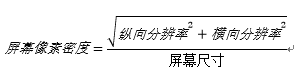
Android-屏幕适配

# 核心概念与单位详解

## 屏幕尺寸、屏幕分辨率、屏幕像素密度

**屏幕尺寸**是指屏幕对角线的长度。单位是英寸，1英寸=2.54厘米，用尺子度量得到。  
**屏幕分辨率**是指在横纵向上的像素点数，单位是px，1px=1像素点，一般是纵向像素横向像素，如1280×720，计数得到。代表纵（横）向可以分辨出多少像素。  
**屏幕像素密度**是指每英寸上的像素点数，单位是dpi，即“dot per inch”的缩写，像素密度和屏幕尺寸和屏幕分辨率有关



屏幕像素密度计算公式

**例如**：计算Nexus5的屏幕像素密度：  
屏幕尺寸：4.95inch、分辨率：1920×1080，屏幕像素密度：445

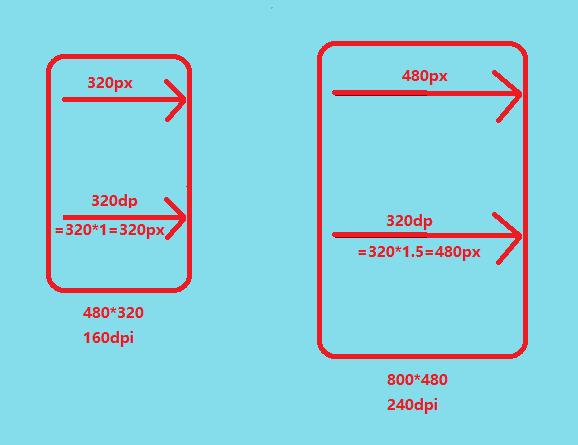
http://upload-images.jianshu.io/upload_images/1099585-d6b4e0e15ca040d8.PNG?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

代码获取屏幕分辨信息

|  |
| --- |
| public void getScreenDensity\_ByResources() {  DisplayMetrics mDisplayMetrics = getResources().getDisplayMetrics();  Log.*d*(TAG, "getScreenDensity\_ByResources,Screen mDisplayMetrics: " + mDisplayMetrics); } |

## dp、dip、dpi、sp、px？之间关系是什么？

**dpi：**屏幕像素密度的单位，“dot per inch”的缩写  
**px：**像素，物理上的绝对单位  
**dip（dp）：**Density Independent Pixels（密度无关像素）的缩写。在**160dpi**条件下，1dip=1px; 如果密度是320dpi，则1dip=2px，以此类推。假如同样都是画一条320px的线，在480\*800分辨率手机上显示为2/3屏幕宽度，在320\*480的手机上则占满了全屏，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。这也是为什么在Android开发中，写布局的时候要尽量使用dp而不是px的原因。  
**sp：**Scale-Independent Pixels的缩写，可以根据文字大小首选项自动进行缩放。Google推荐我们使用12sp以上的大小，通常可以使用12sp，14sp，18sp，22sp，最好不要使用奇数和小数。



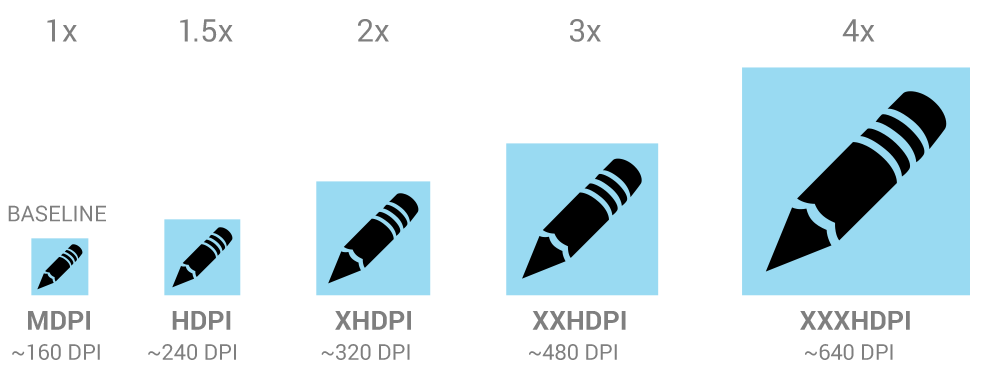
dp的特点

**说明：**如果A设备的参数为480×320，160dpi，B设置的参数为800×480，240dpi。我们要画出一条和屏幕宽度一样长的直线，如果使用px作为单位，必须在A设备上设置为320px，在B设备上设置480px。但是如果我们使用dp作为单位，由于以160dpi为基准，1dp=1px，所以A设备上设置为320dp就等于屏幕宽度（320px），在B设备上设置为320dp就等于320×（240/160）=480px，即B设备的屏幕宽度。这样，使用dp作为单位就可以实现简单的屏幕适配。这知识一种巧合，也有B设备的像素密度不是这样刚刚好的，就需要我们运用别的屏幕适配技术。

## mdpi、hdpi、xdpi、xxdpi、xxxdpi？如何计算和区分？

用于区分不同的像素密度。

| **名称** | **像素密度范围** | **图片大小** |
| --- | --- | --- |
| mdpi | 120dpi~160dpi | 48×48px |
| hdpi | 160dpi~240dpi | 72×72px |
| xhdpi | 240dpi~320dpi | 96×96px |
| xxhdpi | 320dpi~480dpi | 144×144px |
| xxxhdpi | 480dpi~640dpi | 192×192px |



说明：在设计图标时，对于五种主流的像素密度（MDPI、HDPI、XHDPI、XXHDPI 和 XXXHDPI）应按照 2:3:4:6:8 的比例进行缩放。例如，一个启动图标的尺寸为48x48 dp，这表示在 MDPI 的屏幕上其实际尺寸应为 48x48 px，在 HDPI 的屏幕上其实际大小是 MDPI 的 1.5 倍 (72x72 px)，在 XDPI 的屏幕上其实际大小是 MDPI 的 2 倍 (96x96 px)，依此类推。

