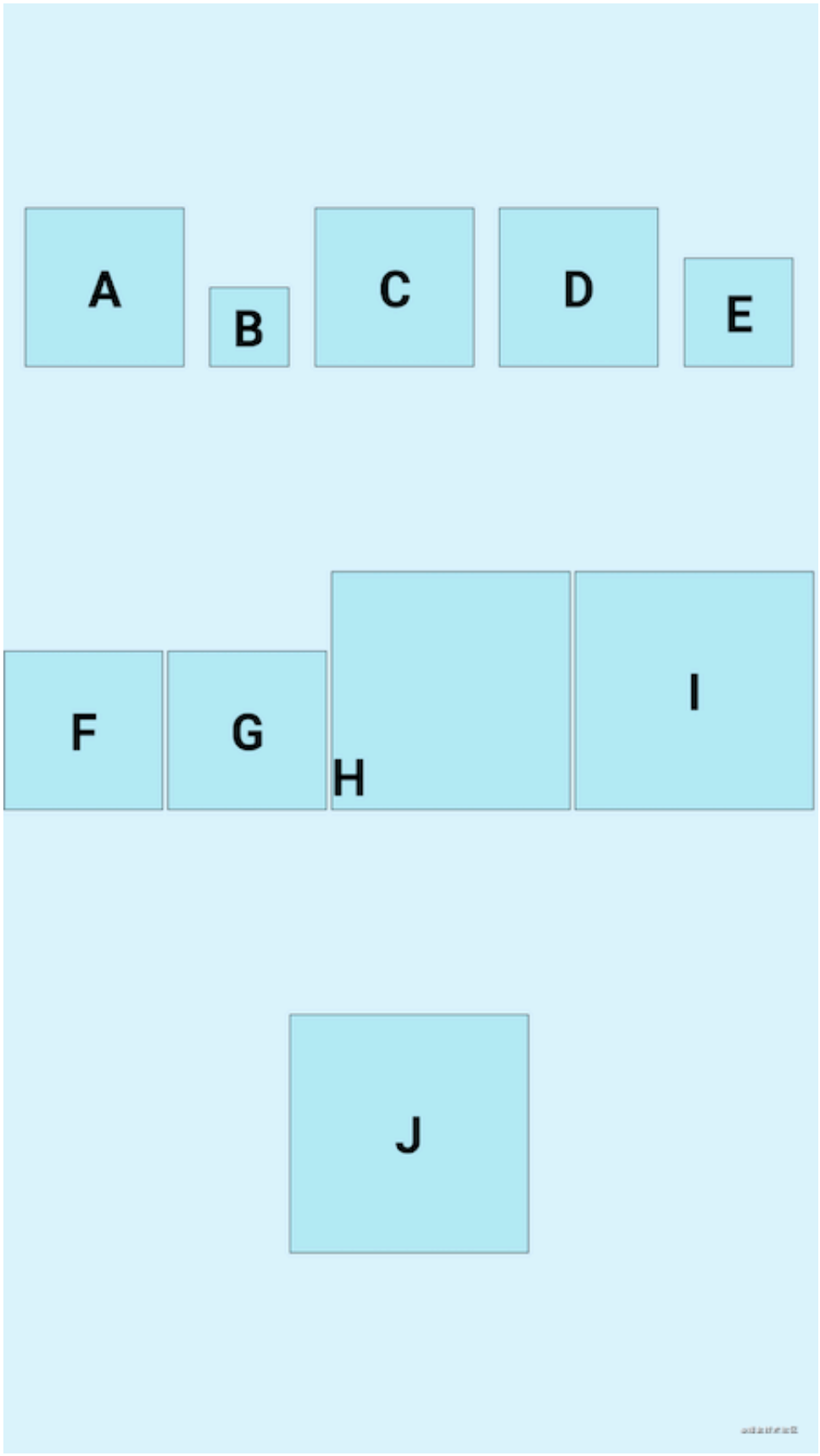
horizontal 水平排列

- **top**


A
B
C
D
E
F
G
H
I
J

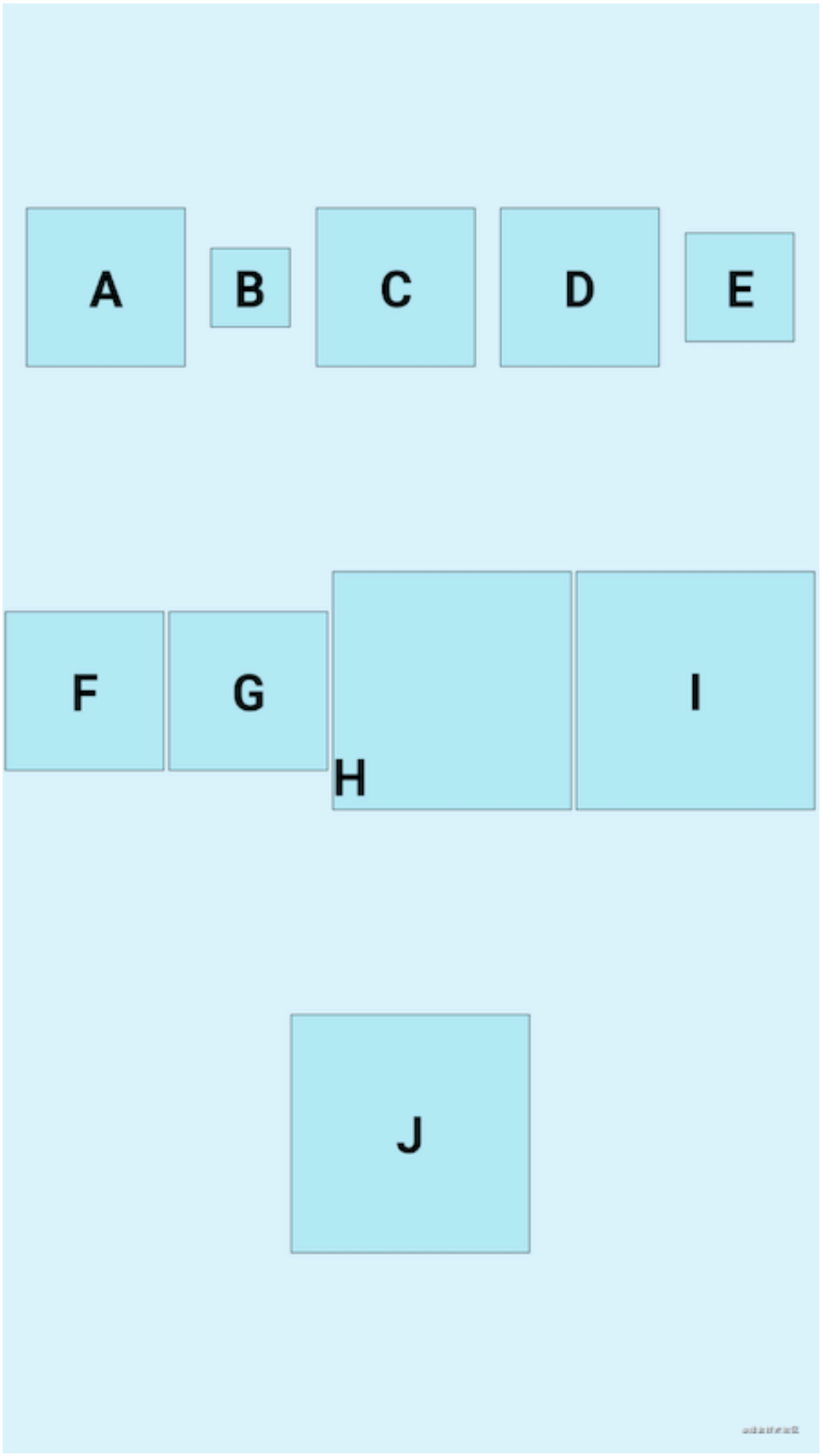
```
<androidx.constraintlayout.helper.widget.Flow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        app:constraint_referenced_ids="A,B,C,D,E,F,G,H,I,J"
        app:flow_verticalAlign="top"
        app:flow_wrapMode="chain"
        app:layout_constraintTop_toTopOf="parent" />
```

