

Android O适配调研

**中国移动互联网基地融合通信项目组**

**2017年12月21日**

目录

[Android O适配调研 1](#_Toc501603740)

[1 Android O介绍 4](#_Toc501603741)

[2 Android O新特性 4](#_Toc501603742)

[2.1 通知变更 4](#_Toc501603743)

[2.2 自定义font资源 6](#_Toc501603744)

[2.3 自适应图标 6](#_Toc501603745)

[2.4 画中画模式 7](#_Toc501603746)

[2.5 自动填充框架 9](#_Toc501603747)

[2.6 Pinned Shortcut 11](#_Toc501603748)

[2.7 TextView字体自动适配 11](#_Toc501603749)

[2.8 媒体增强 11](#_Toc501603750)

[2.9 其它新特性 12](#_Toc501603751)

[3 Android O版本行为变更 13](#_Toc501603752)

[3.1 后台执行限制 13](#_Toc501603753)

[3.1.1 对Background Service 的限制 13](#_Toc501603754)

[3.1.2 对Broadcast 的限制 14](#_Toc501603757)

[3.2 安全性 14](#_Toc501603758)

[3.3 网络连接和 HTTP(S) 连接 14](#_Toc501603759)

[3.4 权限 15](#_Toc501603760)

[3.5 媒体变更 15](#_Toc501603761)

[3.6 Native libraries 15](#_Toc501603762)

[3.7 其他 16](#_Toc501603763)

[4 Android O 版本 API 变更 16](#_Toc501603764)

[4.1 WebView 新 API 16](#_Toc501603765)

[4.2 findViewById 17](#_Toc501603766)

[4.3 统一的 margins 和 padding 17](#_Toc501603767)

[4.4 AnimationSet 17](#_Toc501603768)

[4.5 提醒窗口 17](#_Toc501603769)

[5 参考资料： 18](#_Toc501603770)

# Android O介绍



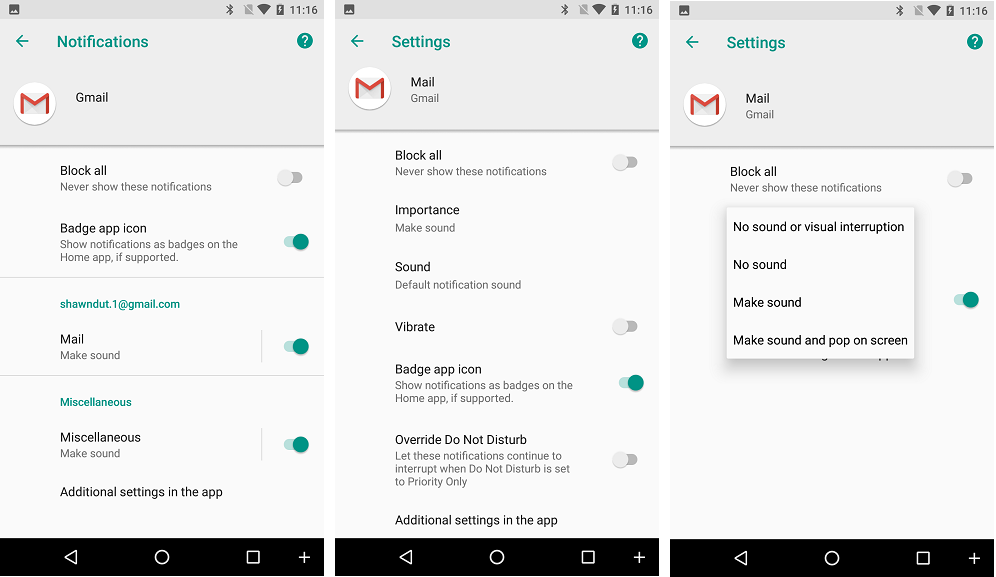


# Android O新特性

Android O 新版本更新和优化主要集中在两个方面：**Fluid Experiences** 和 **Vitals**，**Fluid Experience** 主要包含了四个显著特性：**Notification Dots, Picture In Picture, Autofill Framework 和 Smart Text Selection；**而 **Vitals** 主要在电池续航、安全、启动时间以及稳定性这几个方面做优化。