用来修饰Android中的drawable文件夹及values文件夹，用来区分不同像素密度下的图片和dimen值。

# 解决方案-支持各种屏幕尺寸

我们可以通过以下几种方式来支持各种屏幕尺寸：

## 使用控件属性

### wrap\_content、math\_parent、weight

**wrap\_content：**根据控件的内容设置控件的尺寸  
**math\_parent：**根据父控件的尺寸大小设置控件的尺寸  
**weight：**权重，在线性布局中可以使用weight属性设置控件所占的比例  
我们可以通过以下几种方式来支持各种屏幕尺寸：

**小插曲：**关于android:layout\_weight属性

一般情况，我们都是设置要进行比例分配的方向的宽度为0dp，然后再用权重进行分配，而不使用其他方式？

公式：所占宽度=原来宽度+剩余空间所占百分比的宽度

如下：

<Button

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Button1" />

<Button

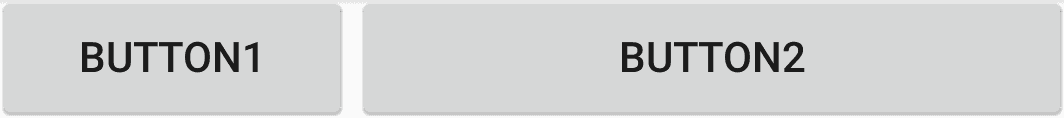
android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="2"

android:text="Button2" />

效果为：



宽度为0dp时，所占比例

设屏幕宽度为L，  
根据公式，  
button1宽度=0+L×1/（1+2）=1/3L  
button2宽度=0+L×2/（1+2）=2/3L  
但如果设置为match\_parent：

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Button1" />

<Button

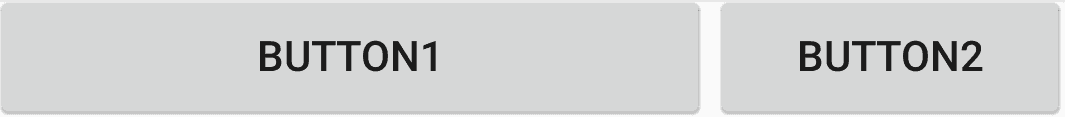
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="2"

android:text="Button2" />

效果为：



宽度为match\_parent时，所占比例

button1宽度=L+（L-2L）×1/3=2/3L  
button2宽度=L+（L-2L）×2/3=1/3L

原理：所有的子view测量结束后，才开始对layout\_weight计算，这样我们可能想到，如果父view已经被占满了， 那么有可能layout\_weight大于0的view对象是不会显示的，而计算layout\_weight的方法也很简单，就是用总高度减去上面分析完 mTotalLength的值，就是剩下，然后去平分给view对象，注意计算权重时优先去 android:android:weightSum（LinearLayout的xml属性）的值，如果不设置该值会计算和，所以该值既然设置了，就一 定要子view的weight的总和相等，否则平分可能不能得到预期效果。

### ImageView的ScaleType属性

设置不同的ScaleType会得到不同的显示效果，一般情况下，设置为CENTER\_INSIDE能获得较好的适配效果。

FIT\_XY：对原图宽高进行放缩，该放缩不保持原比例来填充满ImageView。

MATRIX：不改变原图大小从ImageView的左上角开始绘制，超过ImageView部分不再显示。

CENTER：对原图居中显示，超过ImageView部分不再显示。

CENTER\_CROP：对原图居中显示后进行等比放缩处理，使原图最小边等于ImageView的相应边。

CENTER\_INSIDE：若原图宽高小于ImageView宽高，这原图不做处理居中显示，否则按比例放缩原图宽(高)是之等于ImageView的宽(高)。

FIT\_START：对原图按比例放缩使之等于ImageView的宽高，若原图高大于宽则左对齐否则上对其。

FIT\_CENTER：对原图按比例放缩使之等于ImageView的宽高使之居中显示。

FIT\_END：对原图按比例放缩使之等于ImageView的宽高，若原图高大于宽则右对齐否则下对其。

## 使用相对布局，禁用绝对布局

简单的布局一般都使用线性布局，而略微复杂点的布局，我们使用相对布局，大多数时候，我们都是使用这两种布局的嵌套。

我们使用相对布局的原因是，相对布局能在各种尺寸的屏幕上保持控件间的相对位置。

## 使用限定符

### **使用尺寸**限定符

当我们要在大屏幕上显示不同的布局，就要使用large限定符。例如，在宽的屏幕左边显示列表右边显示列表项的详细信息，在一般宽度的屏幕只显示列表，不显示列表项的详细信息，我们就可以使用large限定符。需要注意的是，这种通过large限定符分辨屏幕尺寸的方法，适用于android3.2之前。在android3.2之后，为了更精确地分辨屏幕尺寸大小，Google推出了最小宽度限定符。如果这个程序运行在屏幕尺寸大于7inch的设备上，系统就会加载res/layout-large/main.xml 而不是res/layout/main.xml，在小于7inch的设备上就会加载res/layout/main.xml。

建议大家多看文档，官方说明：

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

### **使用最小宽度限定符**

最小宽度限定符的使用和large基本一致，只是使用了具体的宽度限定。注意这个宽度=min(W,H),单位是dp，注意换算。这种方式是不区分屏幕方向的.  
res/layout/main.xml，（默认）布局：  
res/layout-sw600dp/main.xml，布局： Small Width 最小宽度  
这就要求我们维护两个相同功能的文件。为了避免繁琐操作，我们就要使用布局别名。

### 使用布局别名

res/layout/main.xml: 单面板布局  
res/layout-large/main.xml: 多面板布局  
res/layout-sw600dp/main.xml: 多面板布局

