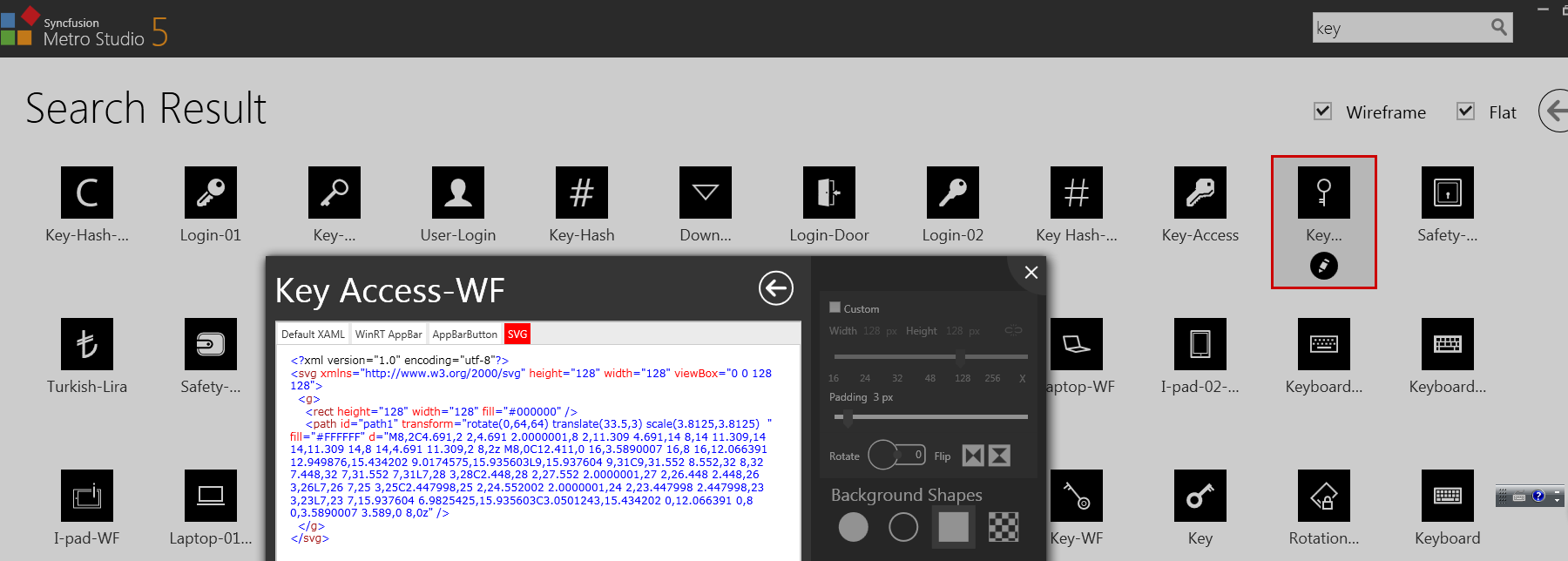
# Android 矢量图

## 工具介绍

### 图片资源编译工具Metro Studio

模板相当的多！支持SVG/PNG!



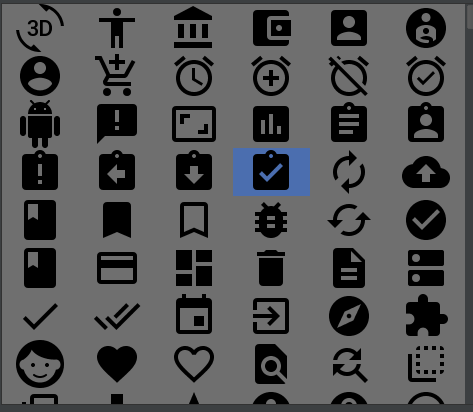
inker在线绘制矢量图的网站[http://app.inker.co](https://link.jianshu.com/?t=http://app.inker.co/)