- **bottom**


A
B
C
D
E
F
G
H
I
J

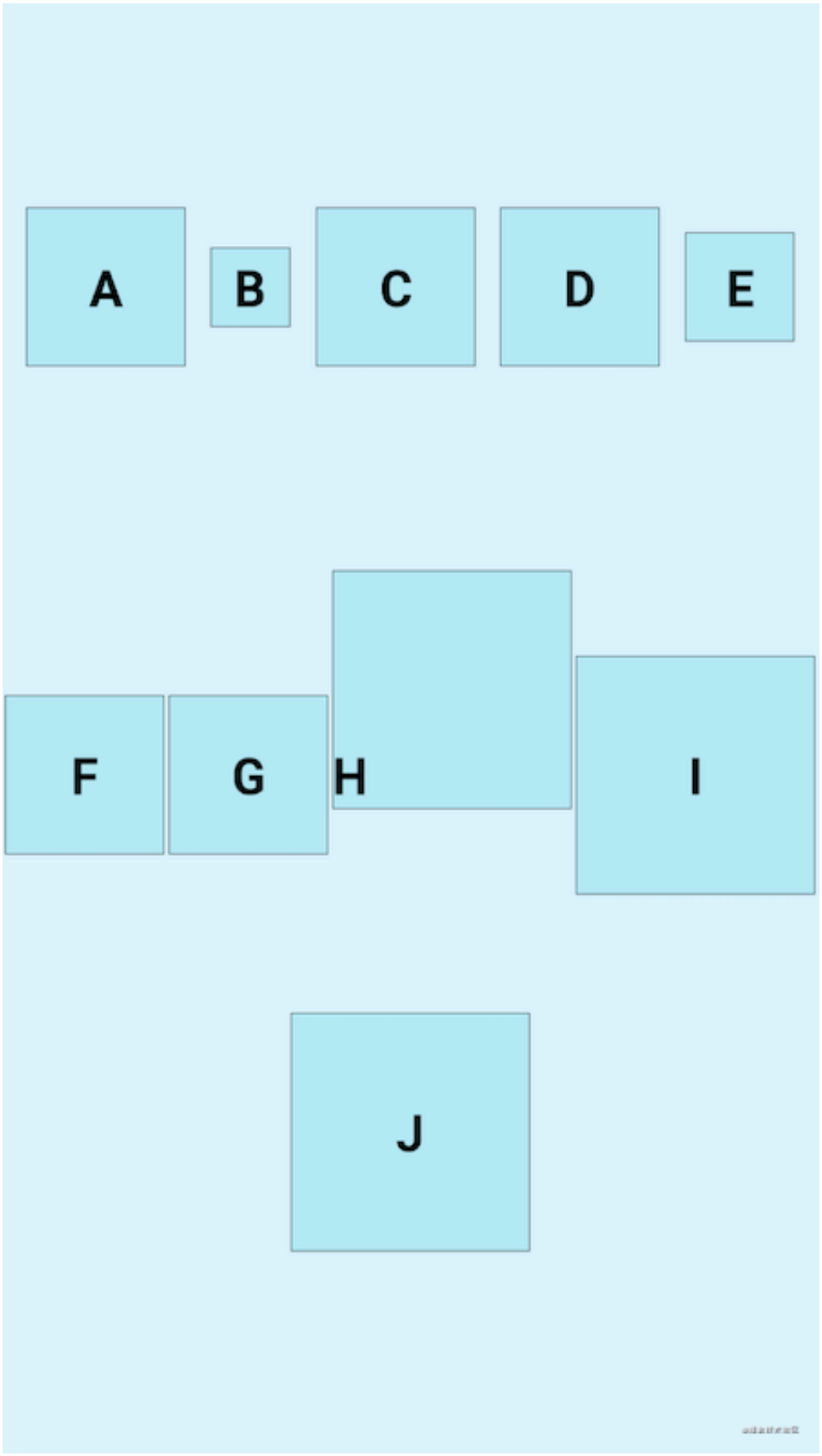
```
<androidx.constraintlayout.helper.widget.Flow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        app:constraint_referenced_ids="A,B,C,D,E,F,G,H,I,J"
        app:flow_verticalAlign="bottom"
        app:flow_wrapMode="chain"
        app:layout_constraintTop_toTopOf="parent" />
```

- **center**


A
B
C
D
E
F
G
H
I
J

```
<androidx.constraintlayout.helper.widget.Flow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        app:constraint_referenced_ids="A,B,C,D,E,F,G,H,I,J"
        app:flow_verticalAlign="center"
        app:flow_wrapMode="chain"
        app:layout_constraintTop_toTopOf="parent" />
```

