由于开源三方定制系统较多，请大家详细描述场景、机型及解决方案，方便其他朋友参考

[问答]-Android开发中有哪些兼容性问题？都是怎么解决的？ [问答] 你在工作中遇到的最复杂的问题或者bug是什么？你是怎么搞定的?

#### **华为P6和P7**

场景：使用MIPush，在华为部分手机上无法推送成功。 机型：[华为P6，华为P7] 解决方案：P6和P7是华为的高端机型，不允许推送，防止骚扰用户，无解。

#### **魅族3和魅族4**

场景：魅族手机ListView的Item中的EditText无法编辑，点击EditText弹出软键盘后，软键盘会立即自动隐藏 机型：[魅族3，魅族4] 解决方案： 方法一：将ListView换成RecyclerView

方法二：<https://github.com/Aspsine/EditTextInListView>

#### **HTC M8**

场景：HTC M8 从一个Activity 使用QQSDK 登陆, 登陆成功后, 返回Activity结果Activity 被销毁了 机型：HTC M8 等某些带有 虚拟 Menu 键盘的手机 解决方案：后来调查发现是这个Activity是全屏,屏蔽了Menu键盘的黑条. 但是跳转到QQ却把那个Menu的黑条显示了出来, 这导致发生了 screenSize 的变化 从而导致我的Activity销毁了. 知道了这个原因, 在manifest中的 configChanges 添加screenSize 解决了这个问题.

#### **所有android4.4机型**

场景：Android4.4系统使用了SystemBarTintManager库修改透明状态栏后，会导致根布局从屏幕顶端开始布局，而不是从ActionBar开始布局 机型：所有android4.4机型 解决方案： 方法一：针对4.4创建一套额外的布局，即layout-v19文件夹，并且在根布局外层再套一层LinearLayout，并在LinearLayout中添加一个属性android:fitsSystemWindows="true"

方法二：是为4.4及以上添加了paddingTop去适配，添加layout觉得不好适配。

方法三：在Build.VERSION.SDK\_INT <= 18的版本中，通过colorDrawable.setAlpha(alpha);设置actionbar背景色透明度的时候，colorDrawable需要设置callback。

final Drawable.Callback mDrawableCallback = new Drawable.Callback() {  
 @Override  
 public void invalidateDrawable(Drawable who) {  
 getActionBar().setBackgroundDrawable(who);  
 }  
  
 @Override  
 public void scheduleDrawable(Drawable who, Runnable what, long when) {  
 }  
  
 @Override  
 public void unscheduleDrawable(Drawable who, Runnable what) {  
 }  
 };  
 colorDrawable.setCallback(mDrawableCallback);

#### **魅族MX3**

场景：Camera拍摄，用setPreviewFormat设置成YV12，预览会变成绿屏，实际用getPreviewFormat显示是支持YV12的 机型：魅族MX3 解决方案：没办法只能设置成NV21了

#### **其代表机型为：三星I8258、华为H30-T00、红米等。**

Camera常见问题： 1）Intent调用手机内相机程序



img

如果我们设置了照片的存储路径，那么很可能会遇到一下三种问题：

问题一：onActivityResult方法中的data返回为空（数据表明，93%的机型的data将会是Null，所以如果我们指定了路径，就不要使用data来获取照片，起码在使用前要做空判断）。 问题二：照片无法存储。 如果自定义存储路径是/mnt/sdcard/lowry/，而手机SD卡下在拍照前没有名为lowry的文件夹，那么部分手机拍照后图片不会保存，导致我们无法获得照片，大多数手机的相机遇到文件夹不存在的情况都会自己创建出不存在的文件夹，而个别手机却不会创建，

解决的方法就是在指定存储路径前先判断路径中的文件夹是否都存在，不存在先创建再调用相机。

问题三：照片可以存储，但是名字不对。 file:///mnt/sdcard/123 1.jpg，由于URI的fromFile方法会将路径中的空格用“%20”取代。

其实对于大多数的手机这都不算事，手机在解析存储路径的时候都会将“%20”替换为空格，这样实际上最终的照片名字还是我们当初指定的名字：123 1.jpg，遗憾的是个别手机（如酷派7260）系统自带的相机没有将“%20”读成空格，拍照后的照片的名字是123%201.jpg，我们用路径“file:///mnt/sdcard/123 1.jpg”能找到照片才怪！



img



img

总结：

（1）使用onActivityResult中的intent(data)前要做空判断。 （2）指定拍照路径时，先检查路径中的文件夹是否都存在，不存在时先创建文件夹再调用 相机拍照。 （3）指定拍照存储路径时，照片的命名中不要包含空格等特殊符号。