svg收费的！

矢量编辑软件。比如：安装illustrator／sketch／Coreldraw/InDesign等。

### android studio

提供了丰富的图片资源，可以右键module,new->vector asset选择，如下



## vector 属性

android:name 定义该drawable的名字

android:width 定义该 drawable 的内部(intrinsic)宽度,支持所有 Android 系统支持的尺寸，通常使用 dp

android:height 定义该 drawable 的内部(intrinsic)高度,支持所有 Android 系统支持的尺寸，通常使用 dp

android:viewportWidth 定义矢量图视图的宽度，视图就是矢量图 path 路径数据所绘制的虚拟画布

android:viewportHeight 定义矢量图视图的高度，视图就是矢量图 path 路径数据所绘制的虚拟画布

android:tint 定义该 drawable 的 tint 颜色。默认是没有 tint 颜色的

android:tintMode 定义 tint 颜色的 Porter-Duff blending 模式，默认值为 src\_in

android:autoMirrored 设置当系统为 RTL (right-to-left) 布局的时候，是否自动镜像该图片。比如 阿拉伯语。

android:alpha 该图片的透明度属性

<group>里面可以定义多了<path>，这样可以方便管理多个<path>

## vector 标签 pathData 语法

Sdf

**[javascript]** [view plain](https://blog.csdn.net/qq1028850792/article/details/80287021) [copy](https://blog.csdn.net/qq1028850792/article/details/80287021)

1. <!--上面的手掌对应的代码实现-->
2. <vector xmlns:android="http://schemas.android.com/apk/res/android"
3. android:width="24dp"
4. android:height="24dp"
5. android:viewportWidth="24"
6. android:viewportHeight="24">
7. <path
8. android:fillColor="#000000"
9. android:pathData="
10. M22,23 q0,4 -4,4 h-7 q-2,0 -3,-1 T1,16 q-0.6,-0.8 0,-2 t5,3
11. q1,1 2,0 T8,4 q0,-1 0.9,-1.1 t1.1,1 1.5,9 q0.25,0.5 0.5,0.5
12. t0.5,-0.5 0,-11 q0.2,-1 1.1,-1.1 t1.1,1.1 1,11 q0.25,0.5 0.5,0.5
13. t0.5,-0.5 0.5,-9 q0.2,-1 1,-1 t1,1 0.5,9 q0.25,0.5 0.5,0.5
14. t0.5,-0.5 1.2,-6.5 q0.3,-1 1,-1 t0.8,1 -0.8,6 T22,23"/>
15. </vector>

### 基本语法

pathData 的指令基本都是由字母跟若干数字组成，数字之间可以用空格或者逗号隔开 (其实逗号会被忽略掉，加上逗号只是一些习惯的问题)。一般来说指令字母分为大小写两种,**大写的字母是基于原点的坐标系(偏移量)，即绝对位置；小写字母是基于当前点坐标系(偏移量)，即相对位置。**

**每个命令都有大小写形式，大写代表后面的参数是绝对坐标，小写表示相对坐标，相对于上一个点的位置。参数之间用空格或逗号隔开。**

### 移动M

move to 移动绘制点，作用相当于把画笔落在哪一点。

**M x,y (m dx, dy)** 移动虚拟画笔到对应的点，但是并不绘制。一开始的时候默认是在(0,0)。

### 直线L

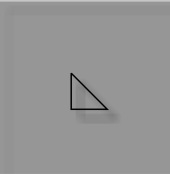
line to 直线，就是一条直线，**注意**，只是直线，直线是没有宽度的，所以你什么也看不到。

android:strokeColor="#333330" android:strokeWidth="10" 设置颜色和线宽

* **L x,y (l dx, dy)** 从当前点划一条直线到对应的点。
* **H x (h dx)** 从当前点绘制水平线，相当于l x,0
* **V y (v dy)** 从当前点绘制垂直线，相当于l 0,y

**[javascript]** [view plain](https://blog.csdn.net/qq1028850792/article/details/80287021) [copy](https://blog.csdn.net/qq1028850792/article/details/80287021)

1. <vector xmlns:android="http://schemas.android.com/apk/res/android"
2. android:width="24dp"
3. android:height="24dp"
4. android:viewportWidth="24"
5. android:viewportHeight="24">
6. <path
7. android:fillColor="#0000"
8. android:strokeColor="#000"
9. android:strokeWidth="0.2"
10. android:pathData=" M10,10 L10,15 L15,15 L10,10"/>
11. </vector>



将上述代码 android:pathData=" M10,10 L10,15 L15,15 L10,10" 替换成以下代码效果相同

android:pathData="M10,10 l 0,5 l 5,0 l-5,-5"

android:pathData="M10,10 V 15 H 15 L10,10"

android:pathData="M10,10 v 5 h 5 l-5,-5"

### 闭合Z

* **Z(或z)** 从结束点绘制一条直线到开始点，闭合路径

上面的图形型也可以由以下代码绘制  
android:pathData="M10,10 v 5 h 5 z"