- baseline

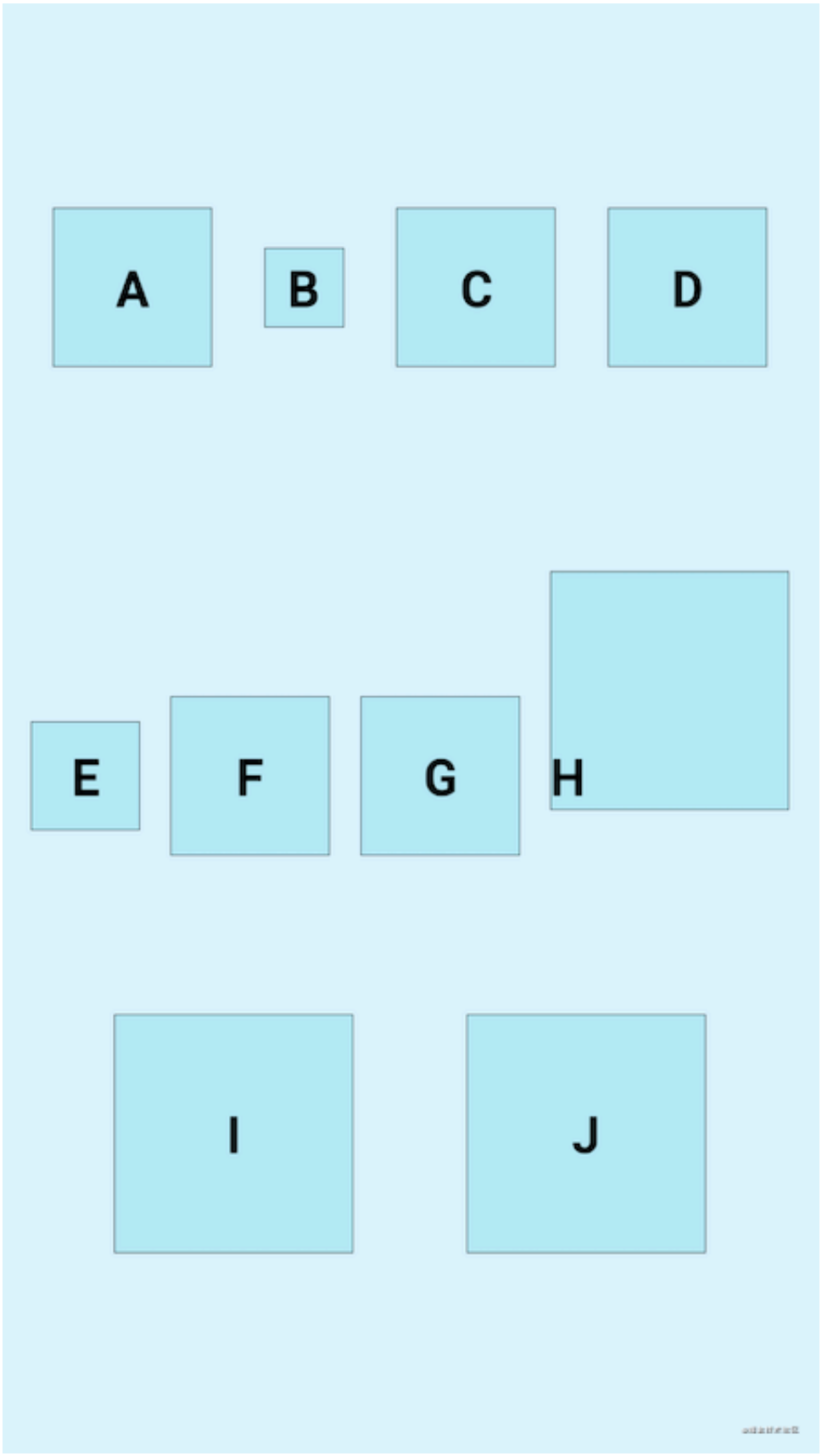

A
B
C
D
E
F
G
H
I
J

```
<androidx.constraintlayout.helper.widget.Flow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        app:constraint_referenced_ids="A,B,C,D,E,F,G,H,I,J"
        app:flow_verticalAlign="baseline"
        app:flow_wrapMode="chain"
        app:layout_constraintTop_toTopOf="parent" />
```

> 垂直方向排列这里就不再作过多的展示了

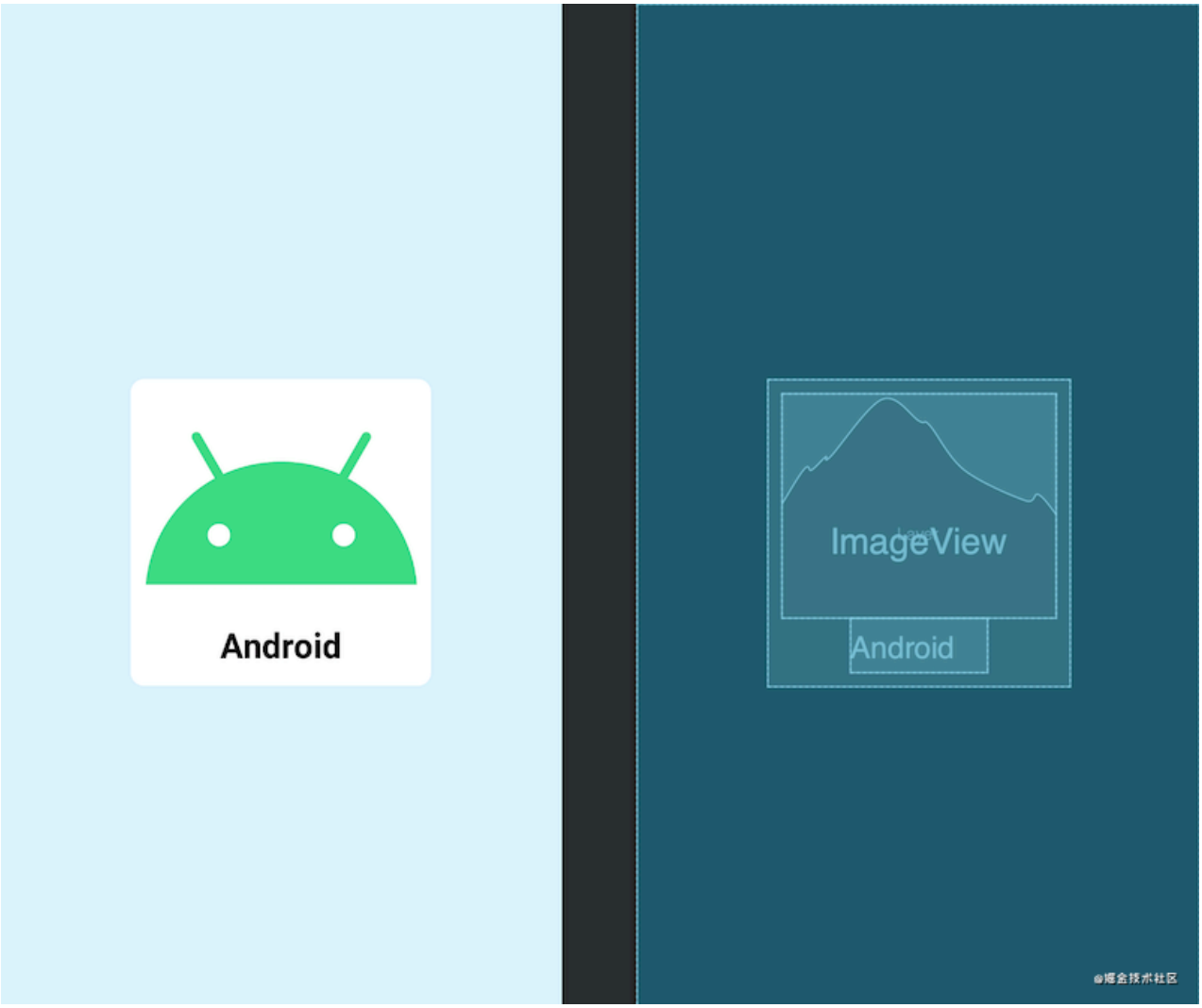
2.5.3 数量约束

> 当 `flow_wrapMode` 属性为 `aligned` 和 `chian` 时，通过 `flow_maxElementsWrap` 属性控制每行最大的子 `View` 数量，例如我们设置为 `flow_maxElementsWrap=4`，效果图如下：