由于后两个文具文件一样，我们可以用以下两个文件代替上面三个布局文件：

res/layout/main.xml 单面板布局  
res/layout/main\_twopanes.xml 双面板布局

然后在res下建立  
res/values/layout.xml、  
res/values-large/layout.xml、  
res/values-sw600dp/layout.xml三个文件。

默认布局  
res/values/layout.xml:

<resources>

<item name="main" type="layout">@layout/main</item>

</resources>

Android3.2之前的平板布局  
res/values-large/layout.xml:

<resources>

<item name="main" type="layout">@layout/main\_twopanes</item>

</resources>

Android3.2之后的平板布局  
res/values-sw600dp/layout.xml:

<resources>

<item name="main" type="layout">@layout/main\_twopanes</item>

</resources>

这样就有了main为别名的布局。  
在activity中setContentView(R.layout.main);

这样，程序在运行时，就会检测手机的屏幕大小，如果是平板设备就会加载res/layout/main\_twopanes.xml，如果是手机设备，就会加载res/layout/main.xml 。我们就解决了只使用一个布局文件来适配android3.2前后的所有平板设备。

### 使用屏幕方向限定符

如果我们要求给横屏、竖屏显示的布局不一样。就可以使用屏幕方向限定符来实现。  
例如，要在平板上实现横竖屏显示不用的布局，可以用以下方式实现。  
res/values-sw600dp-land/layouts.xml:横屏

<resources>

<item name="main" type="layout">@layout/main\_twopanes</item>

</resources>

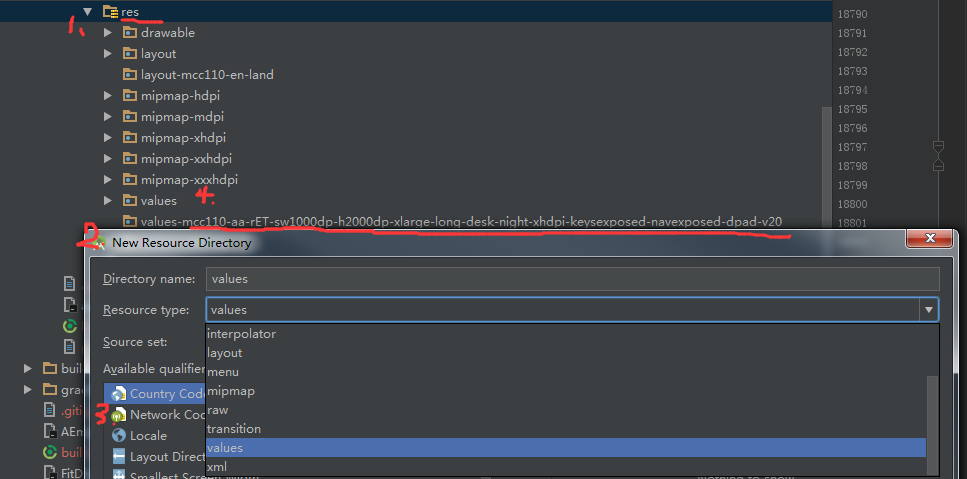
res/values-sw600dp-port/layouts.xml:竖屏

<resources>

<item name="main" type="layout">@layout/main</item>

</resources>

**AS给我们提供了丰富的限定符，可以解决大部分问题了**



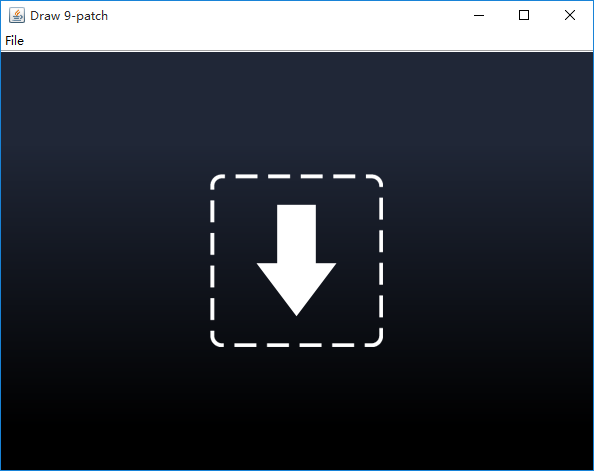
## 使用自动拉伸位图

自动拉伸位图，即android下特有的.9.png图片格式。

当我们需要使图片在拉伸后还能保持一定的显示效果，比如，不能使图片中的重要像素拉伸，不能使内容区域受到拉伸的影响，我们就可以使用.9.png图来实现。

方法1：双击SDK安装目录 sdk\tools\draw9patch.bat，就会打开下图所示的窗口。

方法2: 1把png图片重命名为.9.png;copy到drawable里; 在AS中打开图片



### 使用.9图片和不使用的差异

* [9patch](http://baike.baidu.com/link?url=hIV5poUxtU0ntjDyEZxxkB0IIKeiIEU5XIrIVisfOZjbifl1X4ayg_SU8NbXn3Hy6M_3AD9rn-4PDYhX8eC9sa)和一般图片的区别在于.9图片有四条黑边，而一般的图片没有，这四条黑边就是用来拉伸和指定显示位置的。
* 使用.9图片后，整个图片应该是包裹着你想要显示的内容的，而没有使用的话整个控件布局看起来特别糟糕。