### 弧线

ellipse 圆弧

* **A rx,ry x-axis-rotation large-arc-flag,sweepflag x,y**
* **a rx,ry x-axis-rotation large-arc-flag,sweepflag dx,dy**

rx ry 椭圆半径  
x-axis-rotation x轴旋转角度  
large-arc-flag 为0时表示取小弧度，1时取大弧度（要长的还是短的）  
sweep-flag 0取逆时针方向，1取顺时针方向  
x,y (dx,dy) 终点的位置

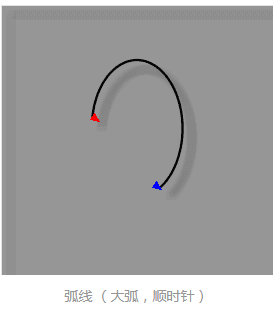
### 画圆

A（圆的x轴半径，y轴半径，x轴旋转角度，0/1-小/大弧度，0/1-逆/顺时针，圆的终点位置x点，圆的终点位置y点）

**Android 绘图和shape圆形**

**https://blog.csdn.net/linghu\_java/article/details/46403817**

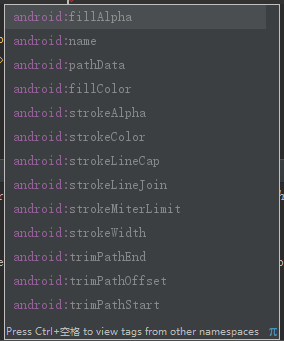
这个弧线的指令比起直线就相对复杂得多了，7个参数容易搞混了。来看个例子  
android:pathData="M8,10 a4,6 0 1,1 6 6"



[Android vector 标签 pathData 详解](https://www.jianshu.com/p/a3cb1e23c2c4)

https://www.cnblogs.com/yuhanghzsd/p/5466846.html

## path其他android属性



android:name 定义该 path 的名字，这样在其他地方可以通过名字来引用这个路径

android:pathData 和 SVG 中 d 元素一样的路径信息。

android:fillColor 定义填充路径的颜色，如果没有定义则不填充路径

android:strokeColor 定义如何绘制路径边框，如果没有定义则不显示边框

android:strokeWidth 定义路径边框的粗细尺寸

android:strokeAlpha 定义路径边框的透明度

android:fillAlpha 定义填充路径颜色的透明度

android:trimPathStart 从路径起始位置截断路径的比率，取值范围从 0 到1

android:trimPathEnd 从路径结束位置截断路径的比率，取值范围从 0 到1

android:trimPathOffset 设置路径截取的范围 Shift trim region (allows showed region to include the start and end), in the range from 0 to 1.

android:strokeLineCap 设置路径线帽的形状，取值为 butt, round, square.

android:strokeLineJoin 设置路径交界处的连接方式，取值为 miter,round,bevel.

android:strokeMiterLimit 设置斜角的上限，Sets the Miter limit for a stroked path.

## <group>

Fg

android:name 定义 group 的名字

android:rotation 定义该 group 的路径旋转多少度，这样图片就被旋转了，注意写数字的时候别晕了。

android:pivotX 定义缩放和旋转该 group 时候的 X 参考点。该值相对于 vector 的 viewport 值来指定的。

android:pivotY 定义缩放和旋转该 group 时候的 Y 参考点。该值相对于 vector 的 viewport 值来指定的。

android:scaleX 定义 X 轴的缩放倍数

android:scaleY 定义 Y 轴的缩放倍数

android:translateX 定义移动 X 轴的位移。相对于 vector 的 viewport 值来指定的。

android:translateY 定义移动 Y 轴的位移。相对于 vector 的 viewport 值来指定的。

## 性能问题

矢量图会有性能问题么，相比于png

<https://blog.csdn.net/a1961613299/article/details/50512729>

理论上讲，SVG的效率可能会不如PNG好，这是因为它需要运行时的计算和对应平台的渲染绘制。而且对于PNG来说的另一优势是在开启硬件加速的设备上，绘制Bitmap一个非常快速的过程。可以想象，让SVG不比PNG慢将是一件很有挑战的事情。

[微信的矢量化解决方案——WeChatSVGCode](https://blog.csdn.net/a1961613299/article/details/50512729)

# Anroid 6 动态权限申请

对于6.0以下的权限及在安装的时候，根据权限声明产生一个权限列表，用户只有在同意之后才能完成app的安装，造成了我们想要使用某个app，就要默默忍受其一些不必要的权限（比如是个app都要访问通讯录、短信等）。而在6.0以后，我们可以直接安装，当app需要我们授予不恰当的权限的时候，我们可以予以拒绝（比如：单机的象棋对战，请求访问任何权限，我都是不同意的）。当然你也可以在设置界面对每个app的权限进行查看，以及对单个权限进行授权或者解除授权。