A
B
C
D
E
F
G
H
I
J

2.6 Layer（层布局）

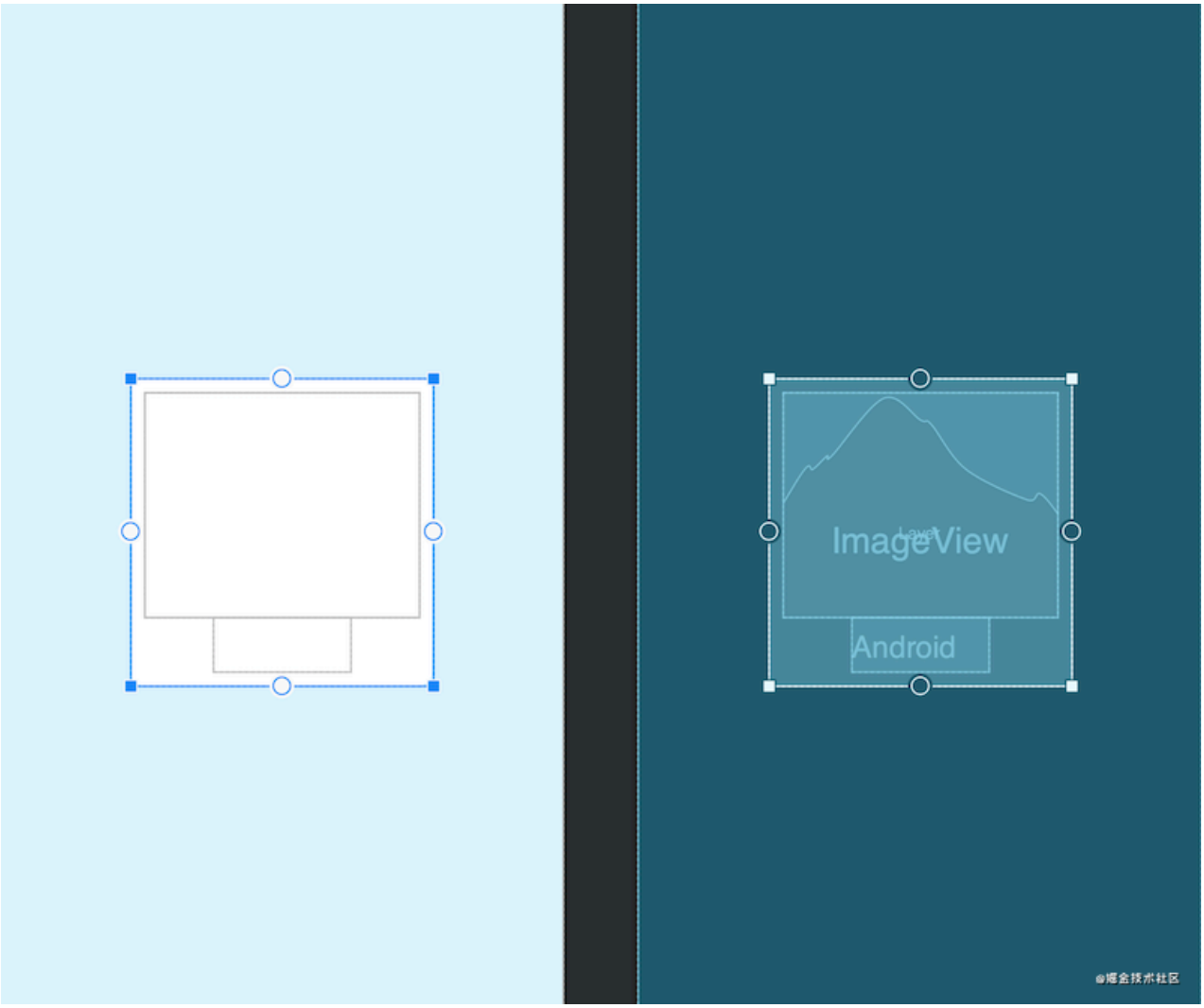
> `Layer`继承自`ConstraintHelper`，是一个约束助手，相对于`Flow`来说，`Layer`的使用较为简单，常用来增加背景，或者共同动画，图层（`Layer`）在布局期间会调整大小，其大小会根据其引用的所有视图进行调整，代码的先后顺序也会决定着它的位置，如果代码在所有引用`view`的最后面，那么它就会在所有`view`的最上面，反之则是最下面，在最上面的时候如果添加背景，就会把引用的`view`覆盖掉，下面展示下添加背景的例子，做动画的例子这里不再展示了




```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

```

```
    <androidx.constraintlayout.helper.widget.Layer
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/common_rect_white_100_10"
        android:padding="10dp"
        app:constraint_referenced_ids="AndroidImg,NameTv" />

    <ImageView
        android:id="@+id/AndroidImg"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:src="@drawable/android"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/NameTv"
        android:layout_width="100dp"
        android:layout_height="40dp"
        android:gravity="center"
        android:text="Android"
        android:textColor="@color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="@id/AndroidImg"
        app:layout_constraintStart_toStartOf="@id/AndroidImg"
        app:layout_constraintTop_toBottomOf="@id/AndroidImg" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

可以看到，当 `Layer` 的代码在所有引用 `view` 的上面时，效果是正常的，因为此时所有的 `view` 都在 `Layer` 的上面，下面我们来看一下 `Layer` 代码在最后面时的情况：