使用.9图片，不使用.9图片

### 绘制.9图片

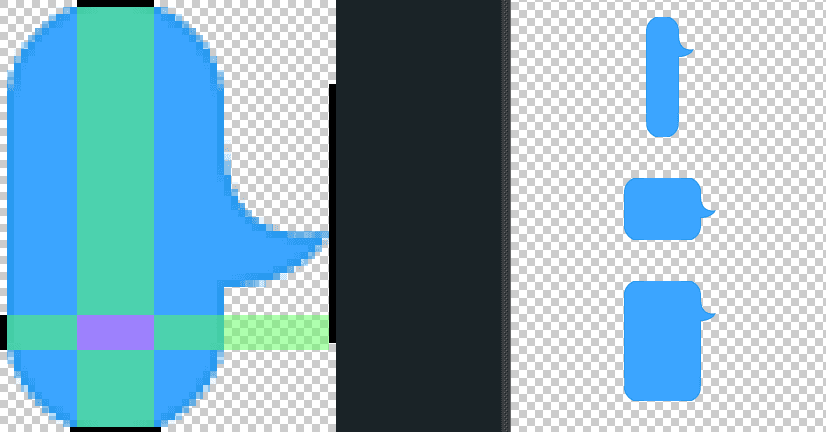
* 绘制之前先来说一下.9图片的四条黑边的意义，每条黑边的意义都不一样。

顶部：在水平拉伸的时候，保持其他位置不动，只在这个点的区域做无限的延伸（拷贝）

左边：在竖直拉伸的时候，保持其他位置不动，只在这个点的区域做无限的延伸（拷贝）

底部：在水平拉伸的时候，指定图片里的内容显示的区域

右边：在竖直拉伸的时候，指定图片里的内容显示的区域



绘制成图

.

### 注意

1.编辑后要在三个分辨率下都放置此图片，不然显示会有误。

2.关于在[Android Studio](http://www.android-studio.org/" \t "_blank)中使用.9图片出错解决方案

* 用过[Android Studio](http://www.android-studio.org/)的童鞋都知道，AS中对.9的图片增加了安全检查机制，你的.9图片只要有不规范的地方都会给你编译报错，本人刚开始用AS时就深受其苦，后来才找到了解决办法。
* 解决方式主要有两种：
  + 一是让AS取消掉对.9图片的安全检查。找到你app目录下的build.gradle文件，打开之后你可以在buildToolsVersion属性之下添加取消安全检查的两行代码。
  + // 取消掉系统对.9图片的检查
  + aaptOptions.cruncherEnabled = false

aaptOptions.useNewCruncher = false

* + 二是 .9图片编译报错，那肯定说明你的.9图片制作不完善，你可以检查一下图片哪里没绘制好。你可以检查一下是否有重复绘制黑边，或者有哪条边没有绘制。这里注意：AS中要求.9图片的四条边都会绘制。

# 平板

针对只在平板设备上使用的dimen，比较简单的办法是只定义在默认资源目录中，这样肯定不会导致crash，但如果在手机设备上引用了这些资源，界面可能会有显示上的问题。

## 用代码判断是否平板

判断的方法有很多种，但我们一直用的是sw600dp的方法，应该也利用这个方式来判断，这样就与资源的选择是一致的，不会导致资源使用了平板版本的，而代码的判断结果是手机设备。

<!-- 在 values-sw600dp/bool.xml 中定义一个 bool 值 -->

<resources>

<bool name="isPad">true</bool>

</resources>

<!-- 同时在 values/bool.xml 中将这个值定义为 false -->

<resources>

<bool name="isPad">false</bool>

</resources>

Java代码里可以这样判断：

public static boolean isPad(Context context) {

return context.getResources().getBoolean(R.bool.isPad);

}

横竖屏切换后，安卓并不会自动更新 resource，因此如果平板上横竖屏使用了不同的尺寸，就需要用代码重新设置一下。如下代码所示，在发生横竖屏切换时，更新左侧fragment的宽度：

@Override

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

if (isPad(this)) {

ViewGroup.LayoutParams layoutParams = mRecyclerView.getLayoutParams();

layoutParams.width = getResources().getDimensionPixelSize(R.dimen.left\_fragment\_width);

mRecyclerView.setLayoutParams(layoutParams);

}

}

都使用Fragment写界面，而不是直接写在Activity中。

## 判断分栏还是单栏

最直观的方式是判断是否是平板，但更好的方式是判断右侧详情栏是否存在，也就是说判断平板的layout右侧独有的ViewGroup是否存在。这样的好处是并不一定需要是平板上才可以分栏。

mRightFragmentContainer = v.findViewById(R.id.right\_fragment);

mTwoPanesMode = mRightFragmentContainer != null;

