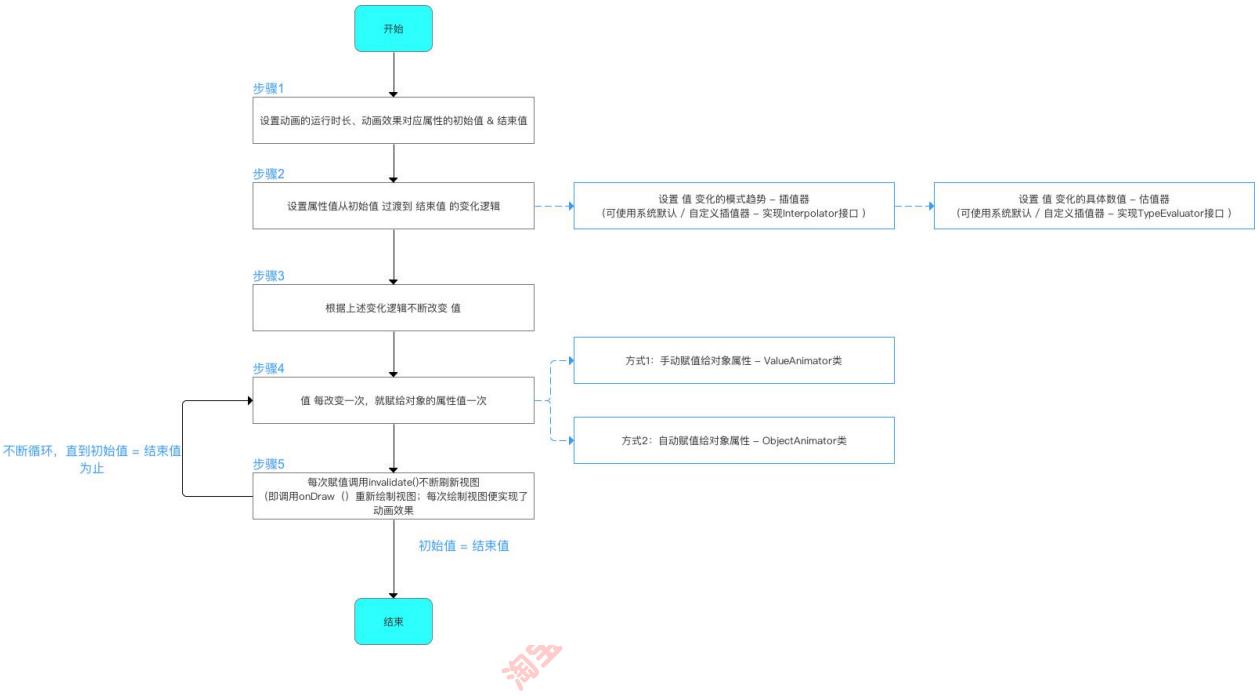
Android 源码相关面试专题

1、Android 属性动画实现原理

工作原理：在一定时间间隔内，通过不断对值进行改变，并不断将该值赋给对象的属性，从而实现该对象在该属性上的动画效果。



1）ValueAnimator：通过不断控制值的变化（初始值->结束值），将值手动赋值给对象的属性，再不断调用 View 的 invalidate()方法，去不断 onDraw 重绘 view，达到动画的效果。

主要的三种方法：

a) ValueAnimator.ofInt(int values)：估值器是整型估值器 IntEaluator

b) ValueAnimator.ofFloat(float values): 估值器是浮点型估值器 FloatEaluator

c) ValueAnimator.ofObject(ObjectEvaluator, start, end):将初始值以对象的形式过渡到结束值，通过操作对象实现动画效果，需要实现 Interpolator 接口，自定义估值器

估值器 TypeEvalutor，设置动画如何从初始值过渡到结束值的逻辑。插值器(Interpolator）决定值的变化模式（匀速、加速等）;估值器（TypeEvalutor）决定值的具体变化数值。


```
// 自定义估值器，需要实现 TypeEvaluator 接口
public class ObjectEvaluator implements TypeEvaluator{

// 复写 evaluate（ ），在 evaluate（ ）里写入对象动画过渡的逻辑
    @Override
    public Object evaluate(float fraction, Object startValue, Object endValue) {
        // 参数说明
        // fraction：表示动画完成度（根据它来计算当前动画的值）
        // startValue、endValue：动画的初始值和结束值
```

```
        ... // 写入对象动画过渡的逻辑

        return value;
        // 返回对象动画过渡的逻辑计算后的值
    }
```