#### **所有手机**

PopupWindow中嵌套EditText，会出现EditText长按无法触发“粘贴”选项，可以改成Dialog嵌套EditText，包括DialogFragment。

#### **三星Note系列,S系列**

会调用Activity的onPause和onStop方法.其他手机会保持在onResume状态

#### **酷派8720L**

场景：在获取系统相机拍照然后保存在本地有时候会保存不上，获取不到地址。 问题原因：通过调试发现当拍完照返回的时候自己设的成员变量值会被回收，估计就是内存不足的原因。重启机器后就好了。 解决方案：无方案。

#### **所有手机**

场景：输入法中的emoji适配，Android4.1之前的系统不支持emoji显示

解决方案：所以对于Android4.1之前的系统，我采用了bitmap来显示emoji。

#### **三星手机**

问题： (1) 摄像头拍照后图片数据不一定能返回 ; onActivityResult的data为空 (2) 三星的camera强制切换到横屏 导致Activity重启生命周期 (但是部分机型 配置 android:configChanges 也不能阻止横竖屏切换); (3) APP Activity A调用系统拍照 --> 拍照 --> 在拍好照片的界面做几次横竖屏转换 --> 返回APP界面Activity A ，A 被销毁。

解决方案：如果 activity 的销毁如果无法避免 那么在activity销毁之前调用 onSaveInstanceState 保存图片的路径 当activity重新创建的时候 会将 onSaveInstanceState 保存的文件传递给onCreate()当中 在onCreate当中 检查照片的地址是否存在文件 以此来判定拍照是否成功 Demo 下载地址: [http://download.csdn.NET/detail/aaawqqq/7653475](http://download.csdn.net/detail/aaawqqq/7653475)

#### **OPPO 手机**

场景：OPPO 手机启动 Service 报 SecurityException 解决方案：try catch 该异常

#### **HTC Desire 820、Lenovo A320T**

场景：这个问题主要在部分机型的4.X系统上遇见，小图标大小没有按照24dp裁剪，而是采用了桌面图标一样的大小96dp



img

解决方案：按照标准来，小图标大小为24dp，大图标为桌面icon图标大小96dp

#### **魅族5.X手机，大图显示问题**

场景：Flyme系统对原生Android源码做了修改，采用BigPictureStyle方式显示大图通知栏的时候，消息与大图重合了，如下图。



img

解决方案

首先，通过BigPictureStyle来实现大图功能肯定是走不通的，因为事实就摆着行不通的嘛。京东的App肯定是通过RemoteViews来实现的。于是，开始走弯路，尝试通过RemoteViews来展示大图。但是谷歌规定，自定义布局展示的通知栏消息最大高度是64dp。那么，京东的App是怎么实现的？在尝试了各种方法以后，最后又是通过投机取巧的方式解决了问题

private void showBigPictureNotificationWithMZ(Context context) {  
 NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION\_SERVICE);  
 Notification.Builder builder = new Notification.Builder(context);  
 Notification notification = generateNotification(builder);  
 notification.bigContentView = mRemoteViews;  
 notificationManager.notify(notifyId, notification);  
}

需要先生成Notification的实例，然后手动给notification.bigContentView赋值，再notify，就可以了

#### **问题二：顶部状态栏(StatusBar)小图标显示异常**

场景：当通知来的时候，如果不在通知栏浏览，会在顶部状态栏出现一个向上翻滚动画的通知消息，这条通知消息左边是一个小图标。部分系统这个小图标显示异常，是一个纯灰色的正方形，如下图。



img

解决方案

首先产生灰色图标的原因就是5.0系统引入了材料设计，谷歌强制使用带有alpha通道的图标，并且RGB的alpha值必须是0(实测不为0也是可以的，但系统会忽略所有RGB值)。因此，使用JPG的图片是不行的，最好的代替方案就是一张背景透明的PNG图片。