新的权限机制更好的保护了用户的隐私，Google将权限分为两类，一类是Normal Permissions，这类权限一般不涉及用户隐私，是不需要用户进行授权的，比如手机震动、访问网络等；另一类是Dangerous Permission，一般是涉及到用户隐私的，需要用户进行授权，比如读取sdcard、访问通讯录等。

## 权限分类

### Normal Permissions如下

ACCESS\_LOCATION\_EXTRA\_COMMANDS

ACCESS\_NETWORK\_STATE

ACCESS\_NOTIFICATION\_POLICY

ACCESS\_WIFI\_STATE

BLUETOOTH

BLUETOOTH\_ADMIN

BROADCAST\_STICKY

CHANGE\_NETWORK\_STATE

CHANGE\_WIFI\_MULTICAST\_STATE

CHANGE\_WIFI\_STATE

DISABLE\_KEYGUARD

EXPAND\_STATUS\_BAR

GET\_PACKAGE\_SIZE

INSTALL\_SHORTCUT

INTERNET

KILL\_BACKGROUND\_PROCESSES

MODIFY\_AUDIO\_SETTINGS

NFC

READ\_SYNC\_SETTINGS

READ\_SYNC\_STATS

RECEIVE\_BOOT\_COMPLETED

REORDER\_TASKS

REQUEST\_INSTALL\_PACKAGES

SET\_ALARM

SET\_TIME\_ZONE

SET\_WALLPAPER

SET\_WALLPAPER\_HINTS

TRANSMIT\_IR

UNINSTALL\_SHORTCUT

USE\_FINGERPRINT

VIBRATE

WAKE\_LOCK

WRITE\_SYNC\_SETTINGS

### Dangerous Permissions:

group:android.permission-group.CONTACTS

permission:android.permission.WRITE\_CONTACTS

permission:android.permission.GET\_ACCOUNTS

permission:android.permission.READ\_CONTACTS

group:android.permission-group.PHONE

permission:android.permission.READ\_CALL\_LOG

permission:android.permission.READ\_PHONE\_STATE

permission:android.permission.CALL\_PHONE

permission:android.permission.WRITE\_CALL\_LOG

permission:android.permission.USE\_SIP

permission:android.permission.PROCESS\_OUTGOING\_CALLS

permission:com.android.voicemail.permission.ADD\_VOICEMAIL

group:android.permission-group.CALENDAR

permission:android.permission.READ\_CALENDAR

permission:android.permission.WRITE\_CALENDAR

group:android.permission-group.CAMERA

permission:android.permission.CAMERA

group:android.permission-group.SENSORS

permission:android.permission.BODY\_SENSORS

group:android.permission-group.LOCATION

permission:android.permission.ACCESS\_FINE\_LOCATION

permission:android.permission.ACCESS\_COARSE\_LOCATION

group:android.permission-group.STORAGE

permission:android.permission.READ\_EXTERNAL\_STORAGE

permission:android.permission.WRITE\_EXTERNAL\_STORAGE

group:android.permission-group.MICROPHONE

permission:android.permission.RECORD\_AUDIO

group:android.permission-group.SMS

permission:android.permission.READ\_SMS

permission:android.permission.RECEIVE\_WAP\_PUSH

permission:android.permission.RECEIVE\_MMS

permission:android.permission.RECEIVE\_SMS

permission:android.permission.SEND\_SMS

permission:android.permission.READ\_CELL\_BROADCASTS

可以通过adb shell pm list permissions -d -g进行查看。

看到上面的dangerous permissions，会发现一个问题，好像危险权限都是一组一组的，恩，没错，的确是这样的，

那么有个问题：分组对我们的权限机制有什么影响吗？

的确是有影响的，如果app运行在Android 6.x的机器上，对于授权机制是这样的。如果你申请某个危险的权限，假设你的app早已被用户授权了**同一组**的某个危险权限，那么系统会立即授权，而不需要用户去点击授权。比如你的app对READ\_CONTACTS已经授权了，当你的app申请WRITE\_CONTACTS时，系统会直接授权通过。此外，对于申请时弹出的dialog上面的文本说明也是对整个权限组的说明，而不是单个权限（ps:这个dialog是不能进行定制的）。