## 通知变更

在Android O 版本重新设计了通知，让通知的管理更加方便，首先引入了通知渠道用来让用户统一管理通知，当 targetApi 设置为 O 版本时，必须要实现一个或多个通知渠道，如果设置为 O 版本之前的版本，在 O 版本设备上的行为则和 7.x 版本保持一致。通知渠道提供了一种将应用内的通知变成一个个易于管理组的方法，通知渠道创建并提交到 notification manager 后，便无法通过编程方式修改通知渠道的行为，这些设置之后则由用户掌控。用户现在可以使用一致的系统 UI 管理大多数与通知有关的设置。所有发布至同一通知渠道的通知都具有相同的行为。当用户长摁通知或者通过设置去到通知的设置页面的时候，可以看到这些界面：



通过这个设置页面可以看到，用户可以将应用所有的通知渠道关闭，或者关闭某一个单独的通知渠道，每一个通知渠道也有相应的行为，用户可以去自行修改，比如修改通知渠道的优先级，如果支持的话，用户还可以去声明通知是否同时显示为应用的角标。Android O 还引入了通知角标，当一个应用的通知没有 dismiss 的时候，会在应用的图标上面显示一个角标，长摁这个应用的图标，也会显示通知的详情信息，这些更改包括：

通知渠道：Android O引入了通知渠道这个概念，可以为要显示的每种类型的通知创建用户可自定义的渠道。官方宣布的内容是“可以对特定通知进行分组”，而这些分组称作“渠道”（channel）。用户可以选择不同“渠道”的通知有不同的显示效果等等。

延后：用户可以让通知稍后再显示。通知重新出现，与他们第一次出现时的重要性相同。应用可以移除被延后的通知。

通知超时：使用Notification.Builder.setTimeout（）创建通知时设置超时。使用此方法指定一个持续时间，在该持续时间之后，通知应被取消。如果需要，可以在指定的超时时间过去之前取消通知。

通知解除：系统现在区分通知是否被用户解除还是被应用程序删除。要检查通知是否被关闭，就应该实现NotificationListenerService类的newonNotificationRemoved（）方法。

背景颜色：可以设置并启用通知的背景颜色，便于用户对不同的通知一目了然。例如，你可以为驾驶路线或正在进行的电话通知设置背景颜色。使用Notification.Builder.setColor（）设置所需的背景颜色。这样做允许您使用Notification.Builder.setColorized（）启用对通知的背景颜色的使用。

消息传递风格：使用MessagingStyle类的通知现在会以折叠形式显示更多内容，对消息相关的通知应该使用MessagingStyle类。还可以使用新的addHistoricMessage（）方法向消息传递相关通知添加历史消息。

**使用场景：针对应用的不同优先级别的通知设置不同的通知渠道，强提醒可以发出声音加震动，弱提醒可能仅仅只需要呼吸灯和角标这样等，这样可以最大程度减少对用户的干扰。**

## 自定义font资源

a、可下载字体

Android O（API level 26）和Support Library 26允许你从一个提供者应用请求字体，而不是将字体和APK绑定，或者，APK可以下载字体。这可以减小你的apk的大小，并且允许多个app共享同一份字体。

要获得下载字体的更多信息，看这里：[Downloadable Fonts](https://developer.android.com/guide/topics/ui/look-and-feel/downloadable-fonts.html" \t "_blank)

b、XML中字体

Android8.0(API level 26)引入了一个新特性，xml中的字体，允许将字体作为资源。这意味着，将不需要将字体绑定为assets。字体将被编译进R文件，并且被系统自动接受为资源。然后你可以通过新的资源类型，font，来访问这些字体。

Support library 26为那些运行运行api14及以上版本的设备提供了这一特性的完整支持。

关于xml中的字体，以及恢复系统字体，请访问这里： [Fonts in XML](https://developer.android.com/guide/topics/ui/look-and-feel/fonts-in-xml.html)

注：Android O支持.otf（OpenType）和.ttf（TrueType）格式

**适用场景：这个比较适合在公司的一系列 APP 上使用，更换皮肤同时更换字体等等。**

## 自适应图标

为了帮助开发者更好的与设备 UI 集成，Android O 支持创建自适应图标，系统可以基于设备选择的蒙版将这些图标显示为不同形状。系统还将实现与图标的自动交互，并在启动器、快捷方式、设置、共享对话框以及概览屏幕中使用它们，可以通过定义两层图层（前景和背景）来指定桌面图标外观，必须提供没有形状和阴影的 PNG 格式图象作为图层，在以前的 Android 版本中，图标大小定义为 48 x 48 dp，但是现在必须按照以下的规范定义图层大小：