#### **问题三：Android 7.X机型，通知栏小图标显示成灰色**

问题详情

这个问题跟第二个有点类似，在7.0系统及以上，有部分应用的小图标是灰色的，大图可以正常显示。碰巧的是，显示异常的小图标，颜色都是灰色的。



img

解决方案

与小图标显示异常解决方案类似，将小图标替换为透明背景的PNG图片。

#### **问题四：RemoteViews显示异常**

问题详情

由于系统提供的通知栏消息类型有时候不能满足要求，部分通知栏消息采用自定义RemoteViews来实现。采用RemoteViews，特别是手动生成Bitmap然后直接传给一个自定义Layout，再通过setContentView方式设置通知栏消息时，会存在各种各样的坑。

Android通知栏的背景色有几种情况，白色、暗色、暗色透明和黑色。如果生成的Bitmap带背景色，这个背景色就很难选择。如果选择黑色背景，那么在白色通知栏的机型上就很难看。因此不能完全在各个系统上面完美展示出来。如果不带背景色，那么字体颜色也面临同样的困惑。试想，如果在白色的背景上显示白色的文字，用户看到白茫茫一片，是什么感受？



img

另一方面，大部分厂商对原生的Android系统都会有各种各样的改造，通知栏的样式也不例外。如果按照原生的样式来设计，那么在大部分国内厂商的机子上显示都和正常的普通通知栏消息不一样。例如华为6.0系统的机子，原生系统的时间线在右上角，华为的在左边，这样会给用户带来错觉。



img

解决方案

详见RemoteViews适配一节。

#### **问题五：通知栏更新频率**

问题详情

每个应用基本都有自更新的逻辑，App开机的时候提示用户升级，点击升级按钮后在Notification出现一个下载带进度条的通知。应用一般是在开启一个工作线程在后台下载，然后在下载的过程中通过回调更新通知栏中的进度条。我们知道，下载进度的快慢是不可控的，如果每次下载中的回调都去更新通知栏，那么可能几百毫秒、几十毫秒、甚至几毫秒就更新一次通知栏，应用可能就会ANR，甚至崩溃。