不过需要注意的是，不要对权限组过多的依赖，尽可能对每个危险权限都进行正常流程的申请，因为在后期的版本中这个权限组可能会产生变化。

## 检查权限hasPermissions

**[java]** [view plain](https://blog.csdn.net/qq1028850792/article/details/80287021) [copy](https://blog.csdn.net/qq1028850792/article/details/80287021)

1. **private** **boolean** hasPermissions() {
2. PackageManager pm = getPackageManager();
3. String packageName = getPackageName();
4. **int** granted = (mAudioToggle.isChecked() ? pm.checkPermission(RECORD\_AUDIO, packageName) : PackageManager
5. .PERMISSION\_GRANTED)
6. | pm.checkPermission(WRITE\_EXTERNAL\_STORAGE, packageName);
7. **return** granted == PackageManager.PERMISSION\_GRANTED;
8. }

## 请求权限requestPermissions

g

**[java]** [view plain](https://blog.csdn.net/qq1028850792/article/details/80287021) [copy](https://blog.csdn.net/qq1028850792/article/details/80287021)

1. @TargetApi(M)
2. **private** **void** requestPermissions() {
3. String[] permissions = mAudioToggle.isChecked()
4. ? **new** String[]{WRITE\_EXTERNAL\_STORAGE, RECORD\_AUDIO}
5. : **new** String[]{WRITE\_EXTERNAL\_STORAGE};
6. **boolean** showRationale = **false**;
7. **for** (String perm : permissions) {
8. showRationale |= shouldShowRequestPermissionRationale(perm);
9. }
10. **if** (!showRationale) {
11. requestPermissions(permissions, REQUEST\_PERMISSIONS);
12. **return**;
13. }

## 回调结果onRequestPermissionsResult

Sdf

**[java]** [view plain](https://blog.csdn.net/qq1028850792/article/details/80287021) [copy](https://blog.csdn.net/qq1028850792/article/details/80287021)

1. @Override
2. **public** **void** onRequestPermissionsResult(**int** requestCode, String[] permissions, **int**[] grantResults) {
3. **if** (requestCode == REQUEST\_PERMISSIONS) {
4. **int** granted = PackageManager.PERMISSION\_GRANTED;
5. **for** (**int** r : grantResults) {
6. granted |= r;
7. }
8. **if** (granted == PackageManager.PERMISSION\_GRANTED) {
9. startCaptureIntent();
10. } **else** {
11. toast("No Permission!");
12. }
13. }
14. }

## REF

[在运行时请求权限](https://developer.android.com/training/permissions/requesting?hl=zh-cn)

# Android文件说明

## **jar与sources.jar**

编译后的class文件。这种文件是看不到源码的

Attach Source.关联源码。对应的source.jar里面

## **jar和aar区别：**

\*.jar：只包含了class文件与清单文件，不包含资源文件，如图片等所有res中的文件。

\*.aar：包含所有资源，class以及res资源文件全部包含

如果你只是一个简单的类库那么使用生成的\*.jar文件即可；如果你的是一个UI库，包含一些自己写的控件布局文件以及字体等资源文件那么就只能使用\*.aar文件。

注意：在更改Android的系统应用时，要引用一个aar文件。之前引用的主要是jar包，而aar文件包含Android的资源文件，如：布局、样式、图片等，如果按照源码中jar的引用方式会遇到编译不过的问题，提示找不到相关的资源文件。

LOCAL\_AAPT\_FLAGS := \ --auto-add-overlay \ --extra-packages <aar package name>

[引用一个aar文件](http://www.voidcn.com/article/p-cmriynxn-bqy.html)

# Native

Midware也是这么用的

注册函数

静态注册

模仿systemUI就好了吧 建立一下java的TcpLinkHandler

修改link源码，完成TcpLinkHandler的注册加载：主要在main.cpp调用？

Java代码，声明native方法

总是加载失败

依赖

[引用jar及so文件](https://blog.csdn.net/zuiwuyuan/article/details/48201655)

# QA

## not accessible for the namespace "classloader-namespace"

https://www.jianshu.com/p/a4af2bdcc3c0

<https://www.jianshu.com/p/4be3d1dafbec>

crash info--------------------------------

java.lang.UnsatisfiedLinkError: dlopen failed: library "/system/lib64/libAppUpgradeLink.so" needed or dlopened by "/system/lib64/libnativeloader.so" is not accessible for the namespace "classloader-namespace"

at java.lang.Runtime.loadLibrary0(Runtime.java:989)

at java.lang.System.loadLibrary(System.java:1562)

at com.dpad.core.data.DpadLinkUtil.<clinit>(DpadLinkUtil.java:6)