2) ObjectAnimator:直接对对象的属性值进行改变操作，从而实现动画效果

ObjectAnimator 继承自 ValueAnimator 类，底层的动画实现机制还是基本值的改变。它是不断控制值的变化，再不断自动赋给对象的属性，从而实现动画效果。这里的自动赋值，是通过调用对象属性的 set/get 方法进行自动赋值，属性动画初始值如果有就直接取，没有则调用属性的 get()方法获取，当值更新变化时，通过属性的 set()方法进行赋值。每次赋值都是调用 view 的 postInvalidate()/invalidate()方法不断刷新视图（实际调用了 onDraw()方法进行了重绘视图）。

```
//Object 需要操作的对象； propertyName 需要操作的对象的属性； values 动画初始值&结束值，
//如果是两个值，则从 a->b 值过渡，如果是三值，则从 a->b->c
ObjectAnimator animator = ObjectAnimator.ofFloat(Object object, String propertyName, float ...values);
```

如果采用 ObjectAnimator 类实现动画，操作的对象的属性必须有 get()和 set()

方法。

其他用法：

1）AnimatorSet 组合动画

AnimatorSet.play(Animator anim)　：播放当前动画

AnimatorSet.after(long delay)　：将现有动画延迟 x 毫秒后执行

AnimatorSet.with(Animator anim)　：将现有动画和传入的动画同时执行

AnimatorSet.after(Animator anim)　：将现有动画插入到传入的动画之后执行

AnimatorSet.before(Animator anim)　：　将现有动画插入到传入的动画之前执行

2) ViewPropertyAnimator 直接对属性操作，View.animate()返回的是一个 ViewPropertyAnimator 对象，之后的调用方法都是基于该对象的操作，调用每个方法返回值都是它自身的实例

View.animate().alpha(0f).x(500).y(500).setDuration(500).setInterpolator()

3)设置动画监听

Animation.addListener(new AnimatorListener() {

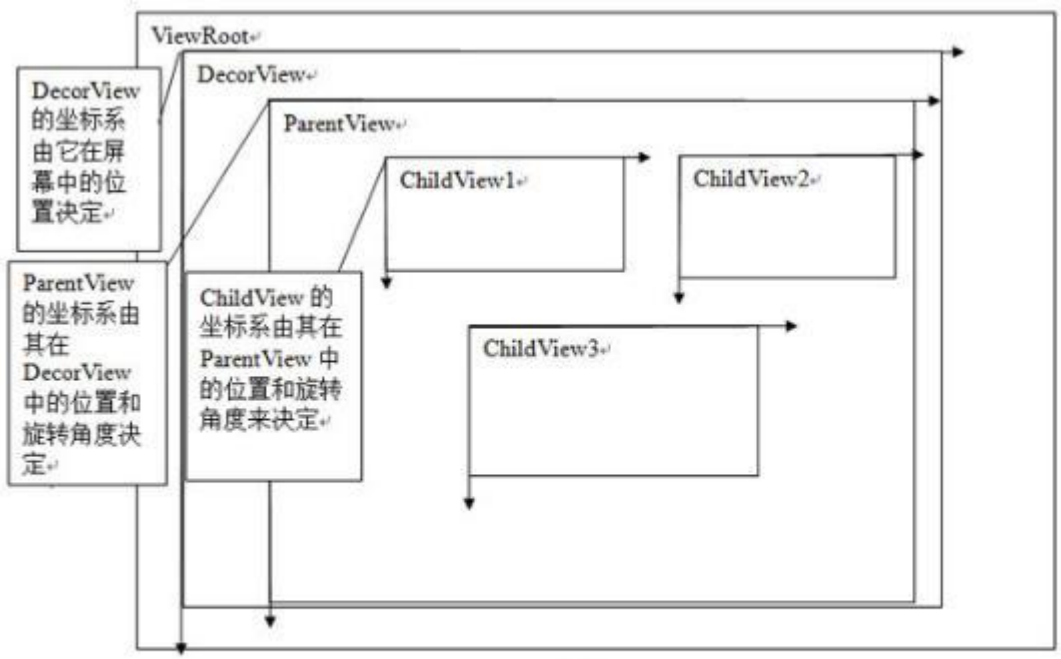
```
        @Override
        public void onAnimationStart(Animation animation) {
            //动画开始时执行
        }

        @Override
        public void onAnimationRepeat(Animation animation) {
            //动画重复时执行
        }

    @Override
        public void onAnimationCancel()(Animation animation) {
            //动画取消时执行
        }

        @Override
        public void onAnimationEnd(Animation animation) {
            //动画结束时执行
        }
    });
```