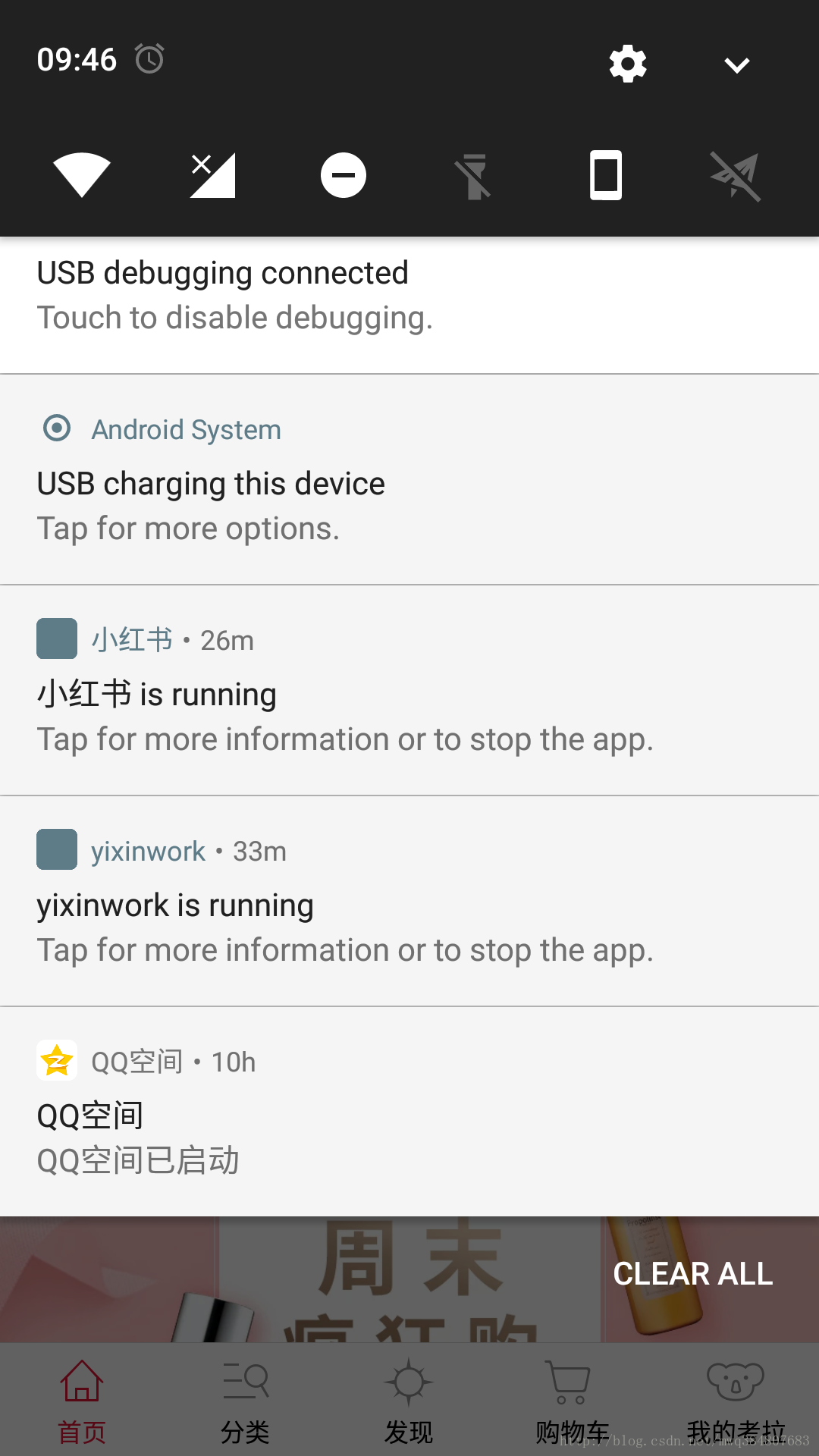
解决方案

控制通知栏更新频率，一般控制在0.5s或者1s就可以了。在某一个更新时间间隔内下载的进度回调直接丢弃，需要注意的是下载完成的回调，需要实时回调通知栏消息显示下载完成。

问题六：恶心的后台通知和“守护”通知 问题详情 但凡存在后台通知或者“守护”通知的应用，在7.0系统以后都会原形毕露.



img



img

解决方案:无

#### **小米推送SDK接入问题**

问题详情

为了提升推送到达，考拉接入了小米推送的SDK。小米推送分为通知栏消息和透传消息，通知栏消息属于系统级推送，在MIUI的机子上可以在进程被杀死的情况下也能收到应用推送。然而有个问题，小米认为应用在前台时，不会回调任何方法；小米认为应用在后台的时候，收到通知栏消息的同时，会回调onNotificationMessageArrived方法。这时候就要小心翼翼地处理这条消息了。因为如果你的应用前后台判断逻辑和小米的不一样，那么就有可能小米帮你发了一条通知栏消息，你自己又发了一遍，造成通知栏消息的重复发送(这个坑考拉踩过T\_T)。另一方面，在7.0系统的机子上，主标题和小图标的颜色是可以改变的，目前小米推送SDK没有开放这个接口供调用方定制。

解决方案

目前只能解决第一个问题——前后台判断的问题。应用是否在后台可以根据以下代码进行判断。在Android 5.0以上，可以通过ActivityManager.RunningAppProcessInfo判断，Android 5.0及以下版本通过ActivityManager.RunningTaskInfo判断。经测试，这个方案在Android 4.4以上结果是可以完全匹配的。

public static boolean isAppInBackgroundInternal(Context context) {  
 ActivityManager manager = (ActivityManager) context.getSystemService(Context.ACTIVITY\_SERVICE);  
 if (Build.VERSION.SDK\_INT > Build.VERSION\_CODES.LOLLIPOP) {  
 List<ActivityManager.RunningAppProcessInfo> runningProcesses = manager.getRunningAppProcesses();  
 if (!ListUtils.isEmpty(runningProcesses)) {  
 for (ActivityManager.RunningAppProcessInfo runningProcess : runningProcesses) {  
 if (runningProcess.importance == ActivityManager.RunningAppProcessInfo.IMPORTANCE\_FOREGROUND) {  
 return false;  
 }  
 }  
 }  
 } else {  
 List<ActivityManager.RunningTaskInfo> task = manager.getRunningTasks(1);  
 if (!ListUtils.isEmpty(task)) {  
 ComponentName info = task.get(0).topActivity;  
 if (null != info) {  
 return !isKaolaProcess(info.getPackageName());  
 }  
 }  
 }  
 return true;  
}