# setContentView原理

## 资源限定符优先级

### **表 2.**配置限定符名称。

|  |  |  |
| --- | --- | --- |
| 配置 | 限定符值 | 描述 |
| MCC 和 MNC | 示例： mcc310 mcc310-mnc004 mcc208-mnc00 等等 | 移动国家代码 (MCC)，（可选）后跟设备 SIM 卡中的移动网络代码 (MNC)。例如，mcc310 是指美国的任一运营商，mcc310-mnc004 是指美国的 Verizon 公司，mcc208-mnc00 是指法国的 Orange 公司。  如果设备使用无线电连接（GSM 手机），则 MCC 和 MNC 值来自 SIM 卡。  也可以单独使用 MCC（例如，将国家/地区特定的合法资源包括在应用中）。如果只需根据语言指定，则改用“语言和区域”限定符（稍后进行介绍）。 如果决定使用 MCC 和 MNC 限定符，请谨慎执行此操作并**[测试](http://lib.csdn.net/base/softwaretest" \t "_blank" \o "软件测试知识库)**限定符是否按预期工作。  另请参阅配置字段 [mcc](http://developer.android.com/reference/android/content/res/Configuration.html#mcc) 和 [mnc](http://developer.android.com/reference/android/content/res/Configuration.html" \l "mnc" \t "_blank)，这两个字段分别表示当前的移动国家代码和移动网络代码。 |
| 语言和区域 | 示例： en fr en-rUS fr-rFR fr-rCA 等等 | 语言通过由两个字母组成的 [ISO 639-1](http://www.loc.gov/standards/iso639-2/php/code_list.php) 语言代码定义，（可选）后跟两个字母组成的 [ISO 3166-1-alpha-2](http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html) 区域码（前带小写字母“r”）。  这些代码不区分大小写；r 前缀用于区分区域码。 不能单独指定区域。  如果用户更改系统设置中的语言，它有可能在应用生命周期中发生改变。 如需了解这会在运行期间给应用带来哪些影响，请参阅[处理运行时变更](http://developer.android.com/intl/zh-cn/guide/topics/resources/runtime-changes.html" \t "_blank)。  有关针对其他语言本地化应用的完整指南，请参阅[本地化](http://developer.android.com/intl/zh-cn/guide/topics/resources/localization.html" \t "_blank)。  另请参阅 [locale](http://developer.android.com/reference/android/content/res/Configuration.html#locale) 配置字段，该字段表示当前的区域设置。 |
| 布局方向 | ldrtl ldltr | 应用的布局方向。ldrtl 是指“布局方向从右到左”。ldltr 是指“布局方向从左到右”，这是默认的隐式值。  它适用于布局、图片或值等任何资源。  例如，若要针对阿拉伯语提供某种特定布局，并针对任何其他“从右到左”语言（如波斯语或希伯来语）提供某种通用布局，则可编码如下：  res/  layout/  main.xml (Default layout)  layout-ar/  main.xml (Specific layout for Arabic)  layout-ldrtl/  main.xml (Any "right-to-left" language, except  for Arabic, because the "ar" language qualifier  has a higher precedence.)  **注：**要为应用启用从右到左的布局功能，必须将 [supportsRtl](http://developer.android.com/guide/topics/manifest/application-element.html" \l "supportsrtl" \t "_blank) 设置为"true"，并将 [targetSdkVersion](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html" \l "target" \t "_blank) 设置为 17 或更高。  此项为API 级别 17 中新增配置。 |
| smallestWidth | sw<N>dp  示例： sw320dp sw600dp sw720dp 等等 | 屏幕的基本尺寸，由可用屏幕区域的最小尺寸指定。 具体来说，设备的 smallestWidth 是屏幕可用高度和宽度的最小尺寸（您也可以将其视为屏幕的“最小可能宽度”）。无论屏幕的当前方向如何，您均可使用此限定符确保应用 UI 的可用宽度至少为 <N>dp。  例如，如果布局要求屏幕区域的最小尺寸始终至少为 600dp，则可使用此限定符创建布局资源 res/layout-sw600dp/。仅当可用屏幕的最小尺寸至少为 600dp 时，系统才会使用这些资源，而不考虑 600dp 所代表的边是用户所认为的高度还是宽度。smallestWidth 是设备的固定屏幕尺寸特性；设备的 smallestWidth 不会随屏幕方向的变化而改变。  设备的 smallestWidth 将屏幕装饰元素和系统 UI 考虑在内。例如，如果设备的屏幕上有一些永久性 UI 元素占据沿 smallestWidth 轴的空间，则系统会声明 smallestWidth 小于实际屏幕尺寸，因为这些屏幕像素不适用于您的 UI。因此，使用的值应该是布局所需要的实际最小尺寸（通常，无论屏幕的当前方向如何，此值都是布局支持的“最小宽度”）。  以下是一些可用于普通屏幕尺寸的值：   * 320，适用于屏幕配置如下的设备：   + 240x320 ldpi（QVGA 手机）   + 320x480 mdpi（手机）   + 480x800 hdpi（高密度手机） * 480，适用于 480x800 mdpi 之类的屏幕（平板电脑/手机）。 * 600，适用于 600x1024 mdpi 之类的屏幕（7 英寸平板电脑）。 * 720，适用于 720x1280 mdpi 之类的屏幕（10 英寸平板电脑）。   应用为多个资源目录提供不同的 smallestWidth 限定符值时，系统会使用最接近（但未超出）设备 smallestWidth 的值。  此项为 API 级别 13 中新增配置。  另请参阅 [android:requiresSmallestWidthDp](http://developer.android.com/guide/topics/manifest/supports-screens-element.html" \l "requiresSmallest" \t "_blank) 属性和[smallestScreenWidthDp](http://developer.android.com/reference/android/content/res/Configuration.html" \l "smallestScreenWidthDp" \t "_blank) 配置字段，前者声明与应用兼容的最小 smallestWidth；后者存放设备的 smallestWidth 值。  如需了解有关设计不同屏幕和使用此限定符的详细信息，请参阅[支持多个屏幕](http://developer.android.com/guide/practices/screens_support.html)开发者指南。 |