1. 两张图层大小都必须为 108 x 108 dp。

2. 图层中心 72 x 72 dp 范围为可视范围。

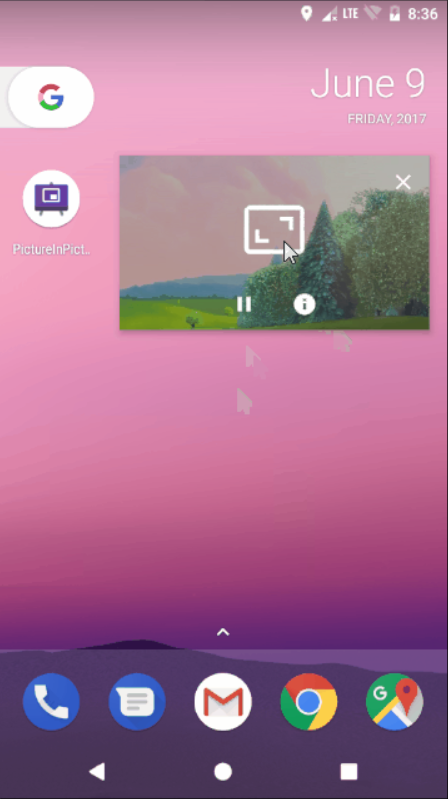
3. 系统会保留四周的 18dp 范围用于生成有趣的视觉效果（如视差和跳动）。

创建自适应图标可以参考：[Create adaptive icons with Android Studio](https://developer.android.com/preview/features/adaptive-icons.html" \l "studio" \t "_blank)

这个就将以前混乱的图标统一化了，现在 Android 很多应用的图标，有的是正方形的，有的带圆弧，而且这圆弧的弧度各有差异，有的图标还是圆形，有的则是不规则的图片，统一之后所有的应用图标将具有一致的风格，比较推荐。

## 画中画模式

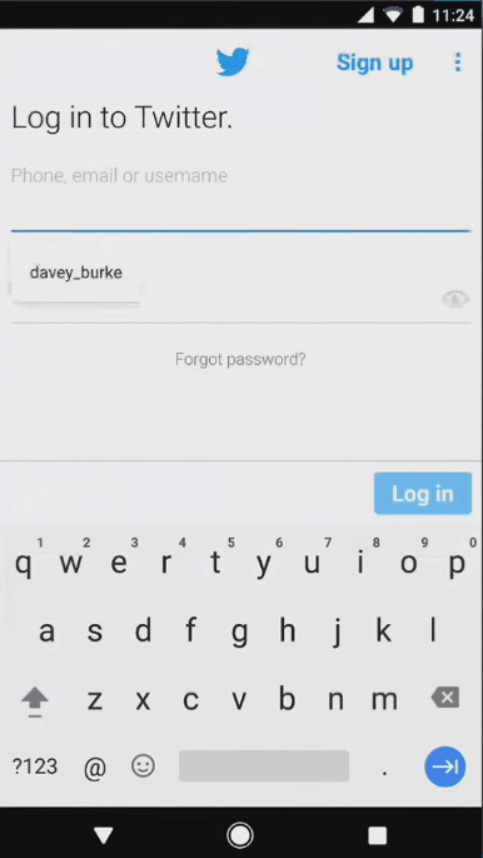
在 Android O 之前，画中画模式已经可用于 Android TV，而 Android O 则让这一个功能可以支持到其他的 Android 设备，当某个 Activity 处于 PIP 模式时，它会回调生命周期的 onPause 方法，所以此时在 PIP 模式下比如视频播放等操作就不能够在 onPause 里面暂停，而应该在 onStop 里面，这一点和多窗口模式一样，来看看它的效果：



要指定 Activity 可以使用 PIP 模式，需要在 Manifest文件中将 android:supportsPictureInPicture 设置为 true，除了上面提到的用户手动将 Activity 变成 PIP 模式，还可以使用 Activity.enterPictureInPictureMode(PictureInPictureArgs args) 动态将 Activity 设置为 PIP 模式， PictureInPictureArgs 这个对象参数是用来指定 Activity 处于 PIP 模式时的行为，此对象还指定了各种属性，例如 Activity 的首选纵横比，同时还可以使用 Activity.setPictureInPictureArgs() 更新 Activity 的 PIP 配置设置，如果 Activity 目前处于 PIP 模式，则会更新此设置。   
　　**使用场景：这个功能就很贴心了，比起以前的 Multi-window，这个更强调两个应用的主次之分，比较适合的场景可能有视频播放页面的最小化，地图应用的最小化等等。针对 QQ 音乐来说可以使用在直播页面，MV 播放页面等等。**