#### **Android通知栏适配**

RemoteViews适配 由于系统自带的通知栏消息样式不能完全满足产品们脑洞大开的需求，有时候我们需要自定义布局样式展示通知栏消息。Android系统可以将自定义布局通过setContent(7.X系统推荐使用setCustomContentView)设置到Notification.Builder中，来实现样式的更变。setContent方法需要传入一个RemoteViews对象，它是一个普通的数据类型，不是View，作用是供其他进程展示视图。RemoteViews只支持4种基本的布局:

FrameLayout LinearLayout RelativeLayout GridLayout 这些布局下面只支持几种视图控件:

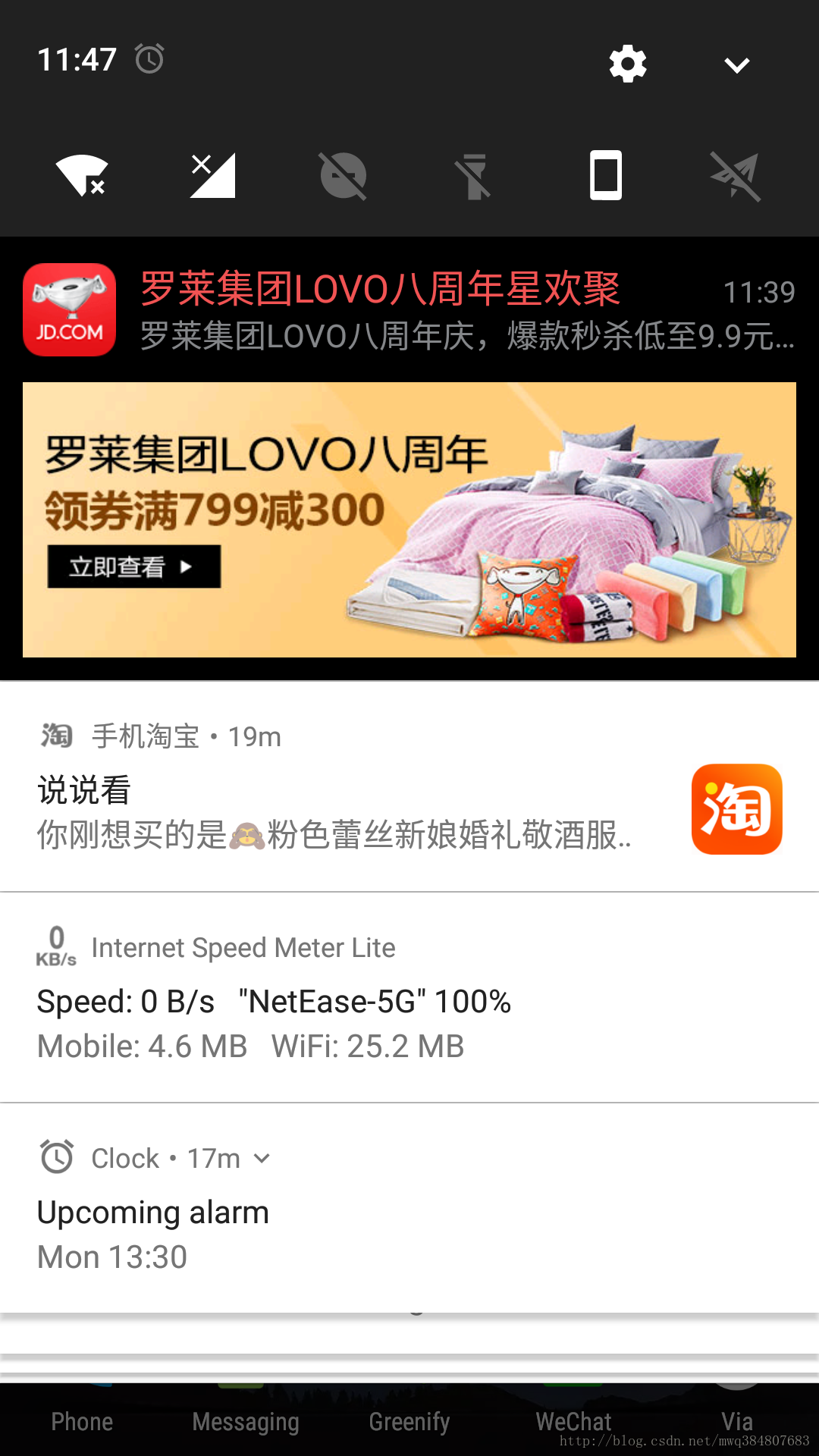
AnalogClock Button Chronometer ImageButton ImageView ProgressBar TextView ViewFlipper ListView GridView StackView AdapterViewFlipper 只能通过上述组合生成一个RemoteViews。