| 可用宽度 | w<N>dp  示例： w720dp w1024dp 等等 | 指定资源应该使用的最小可用屏幕宽度，以 dp 为单位，由 <N> 值定义。在横向和纵向之间切换时，为了匹配当前实际宽度，此配置值也会随之发生变化。  应用为多个资源目录提供不同的此配置值时，系统会使用最接近（但未超出）设备当前屏幕宽度的值。 此处的值考虑到了屏幕装饰元素，因此如果设备显示屏的左边缘或右边缘上有一些永久性 UI 元素，考虑到这些 UI 元素，它会使用小于实际屏幕尺寸的宽度值，这样会减少应用的可用空间。  此项为 API 级别 13 中新增配置。  另请参阅 [screenWidthDp](http://developer.android.com/reference/android/content/res/Configuration.html" \l "screenWidthDp" \t "_blank) 配置字段，该字段存放当前屏幕宽度。  如需了解有关设计不同屏幕和使用此限定符的详细信息，请参阅[支持多个屏幕](http://developer.android.com/guide/practices/screens_support.html)开发者指南。 |
| 可用高度 | h<N>dp  示例： h720dp h1024dp 等等 | 指定资源应该使用的最小可用屏幕高度，以“dp”为单位，由 <N> 值定义。 在横向和纵向之间切换时，为了匹配当前实际高度，此配置值也会随之发生变化。  应用为多个资源目录提供不同的此配置值时，系统会使用最接近（但未超出）设备当前屏幕高度的值。 此处的值考虑到了屏幕装饰元素，因此如果设备显示屏的上边缘或下边缘有一些永久性 UI 元素，考虑到这些 UI 元素，同时为减少应用的可用空间，它会使用小于实际屏幕尺寸的高度值。 非固定的屏幕装饰元素（例如，全屏时可隐藏的手机状态栏）并不在考虑范围内，标题栏或操作栏等窗口装饰也不在考虑范围内，因此应用必须准备好处理稍小于其所指定值的空间。  此项为 API 级别 13 中新增配置。  另请参阅 [screenHeightDp](http://developer.android.com/reference/android/content/res/Configuration.html" \l "screenHeightDp" \t "_blank) 配置字段，该字段存放当前屏幕宽度。  如需了解有关设计不同屏幕和使用此限定符的详细信息，请参阅[支持多个屏幕](http://developer.android.com/guide/practices/screens_support.html)开发者指南。 |
| 屏幕尺寸 | small normal large xlarge | * small：尺寸类似于低密度 QVGA 屏幕的屏幕。小屏幕的最小布局尺寸约为 320x426 dp 单位。例如，QVGA 低密度屏幕和 VGA 高密度屏幕。 * normal：尺寸类似于中等密度 HVGA 屏幕的屏幕。标准屏幕的最小布局尺寸约为 320x470 dp 单位。例如，WQVGA 低密度屏幕、HVGA 中等密度屏幕、WVGA 高密度屏幕。 * large：尺寸类似于中等密度 VGA 屏幕的屏幕。 大屏幕的最小布局尺寸约为 480x640 dp 单位。 例如，VGA 和 WVGA 中等密度屏幕。 * xlarge：明显大于传统中等密度 HVGA 屏幕的屏幕。超大屏幕的最小布局尺寸约为 720x960 dp 单位。在大多数情况下，屏幕超大的设备体积过大，不能放进口袋，最常见的是平板式设备。 此项为 API 级别 9 中新增配置。   **注：**使用尺寸限定符并不表示资源仅适用于该尺寸的屏幕。 如果没有为备用资源提供最符合当前设备配置的限定符，则系统可能使用其中[最匹配](http://developer.android.com/intl/zh-cn/guide/topics/resources/providing-resources.html" \l "BestMatch" \t "_blank)的资源。  **注意：**如果所有资源均使用大于当前屏幕的尺寸限定符，则系统**不**会使用这些资源，并且应用在运行时将会崩溃（例如，如果所有布局资源均用 xlarge 限定符标记，但设备是标准尺寸的屏幕）。  此项为 API 级别 4 中新增配置。  如需了解详细信息，请参阅[支持多个屏幕](http://developer.android.com/guide/practices/screens_support.html)。  另请参阅 [screenLayout](http://developer.android.com/reference/android/content/res/Configuration.html" \l "screenLayout" \t "_blank) 配置字段，该字段表示屏幕是小尺寸、标准尺寸还是大尺寸。 |
| 屏幕纵横比 | long notlong | * long：宽屏，如 WQVGA、WVGA、FWVGA * notlong：非宽屏，如 QVGA、HVGA 和 VGA   此项为 API 级别 4 中新增配置。  它完全基于屏幕的纵横比（宽屏较宽），而与屏幕方向无关。  另请参阅 [screenLayout](http://developer.android.com/reference/android/content/res/Configuration.html" \l "screenLayout" \t "_blank) 配置字段，该字段指示屏幕是否为宽屏。 |
| 屏幕方向 | port land | * port：设备处于纵向（垂直） * land：设备处于横向（水平）   如果用户旋转屏幕，它有可能在应用生命周期中发生改变。 如需了解这会在运行期间给应用带来哪些影响，请参阅[处理运行时变更](http://developer.android.com/intl/zh-cn/guide/topics/resources/runtime-changes.html" \t "_blank)。  另请参阅 [orientation](http://developer.android.com/reference/android/content/res/Configuration.html#orientation) 配置字段，该字段指示当前的设备方向。 |
| UI 模式 | car desk television appliancewatch | * car：设备正在车载手机座上显示 * desk：设备正在桌面手机座上显示 * television：设备正在电视上显示，为用户提供“十英尺”体验，其 UI 位于远离用户的大屏幕上，主要面向方向键或其他非指针式交互 * appliance：设备用作不带显示屏的装置 * watch：设备配有显示屏，戴在手腕上   此项为 API 级别 8 中新增配置，API 13 中新增电视配置，API 20 中新增手表配置。  如需了解应用在设备插入手机座或从中移除时的响应方式，请阅读[确定并监控插接状态和类型](http://developer.android.com/training/monitoring-device-state/docking-monitoring.html)。  如果用户将设备放入手机座中，它有可能在应用生命周期中发生改变。 可以使用 [UiModeManager](http://developer.android.com/reference/android/app/UiModeManager.html" \t "_blank) 启用或禁用其中某些模式。如需了解这会在运行期间给应用带来哪些影响，请参阅[处理运行时变更](http://developer.android.com/intl/zh-cn/guide/topics/resources/runtime-changes.html" \t "_blank)。 |