## 自动填充框架

Android O 还引入了自动填充框架，简化了用户在账号创建、登录和信用卡表单之类的填写工作，在用户选择自动填充框架之后，新老用户都可以使用自动填充框架，我们使用 Chrome 的时候已经体验过了自动填充用户名和密码的功能，只不过这次是在系统层面提供了这样的一种功能，可以快速的填充用户名，地址甚至密码等，而且用户也不需要去担心安全问题。   
　　同样自动填充框架也需要去设置页面打开： Settings > Apps & Notifications > Default apps > Autofill app，关于如何适配自动填充框架：[Autofill Framework](https://developer.android.com/preview/features/autofill.html" \t "_blank)：



这个场景可以参考 Chrome 的记住账号和密码功能，例如现在的 QQ 登录其实是已经做了一个类似的 Auto-Fill 功能，比较适合用在账号密码等场景上。

## Pinned Shortcut

这个功能在 7.1.1 之后就已经有了，长摁桌面图标就会显示一个 App shortcuts 菜单栏，然后长摁菜单栏上的一项，就能把它拖下来成为单独的一个应用图标，被拖下来的单独的应用图标就是Pinned shortcuts，虽然这个功能在 7.1.1 引入的，但是在 O 版本我们可以通过提供的 API 按照一定的步骤去使用代码添加： [Pinning Shortcuts and Widgets](https://developer.android.com/preview/features/pinning-shortcuts-widgets.html#shortcuts)

**适用场景：这个可以将某些应用使用频繁的小功能独立出来，比如 QQ 的最常用联系人，音乐的听歌识曲等功能。**

## TextView字体自动适配

Android O 版本允许设置 TextView 的字体大小根据设置的初始大小自动放大或者缩小，这样就可以让字体的显示在不同的屏幕和不同的显示内容上达到最优的效果，而且使用 Android support library 26 中的 android.support.v4.widget.TextViewCompat 可以让该特性支持到 14 版本，Android O 版本的 TextView 已经可以支持 autosize 了，设置 autosize 特性也非常简单，在 O 版本上，只需要使用 setAutoSizeTextTypeWithDefaults(@AutoSizeTextType int autoSizeTextType) 或者

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:autoSizeTextType="uniform"

/>

就可以了，关于使用的细节：[Autosizing TextViews](https://developer.android.com/preview/features/autosizing-textview.html" \t "_blank)

**适用场景：TextView 显示内容较多的时候，如果可以接受的话，适当的减小字体的大小，让显示的字数增加。**

## 媒体增强

Android O 版本新增 VolumeShaper 类，用来为应用提供声音的淡入淡出等音效；新增AudioFocusRequest 类用来提供检测音频焦点的新功能；新增了以下的方法 getMetrics 方法用来返回一个包含配置和性能信息的 Bundle 对象：

MediaPlayer.getMetrics()

MediaRecorder.getMetrics()

MediaCodec.getMetrics()

MediaExtractor.getMetrics()

　　MediaPlayer 新增了一些新的方法，这些方法可以用来增强应用处理媒体播放的能力：

通过控制缓冲行为改进性能的功能；

在搜索帧时进行精细控制；

播放受数字版权管理保护的材料的功能（和 MediaDrm 类功能类似，但是不同的是这个不会暴露底层的提取器，版权和加密数据：[DRM support](https://developer.android.com/preview/features/media-player.html" \l "drm" \t "_blank)）。

音频录制器 MediaRecorder 现在支持对流式传输有用的 MPEG2\_TS 格式，MediaMuxer 现在可以处理任意数量的音频和视频流，而不再仅限于一个音频曲目和/或一个视频曲目。使用 addTrack() 可混录所需的任意数量的曲目，MediaMuxer 还可以添加一个或多个包含用户定义的每帧信息的元数据曲目。元数据的格式由应用定义。仅对 MP4 容器支持元数据曲目。