```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <ImageView
        android:id="@+id/AndroidImg"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:src="@drawable/android"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
20
21      <TextView
22          android:id="@+id/NameTv"
23          android:layout_width="100dp"
24          android:layout_height="40dp"
25          android:gravity="center"
26          android:text="Android"
27          android:textColor="@color/black"
28          android:textSize="25sp"
29          android:textStyle="bold"
30          app:layout_constraintEnd_toEndOf="@id/AndroidImg"
31          app:layout_constraintStart_toStartOf="@id/AndroidImg"
32          app:layout_constraintTop_toBottomOf="@id/AndroidImg" />
33
34      <androidx.constraintlayout.helper.widget.Layer
35          android:layout_width="match_parent"
36          android:layout_height="match_parent"
37          android:background="@drawable/common_rect_white_100_10"
38          android:padding="10dp"
39          app:constraint_referenced_ids="AndroidImg,NameTv" />
40
41  </androidx.constraintlayout.widget.ConstraintLayout>
```

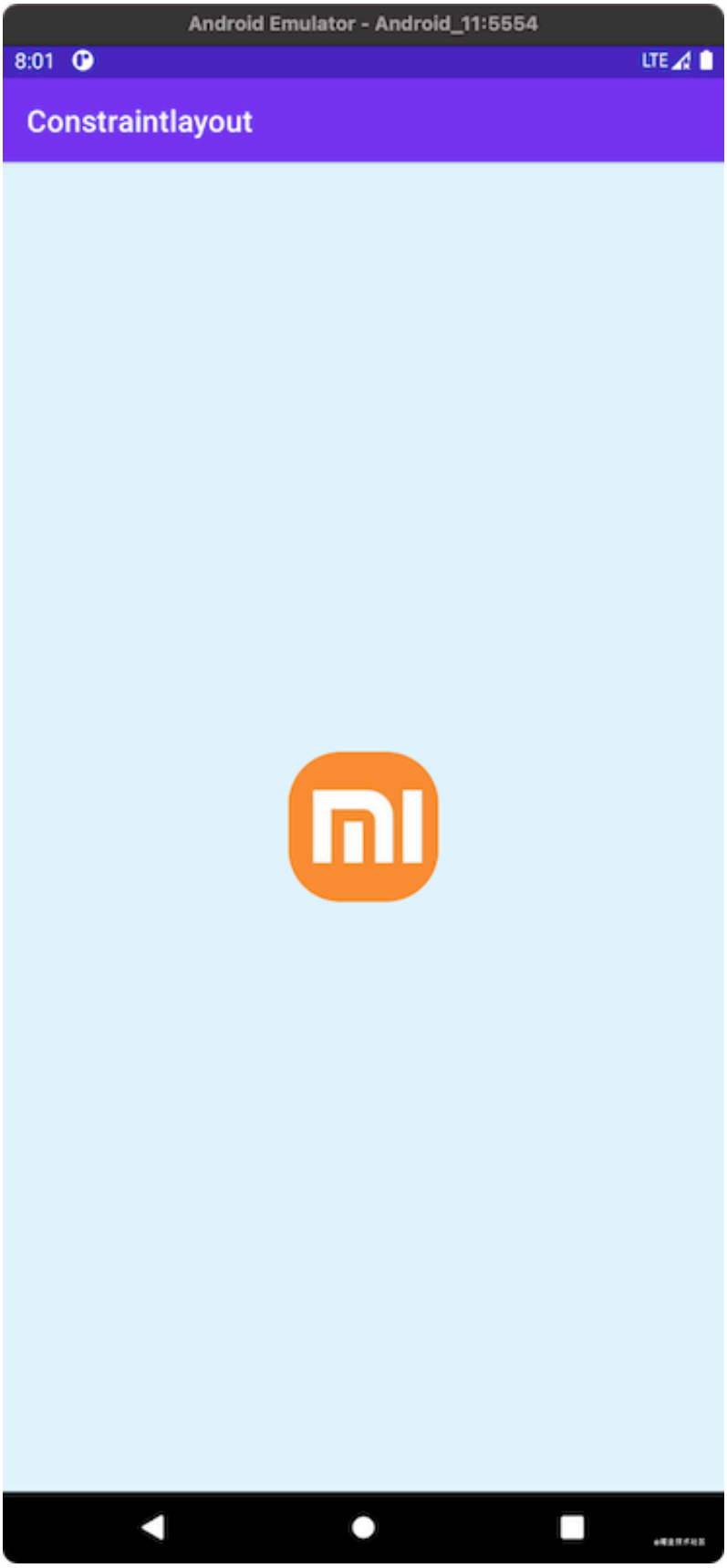
我们可以看到，此时 `Layer` 已经把所有的 `view` 覆盖住了

2.7 ImageFilterButton & ImageFilterView

`ImageFilterButton` 和 `ImageFilterView` 是两个控件，他们之间的关系就和 `ImageButton` 与 `ImageView` 是一样的，所以这里就只拿 `ImageFilterView` 来做讲解。从名字上来看，它们的定位是和过滤有关系的，它们的大致作用有两部分，一是可以用来做圆角图片，二是可以叠加图片资源进行混合过滤，下面一一展示：

2.7.1 圆角图片

`ImageFilterButton` 和 `ImageFilterView` 可以使用两个属性来设置图片资源的圆角，分别是 `roundPercent` 和 `round`， `roundPercent` 接受的值类型是0-1的小数，根据数值的大小会使图片在方形和圆形之间按比例过度， `round=` 可以设置具体圆角的大小，我在使用的过程中发现我的 `AndroidStudio`，没有这两个属性的代码提示，也没有预览效果，但是运行起来是有效果的，可能是没有做好优化吧。最近很热门的一个话题，小米花费200万设计的新logo，我们拿来做做例子：