| 夜间模式 | night notnight | * night：夜间 * notnight：白天   此项为 API 级别 8 中新增配置。  如果夜间模式停留在自动模式（默认），它有可能在应用生命周期中发生改变。在这种情况下，该模式会根据当天的时间进行调整。 可以使用[UiModeManager](http://developer.android.com/reference/android/app/UiModeManager.html) 启用或禁用此模式。如需了解这会在运行期间给应用带来哪些影响，请参阅[处理运行时变更](http://developer.android.com/intl/zh-cn/guide/topics/resources/runtime-changes.html" \t "_blank)。 |
| 屏幕像素密度 (dpi) | ldpi mdpi hdpi xhdpi xxhdpi xxxhdpi nodpi tvdpi | * ldpi：低密度屏幕；约为 120dpi。 * mdpi：中等密度（传统 HVGA）屏幕；约为 160dpi。 * hdpi：高密度屏幕；约为 240dpi。 * xhdpi：超高密度屏幕；约为 320dpi。API 级别 8 中新增配置 * xxhdpi：超超高密度屏幕；约为 480dpi。API 级别 16 中新增配置 * xxxhdpi：超超超高密度屏幕使用（仅限启动器图标，请参阅“支持多个屏幕”中的[注释](http://developer.android.com/guide/practices/screens_support.html" \l "xxxhdpi-note" \t "_blank)）；约为 640dpi。 API 级别 18 中新增配置 * nodpi：它可用于您不希望缩放以匹配设备密度的位图资源。 * tvdpi：密度介于 mdpi 和 hdpi 之间的屏幕；约为 213dpi。它并不是“主要”密度组， 主要用于电视，而大多数应用都不需要它。对于大多数应用而言，提供 mdpi 和 hdpi 资源便已足够，系统将根据需要对其进行缩放。API 级别 13 中引入了此限定符。   六个主要密度之间的缩放比为 3:4:6:8:12:16（忽略 tvdpi 密度）。因此，9x9 (ldpi) 位图相当于 12x12 (mdpi)、18x18 (hdpi)、24x24 (xhdpi) 位图，依此类推。  如果您认为图像资源在电视或其他某些设备上呈现的效果不够好，而想尝试使用 tvdpi 资源，则缩放比例为 1.33\*mdpi。例如，mdpi 屏幕的 100px x 100px 图像应该相当于 tvdpi 的133px x 133px。  **注：**使用密度限定符并不表示资源仅适用于该密度的屏幕。 如果没有为备用资源提供最符合当前设备配置的限定符，则系统可能使用其中[最匹配](http://developer.android.com/intl/zh-cn/guide/topics/resources/providing-resources.html" \l "BestMatch" \t "_blank)的资源。  如需了解有关如何处理不同屏幕密度以及 Android 如何缩放位图以适应当前密度的详细信息，请参阅[支持多个屏幕](http://developer.android.com/guide/practices/screens_support.html" \t "_blank)。 |
| 触摸屏类型 | notouch finger | * notouch：设备没有触摸屏。 * finger：设备有一个专供用户通过手指直接与其交互的触摸屏。   另请参阅 [touchscreen](http://developer.android.com/reference/android/content/res/Configuration.html#touchscreen) 配置字段，该字段指示设备上的触摸屏类型。 |
| 键盘可用性 | keysexposed keyshidden keyssoft | * keysexposed：设备具有可用的键盘。如果设备启用了软键盘（不无可能），那么即使硬键盘没有展示给用户，哪怕设备没有硬键盘，也可以使用此限定符。 如果没有提供或已经禁用软键盘，则只有在显示硬键盘时才会使用此限定符。 * keyshidden：设备具有可用的硬键盘，但它处于隐藏状态，且设备没有启用软键盘。 * keyssoft：设备已经启用软键盘（无论是否可见）。   如果提供了 keysexposed 资源，但未提供 keyssoft 资源，那么只要系统已经启用软键盘，就会使用 keysexposed 资源，而不考虑键盘是否可见。  如果用户打开硬键盘，它有可能在应用生命周期中发生改变。 如需了解这会在运行期间给应用带来哪些影响，请参阅[处理运行时变更](http://developer.android.com/intl/zh-cn/guide/topics/resources/runtime-changes.html)。  另请参阅配置字段 [hardKeyboardHidden](http://developer.android.com/reference/android/content/res/Configuration.html" \l "hardKeyboardHidden" \t "_blank) 和 [keyboardHidden](http://developer.android.com/reference/android/content/res/Configuration.html" \l "keyboardHidden" \t "_blank)，这两个字段分别指示硬键盘的可见性和任何一种键盘（包括软键盘）的可见性。 |
| 主要文本输入法 | nokeys qwerty 12key | * nokeys：设备没有用于文本输入的硬按键。 * qwerty：设备具有标准硬键盘（无论是否对用户可见）。 * 12key：设备具有 12 键硬键盘（无论是否对用户可见）。   另请参阅 [keyboard](http://developer.android.com/reference/android/content/res/Configuration.html#keyboard) 配置字段，该字段指示可用的主要文本输入法。 |
| 导航键可用性 | navexposed navhidden | * navexposed：导航键可供用户使用。 * navhidden：导航键不可用（例如，位于密封盖子后面）。   如果用户显示导航键，它有可能在应用生命周期中发生改变。 如需了解这会在运行期间给应用带来哪些影响，请参阅[处理运行时变更](http://developer.android.com/intl/zh-cn/guide/topics/resources/runtime-changes.html" \t "_blank)。  另请参阅 [navigationHidden](http://developer.android.com/reference/android/content/res/Configuration.html" \l "navigationHidden" \t "_blank) 配置字段，该字段指示导航键是否处于隐藏状态。 |
| 主要非触摸导航方法 | nonav dpad trackball wheel | * nonav：除了使用触摸屏以外，设备没有其他导航设施。 * dpad：设备具有用于导航的方向键。 * trackball：设备具有用于导航的轨迹球。 * wheel：设备具有用于导航的方向盘（不常见）。   另请参阅 [navigation](http://developer.android.com/reference/android/content/res/Configuration.html#navigation) 配置字段，该字段指示可用的导航方法类型。 |
| 平台版本（API 级别） | 示例： v3 v4 v7 等等 | 设备支持的 API 级别。例如，v1 对应于 API 级别 1（带有 Android 1.0 或更高版本系统的设备），v4 对应于 API 级别 4（带有 Android 1.6 或更高版本系统的设备）。如需了解有关这些值的详细信息，请参阅 [Android API 级别](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)文档。 |