## 其它新特性

1. 可以设置 Activity 支持广色域；   
　　2. 可以设置最大的屏幕宽高比；   
　　3. 多屏幕支持，支持设备外接一个显示器；   
　　4. 最新版本 emoji 支持，使用 EmojiCompat 类可以让应用在老版本的应用上显示新的 emoji；   
　　5. 支持点击位置的捕捉；   
　　6. 支持设置应用类别，这些类别用于将应用呈现给用户的用途或功能相同的应用归类在一起，例如按流量消耗、电池消耗和存储消耗将应用归类。   
　　7. Smart Text Selection，这个功能有人可能在今年老罗的锤子手机发布会上看到过，思想是很类似的，复制一段数字，就会出现直接拨打电话的选项，复制一段地址就会弹出进入地图的选项，地图 APP 可以一下这个地方能否成为一个很方便入口。

# Android O版本行为变更

## 后台执行限制

自从Nougat部署后台限制的一些基础,Android O开始花大量的精力去优化电量和交互行为.并加入了额外的后台限制, 主要包括这三个方面:implicit broadcasts, background services, and location updates,所以app在适配Android O版本时就需要去熟悉这些行为:background execution limits and background location limits。

当一个应用程序在后台运行，会消耗系统有限的资源，比如RAM。这可能导致糟糕的的用户体验，尤其是当用户使用的是资源密集型的应用程序，诸如玩游戏或观看视频。为了改善用户体验，Android O强加对后台运行的应用程序的限制。主要有下面两个方面：

Note: These limitations apply only to apps that target Android O. Apps that target API level 25 or lower are not affected.

1. 对Background Service 的限制

当一个应用程序在前台的时候，可以自由运行Foreground 和Background service。但是当应用程序切换到后台的时候，在开始几分钟所有的services都可以正常使用，过几分钟后，系统将会停止所有的Background service，就好比调用了Service.stopSelf( ) 。

在Android O之前，创建一个Foreground Service的方式通常是先创建一个Background Service的方式，再通过startForeground()的方式变为前台。但是在Android O中，如果应用程序运行在后台，那么就不能再创建Background Service，同时也就不能创建Foreground Service。出于这种考虑，Android O引入了一个新的方法：NotificationManager.startServiceInForeground( )。调用这个方法就相当于startService( )在后台创建一个服务，然后立即调用的服务的startForeground( )方法使其变为Foreground。由于新的服务从未在后台，所以不受后台服务的限制。

首先需要确定应用是前台应用还是后台应用，只要满足以下任意一个都认为是前台App：

（1）App有一个可见的Activiy

（2）App有一个前台是service

（3）App与其他前台App有交互，比如远程服务绑定、数据库监听等

      总之，前台App的service使用可以随心所欲，但是当应用退入后台进入idle状态时，系统会停止这个应用的service，停止方式与Service.stopSelf()方法等效。可惜还没固件，不然可以验证这个特性。App从前台刚转变成后台状态时，在有限的数分钟内还是可以启动后台服务的，这么做的原因主要是为了给App一个机会去处理任务。对于后台服务的受限，Android推荐使用JobScheduler jobs来替代曾经的后台服务。另外，提供了新的Context.startForegroundService()的方法来启动服务。

详细的适配指南：[后台执行限制](https://developer.android.google.cn/about/versions/oreo/background.html)

2. 对Broadcast 的限制

如果一个应用创建了一个广播接收器，每当这个receiver收到广播的时候，都会消耗系统资源。如果太多应用注册了某个系统事件的receiver，一个系统事件将会唤起所有的receiver，造成同时大量的系统资源被消耗，从而影响用户体验。为了解决这个问题，在Android N中添加了对广播的限制，在Android O这些限制更加严格了。

a、应用不能够再通过在AndroidManifest中声明广播的时候添加intent-filter来注册广播。

b、不管intent是显示还是隐式，都可继续使用Context.registerReceiver的方式注册广播。

## 安全性

Android O 包含以下与安全性有关的变更：

* 不再支持 SSLv3；
* 应用的 WebView 对象将在多进程模式下运行。网页内容在独立的进程中处理，此进程与包含应用的进程相隔离，以提高安全性；
* 在与未正确实现 TLS 协议版本协商的服务器建立 HTTPS 连接时，HttpsURLConnection 不再尝试回退到之前的 TLS 协议版本并重试的权宜方法；
* Android O 将使用安全计算 (SECCOMP) 过滤器来过滤所有应用。允许的系统调用列表仅限于通过 bionic 公开的系统调用。此外，还提供了其他几个后向兼容的系统调用，但我们不建议使用这些系统调用。