```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
```

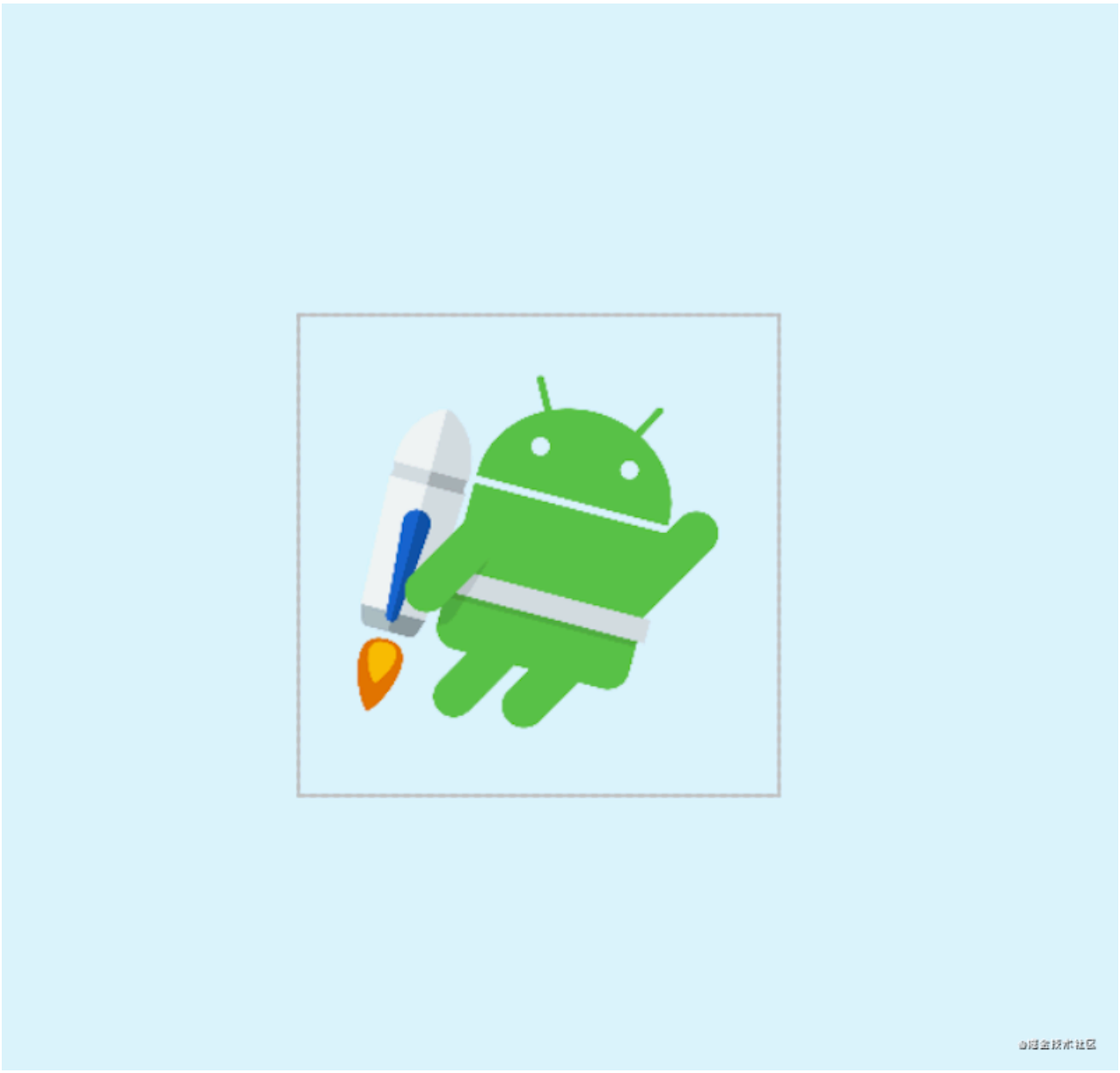
```
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="#DAF3FE"
8       tools:context=".MainActivity"
9       tools:ignore="HardcodedText">
10
11      <androidx.constraintlayout.utils.widget.ImageFilterView
12          android:layout_width="100dp"
13          android:layout_height="100dp"
14          android:src="@drawable/mi"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          app:roundPercent="0.7" />
20
21  </androidx.constraintlayout.widget.ConstraintLayout>
```

> 虽然和小米新logo的圆弧不太一样，不过这也不是我们考虑的地方，可以看到我们使用`roundPercent`设置了圆角为0.7（70%），实现一个圆角图片就是如此简单。

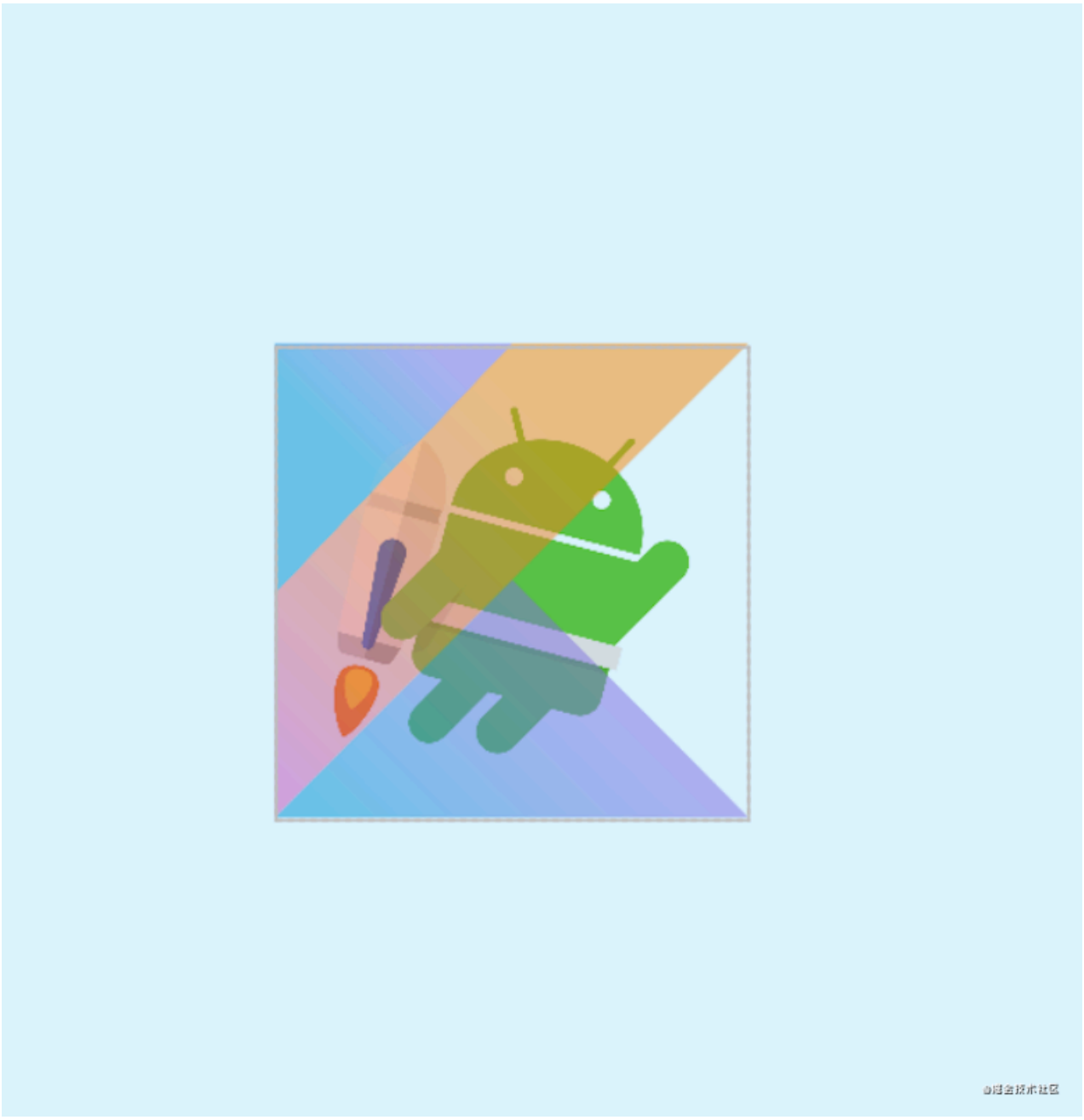
2.7.2 图片过滤

> `ImageFilterButton`和`ImageFilterView`不但可以使用`src`来设置图片资源，还可以使用`altSrc`来设置第二个图片资源，`altSrc`提供的资源将会和`src`提供的资源通过`crossfade`属性形成交叉淡化效果,默认情况下，`crossfade=0`，`altSrc`所引用的资源不可见，取值在0-1。下面看例子：

- **crossfade=0**

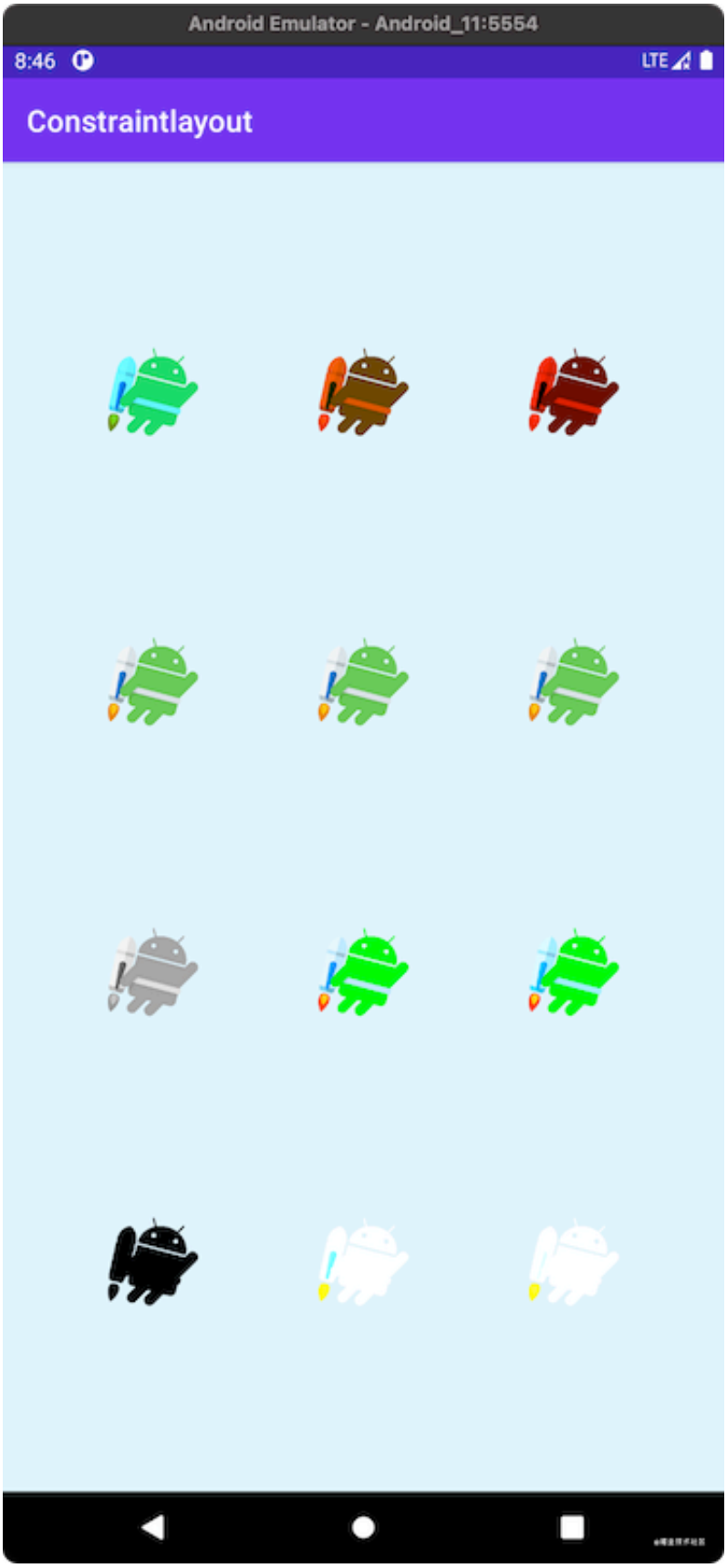


- **crossfade=0.5**



- **crossfade=1**

除此之外，`warmth`属性可以用来调节色温，`brightness`属性用来调节亮度，`saturation`属性用来调节饱和度，`contrast`属性用来调节对比度，下面展示一下各自属性和取值的效果：