2、补间动画实现原理

主要有四种 AlpahAnimation\ ScaleAnimation\ RotateAnimation\ TranslateAnimation 四种，对透明度、缩放、旋转、位移四种动画。在调用 View.startAnimation 时，先调用 View.setAnimation(Animation)方法给自己设置一个 Animation 对象，再调用 invalidate 来重绘自己。在 View.draw(Canvas, ViewGroup, long)方法中进行了 getAnimation(), 并调用了 drawAnimation(ViewGroup, long, Animation, boolean)方法，此方法调用 Animation.getTranformation()方法，再调用 applyTranformation()方法，该方法中主要是对 Transformation.getMatrix().setTranslate/setRotate/setAlpha/setScale 来设置相应的值，这个方法系统会以 60FPS 的频率进行调用。具体是在调 Animation.start()方法中会调用 animationHandler.start()方法，从而调用了 scheduleAnimation() 方法，这里会调用 mChoreographer.postCallback(Choregrapher.CALLBACK_ANIMATION, this, null)放入事件队列中，等待 doFrame()来消耗事件。

当一个 ChildView 要重画时 ,它会调用其成员函数 invalidate() 函数将通知其 ParentView 这个 ChildView 要重画，这个过程一直向上遍历到 ViewRoot，当 ViewRoot 收到这个通知后就会调用 ViewRoot 中的 draw 函数从而完成绘制。View::onDraw() 有一个画布参数 Canvas, 画布顾名思义就是画东西的地方，Android 会为每一个 View 设置好画布，View 就可以调用 Canvas 的方法，比如：drawText, drawBitmap, drawPath 等等去画内容。每一个

ChildView 的画布是由其 ParentView 设置的，ParentView 根据 ChildView 在其内部的布局来调整 Canvas，其中画布的属性之一就是定义和 ChildView 相关的坐标系，默认是横轴为 X 轴，从左至右，值逐渐增大，竖轴为 Y 轴，从上至下，值逐渐增大。



Android 补间动画就是通过 ParentView 来不断调整 ChildView 的画布坐标系来实现的，在 ParentView 的 dispatchDraw 方法会被调用。

dispatchDraw()

{

....

Animation a = ChildView.getAnimation()

Transformation tm = a.getTransformation();

Use tm to set ChildView's Canvas;

Invalidate();

....

}

这里有两个类：Animation 和 Transformation，这两个类是实现动画的主要的类，Animation 中主要定义了动画的一些属性比如开始时间、持续时间、是否重复播放等，这个类主要有两个重要的函数：getTransformation 和 applyTransformation，在 getTransformation 中 Animation 会根据动画的属性来产生一系列的差值点，然后将这些差值点传给 applyTransformation，这个函数将根据这些点来生成不同的 Transformation，Transformation 中包含一个矩阵和 alpha 值，矩阵是用来做平移、旋转和缩放动画的，而 alpha 值是用来做 alpha 动画的（简单理解的话，alpha 动画相当于不断变换透明度或颜色来实现动画），调用 dispatchDraw 时会调用 getTransformation 来得到当前的 Transformation。某一个 View 的动画的绘制并不是由他自己完成的而是由它的父 view 完成。