## 网络连接和 HTTP(S) 连接

　Android O 对网络连接和 HTTP(S) 连接行为做出了不少变更，其中包括无正文的 OPTIONS 请求现在有 Content-Length: 0 标头；HttpURLConnection 在包含斜线的主机或颁发机构名称后面附加一条斜线，将 <http://example.com> 转化为 <http://example.com/> ；通过 ProxySelector.setDefault() 设置的自定义代理选择器的范围变化；URI 不能包含空白标签；如果之前执行的 connect() 方法失败，send(java.net.DatagramPacket) 方法将会引发 SocketException；在回退到 TCP Echo 协议之前，InetAddress.isReachable() 会尝试执行 ICMP；隧道 HTTP(S) 连接处理进行了一些变更。   
　　变更内容比较繁多：[网络连接和 HTTP(S) 连接](https://developer.android.com/preview/behavior-changes.html#networking-all)。

## 权限

　在 Android O 之前，如果应用在运行时请求权限并且被授予该权限，系统会错误地将属于同一权限组并且在清单中注册的其他权限也一起授予应用。

对于针对 Android O 的应用，此行为已被纠正。系统只会授予应用明确请求的权限。然而，一旦用户为应用授予某个权限，则所有后续对该权限组中权限的请求都将被自动批准。

例如，假设某个应用在其清单中列出 READ\_EXTERNAL\_STORAGE 和 WRITE\_EXTERNAL\_STORAGE。应用请求 READ\_EXTERNAL\_STORAGE，并且用户授予了该权限。如果该应用针对的是 API 级别 24 或更低级别，系统还会同时授予 WRITE\_EXTERNAL\_STORAGE，因为该权限也属于同一 STORAGE 权限组并且也在清单中注册过。如果该应用针对的是 Android O，则系统此时仅会授予 READ\_EXTERNAL\_STORAGE；不过，如果该应用后来又请求 WRITE\_EXTERNAL\_STORAGE，则系统会立即授予该权限，而不会提示用户。   
媒体

具体可以参考：[Android8.0运行时权限策略变化和适配方案](http://blog.csdn.net/yanzhenjie1003/article/details/76719487)

## 媒体变更

* 使用 AudioTrack 时，如果应用请求了足够大的音频缓冲区，则框架将尝试使用深度缓冲区输出（如果可用）；
* 音频流类型应仅用于音量控制；所有其他流类型的使用（例如 AudioTrack 构造函数）仍有效，但系统会将其作为错误记录下来；
* 当用户打电话时，活动的媒体流将在通话期间静音；
* 所有与音频相关的 API 均使用 AudioAttributes 来描述音频播放用例；
* 框架会执行音频闪避，进行 AUDIOFOCUS\_GAIN\_TRANSIENT\_MAY\_DUCK 时，应用不会失去焦点。新的 API 适用于需要暂停而不是闪避的应用。不过，Android O 中未提供此行为。

## Native libraries

　在针对 Android O 的应用中，如果 Native libraries 包含任何可写且可执行的代码段，则不会再加载 Native libraries，可写和可执行必须是在新版本必须是互斥的，倘若某些应用的 Native libraries 包含不正确的加载代码段，则此变更可能会导致这些应用停止工作，官方文档：[Writable and Executable Segments](https://android.googlesource.com/platform/bionic/+/master/android-changes-for-ndk-developers.md#Writable-and-Executable-Segments-Enforced-for-API-level-26)。

## 其他

1. ContentProvider 支持分页，即获取内容的选中区域的子集；

2. ContentProvider 和 ContentResolver 增加 refresh 方法，用来让客户端更容易的知道数据是不是最新；

3. JobScheduler 更新，让应用更容易遵从后台执行限制；

4. 集合的处理的变化，AbstractCollection.removeAll() 和 AbstractCollection.retainAll() 始终引发 NullPointerException；

5. [语言区域和国际化变化](https://developer.android.com/preview/behavior-changes.html#lai)；

6. [联系人提供程序使用情况统计方法的变更](https://developer.android.com/preview/behavior-changes.html#cpu)；

7. [蓝牙 ScanRecord.getBytes() 方法检索的数据长度变更](https://developer.android.com/preview/behavior-changes.html#bt)；

8. [输入和导航](https://developer.android.com/preview/behavior-changes.html#ian)；

# Android O 版本 API 变更

## WebView 新 API

Android O 预览版本提供了几个新的 API 用来管理 WebView：

* Version API

第一个是提供获取 WebView 版本信息的 API:

PackageInfo webViewPackageInfo = WebView.getCurrentWebViewPackage();

Log.d("MY\_APP\_TAG", "WebView version: " + webViewPackageInfo.versionName);

Google Safe Browsing API可以再 Manifest 中配置 enable，然后在 WebView 打开未知不安全 url 的时候提示用户：

<manifest>

<meta-data android:name="android.webkit.WebView.EnableSafeBrowsing"

android:value="true" />

...