### Android 如何找到最匹配资源

drawable/

drawable-en/

drawable-fr-rCA/

drawable-en-port/

drawable-en-notouch-12key/

drawable-port-ldpi/

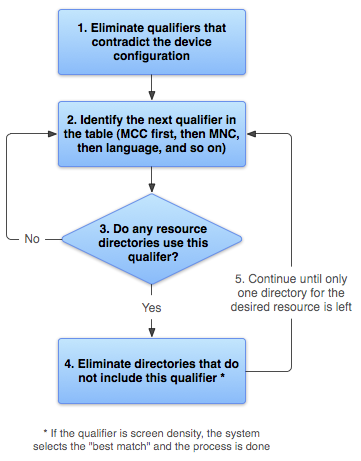
drawable-port-notouch-12key/

同时，假设设备配置如下：

区域设置 = en-GB   
屏幕方向 = port   
屏幕像素密度 = hdpi   
触摸屏类型 = notouch   
主要文本输入法 = 12key

通过将设备配置与可用的备用资源进行比较，Android 从 drawable-en-port 中选择 Drawable。

系统使用以下逻辑决定要使用的资源：



**图 2.**Android 如何找到最匹配资源的流程图。

1. 淘汰与设备配置冲突的资源文件。

drawable-fr-rCA/ 目录与 en-GB 区域设置冲突，因而被淘汰。

drawable/

drawable-en/

drawable-en-port/

drawable-en-notouch-12key/

drawable-port-ldpi/

drawable-port-notouch-12key/

**例外：**屏幕像素密度是唯一一个未因冲突而被淘汰的限定符。 尽管设备的屏幕密度为 hdpi，但是 drawable-port-ldpi/ 未被淘汰，因为此时每个屏幕密度均视为匹配。如需了解详细信息，请参阅[支持多个屏幕](http://developer.android.com/guide/practices/screens_support.html" \t "_blank)文档。

1. 选择列表（[表 2](http://developer.android.com/intl/zh-cn/guide/topics/resources/providing-resources.html" \l "table2" \t "_blank)）中（下一个）优先级最高的限定符。（先从 MCC 开始，然后下移。）
2. 是否有资源目录包括此限定符？
   * 若无，请返回到第 2 步，看看下一个限定符。（在该示例中，除非达到语言限定符，否则答案始终为“否”。）
   * 若有，请继续执行第 4 步。
3. 淘汰不含此限定符的资源目录。在该示例中，系统会淘汰所有不含语言限定符的目录。
4. drawable-en/
5. drawable-en-port/
6. drawable-en-notouch-12key/

**例外：**如果涉及的限定符是屏幕像素密度，则 Android 会选择最接近设备屏幕密度的选项。通常，Android 倾向于缩小大型原始图像，而不是放大小型原始图像。请参阅[支持多个屏幕](http://developer.android.com/guide/practices/screens_support.html" \t "_blank)。

1. 返回并重复第 2 步、第 3 步和第 4 步，直到只剩下一个目录为止。在此示例中，屏幕方向是下一个判断是否匹配的限定符。因此，未指定屏幕方向的资源被淘汰：
2. drawable-en-port/

剩下的目录是 drawable-en-port。

# 其他

<supports-screens

android:anyDensity="true"

android:largeScreens="true"

android:normalScreens="true"

android:xlargeScreens="true" />

Manifest.xml中配置<supports-screens>可以禁止在不合适的设备上安装

<activity

android:name="dji.pilot2.main.activity.DJIMainFragmentActivity"

android:configChanges="orientation|screenSize|keyboardHidden|keyboard"

android:theme="@style/AppThemeMine" />

# zs600方案

5.5寸的屏幕是：400dpi，xxhdpi,像素为1080\*1920，layout-768x432和带屏遥控器的一样：2.5倍的demen

7.85是适配320分辨率（2048\*1536）；2倍

目前我们定义了

Land：横屏

Large：7.0寸以上

Layout-xlarge：

w1024dp

w1280dp

GO： DJIOriLayout.setOrientationByDevice(this);控制了

系统强制设为了横屏，代码找不到

之前应该在large里面适配过了吧，其实可以模仿的

编译选项中，为何控制了pad版本出来是另外的界面？

我们的平板是用的large哦，

原则：相对位置不变，在利用dimens修饰符适配，相对位置改变，用layout的修饰符适配，逻辑修改，用编译变量控制。

Size(large)>sw >land >768\*432> x xhdpi

5.5的layout最佳方案：layout-sw432dp-w432dp-normal-land(768\*432)

7.85的最佳方案：layout-sw768dp-w768dp-xlarge-land(1024\*768)

Dimen呢、drawalbe呢？在手机上正常么？

如何自定义限定符号

# 参考

1. [android-View工作原理（三）视图大小计算过程（measure过程）](http://xixinfei.iteye.com/blog/2114701)

<http://xixinfei.iteye.com/blog/2114701>

2. 安卓平板化趟过的坑

<http://www.jianshu.com/p/042c1f268c69>

3. Android中.9图片的含义及制作教程

<http://www.jianshu.com/p/3fd048644e3f>

4.枚举法像素值

<http://blog.csdn.net/lmj623565791/article/details/45460089>

5. Manifest.xml中配置<supports-screens>的作用及参数解析

http://blog.csdn.net/cangchen/article/details/44237237

6. developer

<https://developer.android.com/guide/practices/screens_support.html>

1. Android源码系列之深入理解ImageView的ScaleType属性

<http://www.jb51.net/article/85983.htm>

9. 请参考：4.1. 资源限定符优先级

http://developer.[**Android**](http://lib.csdn.net/base/android).com/intl/zh-cn/guide/topics/resources/providing-resources.html#BestMatch