1）补间动画 TranslateAnimation,View 位置移动了，可是点击区域还在原来的位置，为什么？

View 在做动画是，根据动画时间的插值，计算出一个 Matrix，不停的 invalidate，

在 onDraw 中的 Canvas 上使用这个计算出来的 Matrix 去 draw view 的内容。某个 view 的动画绘制并不是由它自己完成，而是由它的父 view 完成，使它的父 view 画布进行了移动，而点击时还是点击原来的画布。使得它看起来变化了。

3、Android 各个版本 API 的区别

主要记住一些大版本变化:

android3.0 代号 Honeycomb，引入 Fragments, ActionBar,属性动画，硬件加速

android4.0 代号 Ice Cream，API14：截图功能，人脸识别，虚拟按键，3D 优化驱动

android5.0 代号 Lollipop API21：调整桌面图标及部件透明度等

android6.0 代号 M Marshmallow API23，软件权限管理，安卓支付，指纹支持，App 关联，

android7.0 代号 N Preview API24，多窗口支持(不影响 Activity 生命周期)，增加了 JIT 编译器，引入了新的应用签名方案 APK Signature Scheme v2（缩

短应用安装时间和更多未授权 APK 文件更改保护）,严格了权限访问

android8.0 代号 O　API26，取消静态广播注册，限制后台进程调用手机资源，桌面图标自适应

android9.0 代号 P API27，加强电池管理，系统界面添加了 Home 虚拟键，提供人工智能 API，支持免打扰模式

4、Requestlayout，onlayout，onDraw，DrawChild 区别与联系

requestLayout()方法 ：会导致调用 measure()过程 和 layout()过程 。 说明：只是对 View 树重新布局 layout 过程包括 measure()和 layout()过程 如果 view 的 l,t,r,b 没有必变，那就不会触发 onDraw；但是如果这次刷新是在动画里，mDirty 非空，就会导致 onDraw。

onLayout()方法(如果该 View 是 ViewGroup 对象，需要实现该方法，对每个子视图进行布局)

onDraw()方法绘制视图本身 (每个 View 都需要重载该方法，ViewGroup 不需要实现该方法)

drawChild()去重新回调每个子视图的 draw()方法

5、invalidate 和 postInvalidate 的区别及使用

View.invalidate(): 层层上传到父级，直到传递到 ViewRootImpl 后触发了 scheduleTraversals()，然后整个 View 树开始重新按照 View 绘制流程进行重绘任务。

invalidate:在 ui 线程刷新 view

postInvalidate：在工作线程刷新 view（底层还是 handler）其实它的原理就是 invalidate+handler

View.postInvalidate 最终会调用 ViewRootImpl.dispatchInvalidateDelayed() 方法

```
public void dispatchInvalidateDelayed(View view, long delayMilliseconds)
{
        Message msg = mHandler.obtainMessage(MSG_INVALIDATE, view);
        mHandler.sendMessageDelayed(msg, delayMilliseconds);
    }
```

这里的 mHandler 是 ViewRootHandler 实例，在该 Handler 的 handleMessage 方法中调用了 view.invalidate()方法。

```
case MSG_INVALIDATE:
```

```
((View) msg.obj).invalidate();
break;
```