<application> ... </application>

</manifest>

Termination Handle API

WebView 绘制进程被杀或者 Crash 的回调；

Renderer Importance API

用来设置 WebView 绘制进程的优先级别，为了提供应用的稳定性，一般情况下应用不需要去修改绘制进程优先级，如果需要使用请和 Termination Handle API 一起搭配使用；

## findViewById

　　findViewById 函数现在返回的是 <T extends View>，所以以后 findViewById 就不需要强转了。

## 统一的 margins 和 padding

　　Android 引入了几个新的 xml 属性：

* `layout\_marginVertical`，同时设置 `layout\_marginTop` 和 `layout\_marginTop` 属性；
* `layout\_marginHorizontal`，同时设置 `layout\_marginLeft` 和 `layout\_marginRight`属性；
* `paddingVertical`，同时设置 `paddingTop` 和 `paddingBottom`属性；
* `paddingHorizontal`，同时设置 `paddingLeft` 和 `paddingRight`属性；

## AnimationSet

　　Android O 中，AnimationSet API 现在支持了动画的 seek 和动画倒转播放，seek 操作可以设置 AnimationSet 从指定的点开始播放，倒转播放则将以前需要重复定义两个相反的动画操作简化成只需要定义一个动画即可。

## 提醒窗口

　　在 Android O 版本之前，应用如果不用 type\_toast 显示悬浮窗，一般正常情况下都会使用声明 SYSTEM\_ALERT\_WINDOW 权限 + 使用 TYPE\_SYSTEM\_ERROR 的形式来显示悬浮窗，国内的第三方 ROM 也会对此有限制，Android 6.0 版本曾经对悬浮窗有过一次限制，必须要用户手动开启“在其他应用之上显示”的权限才能展示悬浮窗，在 Android O 之后 SYSTEM\_ALERT\_WINDOW 权限的应用无法再使用以下窗口类型来在其他应用和系统窗口上方显示提醒窗口：

* TYPE\_PHONE
* TYPE\_PRIORITY\_PHONE
* TYPE\_SYSTEM\_ALERT
* TYPE\_SYSTEM\_OVERLAY
* TYPE\_SYSTEM\_ERROR

相反，应用必须使用名为 TYPE\_APPLICATION\_OVERLAY 的新窗口类型，在使用 TYPE\_APPLICATION\_OVERLAY 窗口类型显示应用的提醒窗口时，也增加了一下限制：

* 应用的提醒窗口始终显示在状态栏和输入法等关键系统窗口的下面；
* 系统可以移动使用 TYPE\_APPLICATION\_OVERLAY 窗口类型的窗口或调整其大小，以改善屏幕显示效果；
* 通过打开通知栏，用户可以访问设置来阻止应用显示使用 TYPE\_APPLICATION\_OVERLAY 窗口类型显示的提醒窗口。

# 参考资料：

【1】[Android Oreo](https://developer.android.google.cn/about/versions/oreo/index.html#all-apps)

【2】[Android O适配](http://blog.csdn.net/tscyds/article/details/73614168)

【3】[Android 8.0新特性调研](http://blog.csdn.net/wei_lei/article/details/69473786)

【4】[Android 8.0 功能和 API](http://blog.csdn.net/u011324501/article/details/77532616)

【5】[Android O 功能和 API](https://www.cnblogs.com/yangjies145/p/7082578.html)

【6】[Android O新特性和行为变更总结](http://blog.csdn.net/self_study/article/details/74918289)

【7】[Android8.0运行时权限策略变化和适配方案](http://blog.csdn.net/yanzhenjie1003/article/details/76719487)

【8】[JobScheduler 改进](https://developer.android.google.cn/preview/api-overview.html#jobscheduler)

【9】[在Android O中使用自定义字体](http://blog.csdn.net/fjnu_se/article/details/72861554)

【10】[Downloadable Fonts](https://developer.android.google.cn/guide/topics/ui/look-and-feel/downloadable-fonts.html)