```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAF3FE"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <androidx.constraintlayout.utils.widget.ImageFilterView
        android:id="@+id/view1"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:src="@drawable/jetpack"
        app:layout_constraintBottom_toBottomOf="@id/view2"
        app:layout_constraintEnd_toStartOf="@+id/view2"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@id/view2"
        app:warmth="0" />

    <androidx.constraintlayout.utils.widget.ImageFilterView
        android:id="@+id/view2"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:src="@drawable/jetpack"
        app:layout_constraintBottom_toTopOf="@+id/view5"
        app:layout_constraintEnd_toStartOf="@+id/view3"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/view1"
        app:layout_constraintTop_toTopOf="parent"
        app:warmth="5" />

    <androidx.constraintlayout.utils.widget.ImageFilterView
        android:id="@+id/view3"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:src="@drawable/jetpack"
        app:layout_constraintBottom_toBottomOf="@id/view2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0"
        app:layout_constraintStart_toEndOf="@+id/view2"
        app:layout_constraintTop_toTopOf="@id/view2"
        app:warmth="9"
        tools:layout_editor_absoluteY="0dp" />

    <androidx.constraintlayout.utils.widget.ImageFilterView
        android:id="@+id/view4"
```

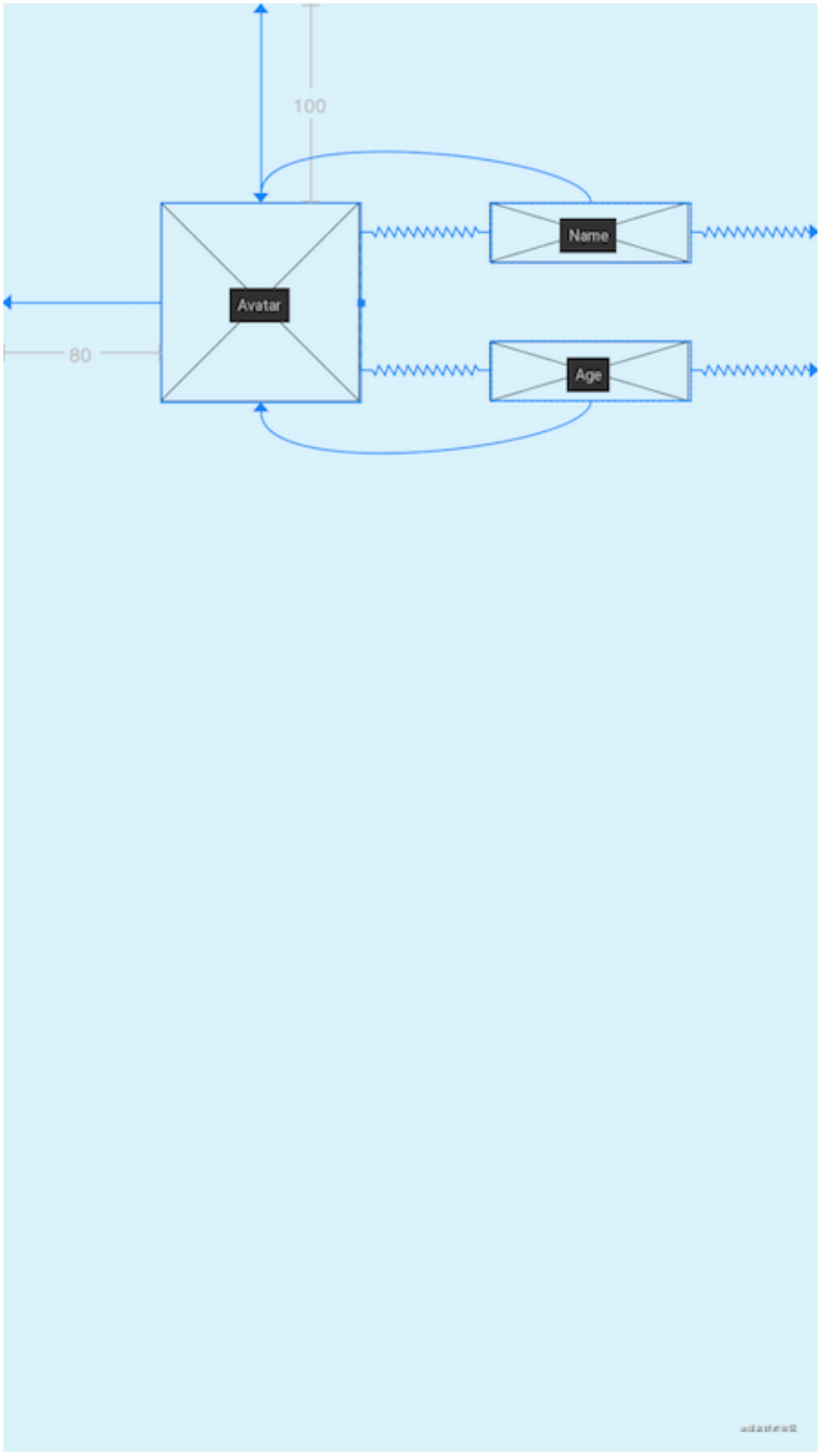
```
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:brightness="0"
            app:layout_constraintBottom_toBottomOf="@id/view5"
            app:layout_constraintEnd_toStartOf="@+id/view5"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@id/view5"
            tools:layout_editor_absoluteY="136dp" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view5"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:brightness="5"
            app:layout_constraintBottom_toTopOf="@+id/view8"
            app:layout_constraintEnd_toStartOf="@+id/view6"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toEndOf="@+id/view4"
            app:layout_constraintTop_toBottomOf="@+id/view2" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view6"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:brightness="9"
            app:layout_constraintBottom_toBottomOf="@id/view5"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toEndOf="@+id/view5"
            app:layout_constraintTop_toTopOf="@id/view5"
            tools:layout_editor_absoluteY="136dp" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view7"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:layout_constraintBottom_toBottomOf="@id/view8"
            app:layout_constraintEnd_toStartOf="@+id/view8"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@id/view8"
```