自定义布局与视图

除了上面提到的布局与控件，有没有办法自定义布局与视图呢？我们知道，任何一个View，都可以生成一个Bitmap对象，支持的视图控件里有ImageView，可以通过ImageView.setBitmapResource()将自定义视图设置到一个ImageView中，然后再随便放到一个布局上，就可以实现通知栏消息的任意布局。理想是美好的，但现实是残酷的。使用这种方式自定义的布局，会存在与原生的通知栏消息样式不一致的可能，包括小图标/大图标的大小，字体的大小与颜色，时间的显示方式(不同版本的时间显示位置和样式都不一样)。下面解决一个最关键，也最致命的问题——字体颜色。如果字体颜色和背景颜色一样，那这条通知栏消息就没法看了，如RemoteViews显示异常一节介绍的一样。

解决字体颜色和背景颜色一样的问题有三种解决方案，分别是：

背景色固定不透明，字体颜色与背景色形成反差。（360和京东的做法） 背景色透明，字体颜色采用系统原生的notification\_style。 背景色透明，通过特殊方式拿到通知栏字体颜色和字体大小。



img

其中，第一种方案简单，能够兼容所有厂商机型。例如京东固定背景色为黑色，字体为红色。这种方式的唯一缺陷是样式上不能与普通通知栏消息重合，在白色背景的通知栏上极为显眼。第二种方式，通过阅读源码可知，系统的通知栏标题和内容采用的颜色分别是@android:color/primary\_text\_dark和@android:color/secondary\_text\_dark，但踩过坑之后发现并非所有的机型默认都是这两个颜色，有可能获取不到值。因此这种方案只能作为参考，不能用于实际环境中。最后详细介绍一下第三种方式。

#### **Android默认字体颜色获取**

这种方案有一点投机取巧，是网上寻找代替方案时在简书上找到的，作者是hackware。思路就是通过Notification.Builder生成一条空的Notification，但不调用notify()方法，然后通过这条Notification想办法获取里面的布局元素，通过遍历，就能拿到对应的字体和颜色了。具体看代码：