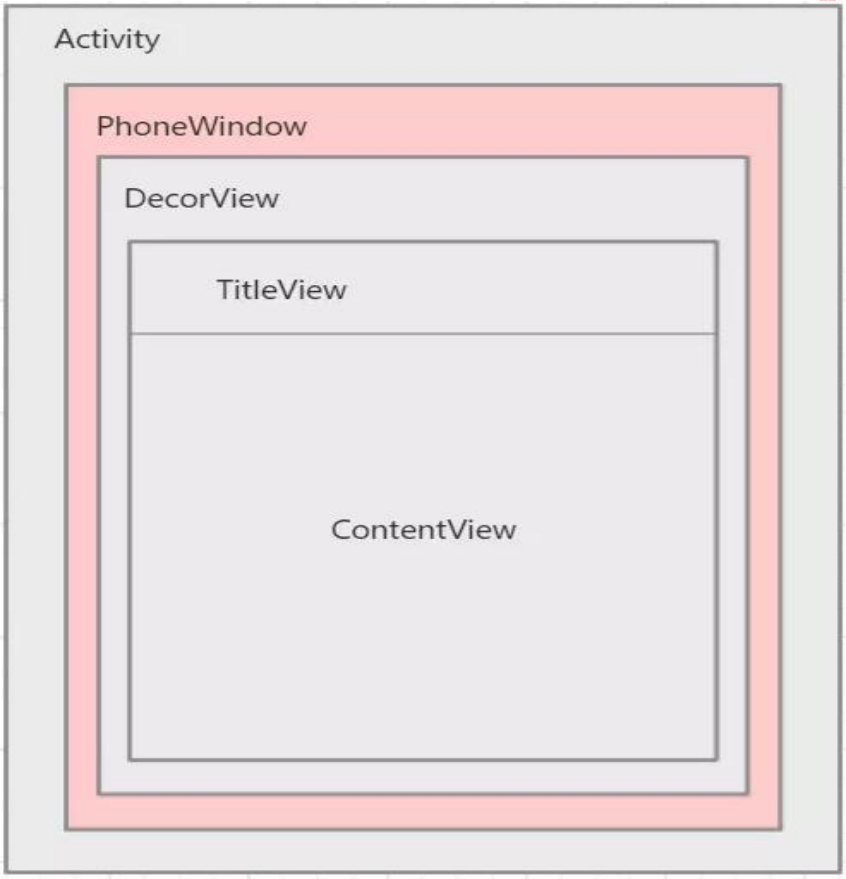
6、Activity-Window-View 三者的差别

Activity：是安卓四大组件之一，负责界面展示、用户交互与业务逻辑处理；

Window：就是负责界面展示以及交互的职能部门，就相当于 Activity 的下属，Activity 的生命周期方法负责业务的处理；

View：就是放在 Window 容器的元素，Window 是 View 的载体，View 是 Window 的具体展示。

三者的关系： Activity 通过 Window 来实现视图元素的展示，window 可以理解为一个容器，盛放着一个个的 view，用来执行具体的展示工作。



7、谈谈对 Volley 的理解

8、如何优化自定义 View

1）在要在 onDraw 或是 onLayout()中去创建对象，因为 onDraw()方法可能会被频繁调用，可以在 view 的构造函数中进行创建对象；

2）降低 view 的刷新频率，尽可能减少不必要的调用 invalidate()方法。或是调用带四种参数不同类型的 invalidate()，而不是调用无参的方法。无参变量需要刷新整个 view，而带参数的方法只需刷新指定部分的 view。在 onDraw()方法中减少冗余代码。

3）使用硬件加速，GPU 硬件加速可以带来性能增加。

4）状态保存与恢复，如果因内存不足，Activity 置于后台被杀重启时，View 应尽可能保存自己属性，可以重写 onSaveInstanceState 和 onRestoreInstanceState 方法，状态保存。

9、低版本 SDK 如何实现高版本 api？

使用@TargetApi 注解·

当代码中有比 AndroidManifest 中设置的 android:minSdkVersion 版本更高

的方法，此时编译器会提示警告，解决方法是在方法上加上@SuppressLint("NewApi"）或者@TargetApi()。但它们仅是屏蔽了 android lint 错误，在方法中还要判断版本做不同的操作。

@SuppressLint("NewApi"）屏蔽一切新 api 中才能使用的方法报的 android lint 错误

@TargetApi() 只屏蔽某一新 api 中才能使用的方法报的 android lint 错误，如@TargetApi(11)如果在方法中用了只有 API14 才开始有的方法，还是会报错。

10、描述一次网络请求的流程

1)域名解析

浏览器会先搜索自身 DNS 缓存且对应的 IP 地址没有过期；若未找到则搜索操作系统自身的 DNS 缓存；若还未找到则读本地的 hotsts 文件；还未找到会在 TCP/IP 设置的本地 DNS 服务器上找，如果要查询的域名在本地配置的区域资源中，则完成解析；否则根据本地 DNS 服务器会请求根 DNS 服务器；根 DNS 服务器是 13 台根 DNS，会一级一级往下找。