```
            app:saturation="0"
            tools:layout_editor_absoluteY="285dp" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view8"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:layout_constraintBottom_toTopOf="@+id/view11"
            app:layout_constraintEnd_toStartOf="@+id/view9"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toEndOf="@+id/view7"
            app:layout_constraintTop_toBottomOf="@+id/view5"
            app:saturation="5" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view9"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:layout_constraintBottom_toBottomOf="@id/view8"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toEndOf="@+id/view8"
            app:layout_constraintTop_toTopOf="@id/view8"
            app:saturation="9"
            tools:layout_editor_absoluteY="296dp" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view10"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:contrast="0"
            app:layout_constraintBottom_toBottomOf="@id/view11"
            app:layout_constraintEnd_toStartOf="@+id/view11"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@id/view11"
            tools:layout_editor_absoluteY="437dp" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view11"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
```

```
            app:contrast="5"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/view12"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toEndOf="@+id/view10"
            app:layout_constraintTop_toBottomOf="@+id/view8" />

        <androidx.constraintlayout.utils.widget.ImageFilterView
            android:id="@+id/view12"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/jetpack"
            app:contrast="9"
            app:layout_constraintBottom_toBottomOf="@id/view11"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toEndOf="@+id/view11"
            app:layout_constraintTop_toTopOf="@id/view11"
            tools:layout_editor_absoluteY="437dp" />

    </androidx.constraintlayout.widget.ConstraintLayout>
```

2.8 MockView

> 你家产品经理经常会给你画原型图，但这绝对不是他们的专属，我们也有自己的原型图，一个成熟的程序员要学会给自己的产品经理画大饼，我们可以使用`MockView`来充当原型图，下面看例子：




```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="#DAF3FE"
8       tools:context=".MainActivity"
9       tools:ignore="HardcodedText">
10
11      <androidx.constraintlayout.utils.widget.MockView
12          android:id="@+id/Avatar"
13          android:layout_width="100dp"
14          android:layout_height="100dp"
15          android:layout_marginStart="80dp"
16          android:layout_marginTop="100dp"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent" />
19
20      <androidx.constraintlayout.utils.widget.MockView
21          android:id="@+id/Name"
22          android:layout_width="100dp"
23          android:layout_height="30dp"
24          app:layout_constraintEnd_toEndOf="parent"
25          app:layout_constraintStart_toEndOf="@id/Avatar"
26          app:layout_constraintTop_toTopOf="@id/Avatar" />
27
28      <androidx.constraintlayout.utils.widget.MockView
29          android:id="@+id/Age"
30          android:layout_width="100dp"
31          android:layout_height="30dp"
32          app:layout_constraintBottom_toBottomOf="@id/Avatar"
33          app:layout_constraintEnd_toEndOf="parent"
34          app:layout_constraintStart_toEndOf="@id/Avatar" />
35
36  </androidx.constraintlayout.widget.ConstraintLayout>
```

> 嗯！有内味了！

2.9 ConstraintProperties（流式API）

> 2.0 提供了 `ConstraintProperties` 可以使用 `流式 API` 修改属性

```
val properties = ConstraintProperties(findViewById(R.id.image))
    properties.translationZ(32f)
          .margin(ConstraintSet.START, 43)
          .apply()
```

三. MotionLayout

> `Motion Layout`是`Constraint Layout 2.0`中最令人期待的功能之一。它提供了一个丰富的动画系统来协调多个视图之间的动画效果。`MotionLayout`基于`ConstraintLayout`，并在其之上进行了扩展，允许您在多组约束(或者`ConstraintSets`)之间进行动画的处理。您可以对视图的移动、滚动、缩放、旋转、淡入淡出等一系列动画行为进行自定义，甚至可以定义各个动画本身的自定义属性。它还可以处理手势操作所产生的物理移动效果，以及控制动画的速度。使用`MotionLayout`构建的动画是可追溯且可逆的，这意味着您可以随意切换到动画过程中任意一个点，甚至可以倒着执行动画效果。`Android Studio`集成了`Motion Editor`(动作编辑器)，可以利用它来操作`MotionLayout`对动画进行生成、预览和编辑等操作。这样一来，在协调多个视图的动画时，就可以做到对各个细节进行精细操控。由于我自己也没有用过，且说起来篇幅也挺大，这里就不再讲解`MotionLayout`（主要是我也不会）

结语

> 至此，关于**Constraint Layout**的内容基本已经介绍完毕，因为内容较多，代码示例代码和图片也比较多，一次性看完实属不易，可以点击收藏供以后翻阅，写这篇文章我是经历了无数次放弃和重新拾起，内容确实太多了，再加上也已经有很多不错的博文来介绍**Constraint Layout**，但是他们的肯定没有我的全！😂
> 如有错误请及时联系我，我会尽快修改更正。

参考

- 新小梦：Constraintlayout 2.0：你们要的更新来了
- 谷歌开发者：Constraint Layout 2.0 用法详解
- constraintlayout网站