private static final String NOTIFICATION\_TITLE = "notification\_title";  
public static final int INVALID\_COLOR = -1; // 无效颜色  
private static int notificationTitleColor = INVALID\_COLOR; // 获取到的颜色缓存  
/\*\*  
 \* 获取系统通知栏主标题颜色，根据Activity继承自AppCompatActivity或FragmentActivity采取不同策略。  
 \*  
 \* @param context 上下文环境  
 \* @return 系统主标题颜色  
 \*/  
public static int getNotificationColor(Context context) {  
 try {  
 if (notificationTitleColor == INVALID\_COLOR) {  
 if (context instanceof AppCompatActivity) {  
 notificationTitleColor = getNotificationColorCompat(context);  
 } else {  
 notificationTitleColor = getNotificationColorInternal(context);  
 }  
 }  
 } catch (Exception ignored) {  
 }  
 return notificationTitleColor;  
}  
/\*\*  
 \* 通过一个空的Notification拿到Notification.contentView，通过{@link RemoteViews#apply(Context, ViewGroup)}方法返回通知栏消息根布局实例。  
 \*  
 \* @param context 上下文  
 \* @return 系统主标题颜色  
 \*/  
private static int getNotificationColorInternal(Context context) {  
 Notification.Builder builder = new Notification.Builder(context);  
 builder.setContentTitle(NOTIFICATION\_TITLE);  
 Notification notification = builder.build();  
 try {  
 ViewGroup root = (ViewGroup) notification.contentView.apply(context, new FrameLayout(context));  
 TextView titleView = (TextView) root.findViewById(android.R.id.title);  
 if (null == titleView) {  
 iteratorView(root, new Filter() {  
 @Override  
 public void filter(View view) {  
 if (view instanceof TextView) {  
 TextView textView = (TextView) view;  
 if (NOTIFICATION\_TITLE.equals(textView.getText().toString())) {  
 notificationTitleColor = textView.getCurrentTextColor();  
 }  
 }  
 }  
 });  
 return notificationTitleColor;  
 } else {  
 return titleView.getCurrentTextColor();  
 }  
 } catch (Exception e) {  
 DebugLog.e(e.getMessage());  
 return getNotificationColorCompat(context);  
 }  
}  
/\*\*  
 \* 使用getNotificationColorInternal()方法，Activity不能继承自AppCompatActivity（实测5.0以下机型可以，5.0及以上机型不行），  
 \* 大致的原因是默认通知布局文件中的ImageView（largeIcon和smallIcon）被替换成了AppCompatImageView，  
 \* 而在5.0及以上系统中，AppCompatImageView的setBackgroundResource(int)未被标记为RemotableViewMethod，导致apply时抛异常。  
 \*  
 \* @param context 上下文  
 \* @return 系统主标题颜色  
 \*/  
private static int getNotificationColorCompat(Context context) {  
 try {  
 Notification.Builder builder = new Notification.Builder(context);  
 Notification notification = builder.build();  
 int layoutId = notification.contentView.getLayoutId();  
 ViewGroup root = (ViewGroup) LayoutInflater.from(context).inflate(layoutId, null);  
 TextView titleView = (TextView) root.findViewById(android.R.id.title);  
 if (null == titleView) {  
 return getTitleColorIteratorCompat(root);  
 } else {  
 return titleView.getCurrentTextColor();  
 }  
 } catch (Exception e) {  
 }  
 return INVALID\_COLOR;  
}  
private static void iteratorView(View view, Filter filter) {  
 if (view == null || filter == null) {  
 return;  
 }  
 filter.filter(view);  
 if (view instanceof ViewGroup) {  
 ViewGroup viewGroup = (ViewGroup) view;  
 for (int i = 0; i < viewGroup.getChildCount(); i++) {  
 View child = viewGroup.getChildAt(i);  
 iteratorView(child, filter);  
 }  
 }  
}  
private static int getTitleColorIteratorCompat(View view) {  
 if (view == null) {  
 return INVALID\_COLOR;  
 }  
 List<TextView> textViews = getAllTextViews(view);  
 int maxTextSizeIndex = findMaxTextSizeIndex(textViews);  
 if (maxTextSizeIndex != Integer.MIN\_VALUE) {  
 return textViews.get(maxTextSizeIndex).getCurrentTextColor();  
 }  
 return INVALID\_COLOR;  
}  
private static int findMaxTextSizeIndex(List<TextView> textViews) {  
 float max = Integer.MIN\_VALUE;  
 int maxIndex = Integer.MIN\_VALUE;  
 int index = 0;  
 for (TextView textView : textViews) {  
 if (max < textView.getTextSize()) {  
 // 找到字号最大的字体，默认把它设置为主标题字号大小  
 max = textView.getTextSize();  
 maxIndex = index;  
 }  
 index++;  
 }  
 return maxIndex;  
}  
/\*\*  
 \* 实现遍历View树中的TextView，返回包含TextView的集合。  
 \*  
 \* @param root 根节点  
 \* @return 包含TextView的集合  
 \*/  
private static List<TextView> getAllTextViews(View root) {  
 final List<TextView> textViews = new ArrayList<>();  
 iteratorView(root, new Filter() {  
 @Override  
 public void filter(View view) {  
 if (view instanceof TextView) {  
 textViews.add((TextView) view);  
 }  
 }  
 });  
 return textViews;  
}  
  
private interface Filter {  
 void filter(View view);  
}

#### **RemoteViews适配方案**

获取系统通知标题颜色，如果能够获取到，那么标题、内容和时间的颜色都设置为标题颜色。 获取不到的情况下，遍历系统通知里的所有文字，取字号最大的那条文字的颜色作为标题、内容和时间的颜色。 以上两个步骤的实现在getNotificationColor()方法里。如果还获取不到，那么标题和内容采用Android原生系统提供的，其中标题是@android:color/primary\_text\_dark，内容是@android:color/secondary\_text\_dark。 有一点需要说明的是，以上适配只适合在Android 7.0以下系统。Android 7.0+修改了Notification，采用@android:color/primary\_text\_dark和@android:color/secondary\_text\_dark已经获取不到颜色值了，考虑到7.0所采用的通知栏主色调是白色，因此目前暂时的解决方案是遇到7.0的系统采用黑色字体。面对众多厂商的源码修改，目前测试有ZUK的7.0系统为暗色背景，暂时的解决方案是根据机型适配。

参考链接：<http://iluhcm.com/2017/03/12/experience-of-adapting-to-android-notifications/>