2）TCP 三次握手

客户端先发送 SYN=1，ACK=0，序列号 seq=x 报文；（SYN 在连接建立时用

来同步序号，SYN=1，ACK=0 代表这是一个连接请求报文，对方若同意建立连接，则应在响应报文中使 SYN=1，ACK=1）

服务器返回 SYN=1，ACK=1，seq=y, ack=x+1；

客户端再一次确认，但不用 SYN 了，回复服务端, ACK=1, seq=x+1, ack=y+1

3）建立 TCP 连接后发起 HTTP 请求

客户端按照指定的格式开始向服务端发送 HTTP 请求，HTTP 请求格式由四部分组成，分别是请求行、请求头、空行、消息体，服务端接收到请求后，解析 HTTP 请求，处理完成逻辑，最后返回一个具有标准格式的 HTTP 响应给客户端。

4）服务器响应 HTTP 请求

服务器接收处理完请求后返回一个 HTTP 响应消息给客户端，HTTP 响应信息格式包括：状态行、响应头、空行、消息体

5）浏览器解析 HTML 代码，请求 HTML 代码中的资源

浏览器拿到 html 文件后，就开始解析其中的 html 代码，遇到 js/css/image 等静态资源时，向服务器发起一个 http 请求，如果返回 304 状态码，浏览器会直

接读取本地的缓存文件。否则开启线程向服务器请求下载。

6）浏览器对页面进行渲染并呈现给用户

7）TCP 的四次挥手

当客户端没有东西要发送时就要释放连接（提出中断连接可以是 Client 也可以是 Server），客户端会发送一个 FIN=1 的没有数据的报文，进入 FIN_WAIT 状态，服务端收到后会给客户端一个确认，此时客户端不能发送数据，但可接收信息。

11、HttpUrlConnection 和 okhttp 关系

两者都可以用来实现网络请求，android4.4 之后的 HttpUrlConnection 的实现是基于 okhttp

-Bitmap 对象的理解

-looper 架构

-ActivityThread，AMS，WMS 的工作原理

-自定义 View 如何考虑机型适配

在 onMeasure()的 getDefaultSize()的默认实现中，当 view 的测量模式是 AT_MOST 或 EXACTLY 时，View 的大小都会被设置成子 View MeasureSpec 的 specSize.子 view 的 MeasureSpec 值是根据子 View 的布局参数和父容器的

MeasureSpec 值计算得来。当子 view 的布局参数是 wrap_content 时，对应的测量模式是 AT_MOST，大小是 parentSize,

-自定义 View 的事件

-AstncTask+HttpClient 与 AsyncHttpClient 有什么区别？

-LaunchMode 应用场景

-AsyncTask 如何使用?

-SpareArray 原理

-请介绍下 ContentProvider 是如何实现数据共享的？

-AndroidService 与 Activity 之间通信的几种方式

-IntentService 原理及作用是什么？

原理：IntentService 是继承 Service 的一个抽象类，它在 onCreate()方法中创建了一个 HandlerThread,并启动该线程。HandlerThread 是带有自己消息队列和 Looper 的线程，根据 HandlerThread 的 looper 创建一个 Handler，这样 IntentService 的 ServiceHandler 的 handleMessage()方法就运行在子线程中。handleMessage 中调用了 onHandleIntent()方法，它是一个抽象方法，继承 IntentService 类需要实现该方法，把耗时操作放在 onHandleIntent()方法中，等耗时操作运行完成后，会调用 stopSelf()方法，服务会调用 onDestory()方法消毁自己。如果 onHandleIntent()中的耗时操作未运行完前就调用了 stopSelf()方法，服务调用 onDestory()方法，但耗时操作会继续运行，直至运行完毕。如果同时多次启动 IntentService，任务会放在一个队列中，onCreate()和 onDestory()方法都只会运行一次。

作用：用来处理后台耗时操作，如读取数据库或是本地文件等。

-说说 Activity、Intent、Service 是什么关系

-ApplicationContext 和 ActivityContext 的区别

-SP 是进程同步的吗?有什么方法做到同步？

-谈谈多线程在 Android 中的使用

-进程和 Application 的生命周期

-封装 View 的时候怎么知道 view 的大小

-RecycleView 原理

-AndroidManifest 的作用与理解