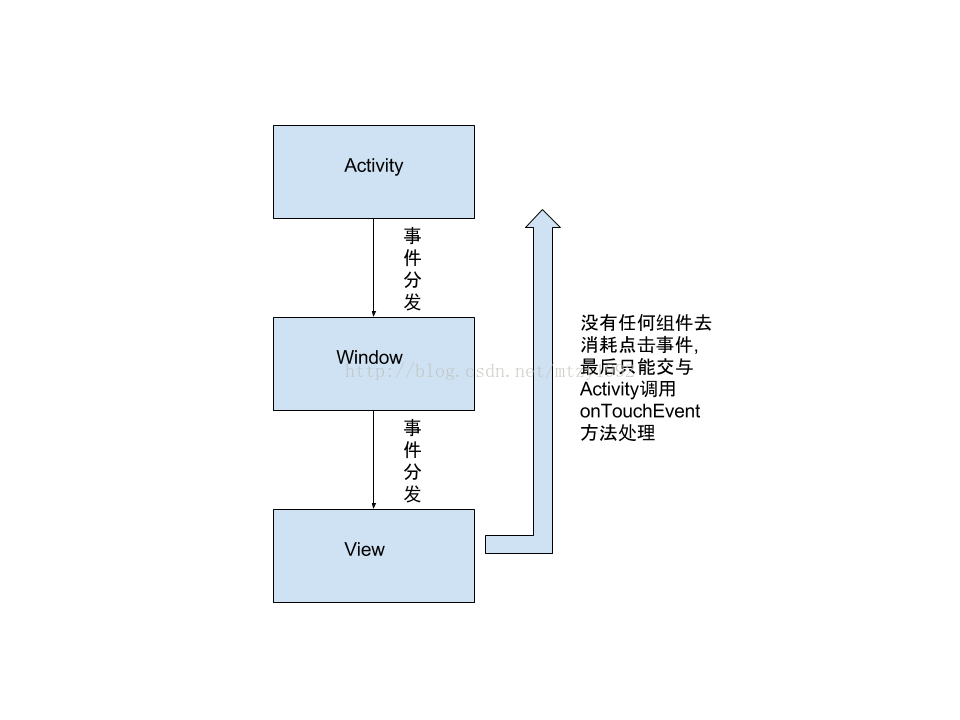
View事件分发流程：

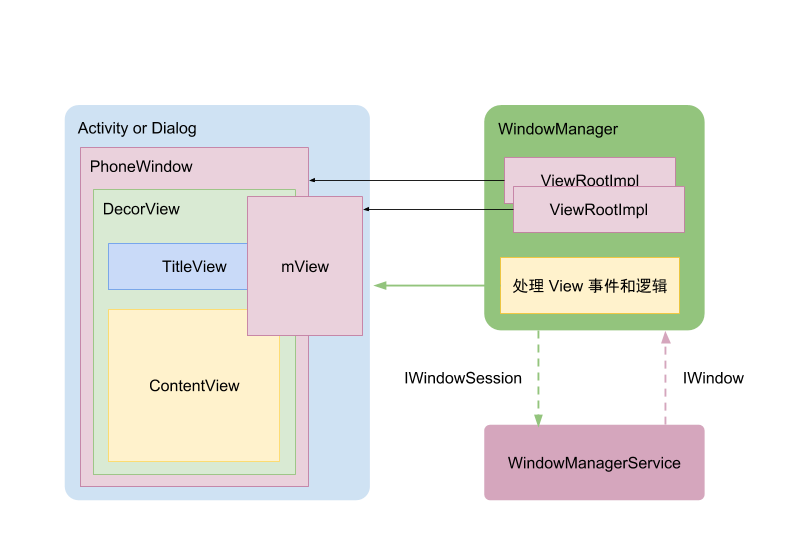


事件分发流程：

Activity dispatchTouchEvent 开始

Activity onTouchEvent 结束

Window窗口



事件由Activity dispatchTouchEvent 开始传递

经过DecorView默认向下传递

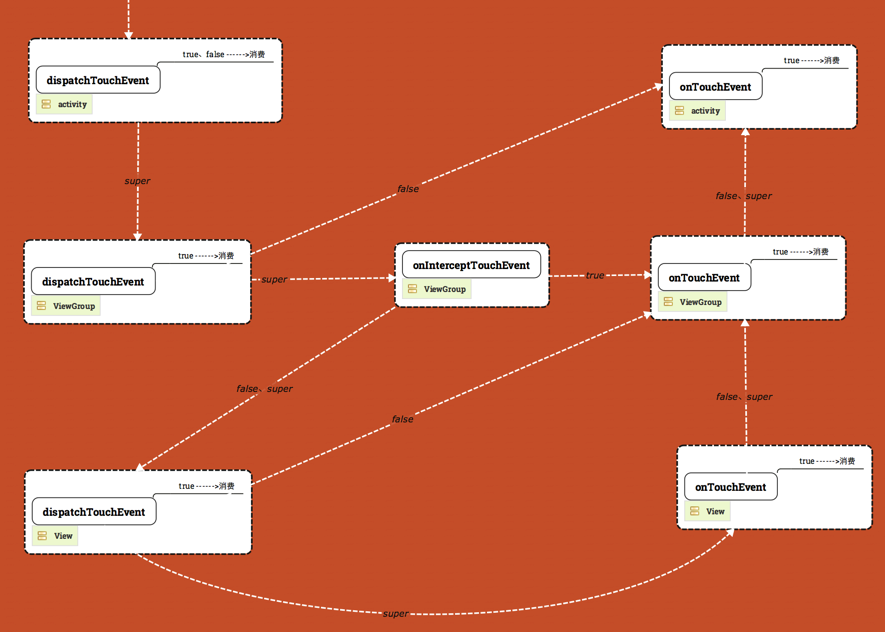
假设contentView有 ViewGroupA,ViewB

嵌套关系，Activity（A（B））

Activity 有2个方法：DispatchTouchEvent , OnTouchEvent

ViewGroup有3个方法：DispatchTouchEvent , OnInterruptTouchEvent,OnTouchEvent

View有2个方法：DispatchTouchEvent , OnTouchEvent



需要注意的方法如下：

1. diapstchTouchEvent返回true表示自身事件被消费，事件至此不再传递

返回false表示事件由上级view的onTouchEvent处理，本身以及子view不会收到后续传递的事件

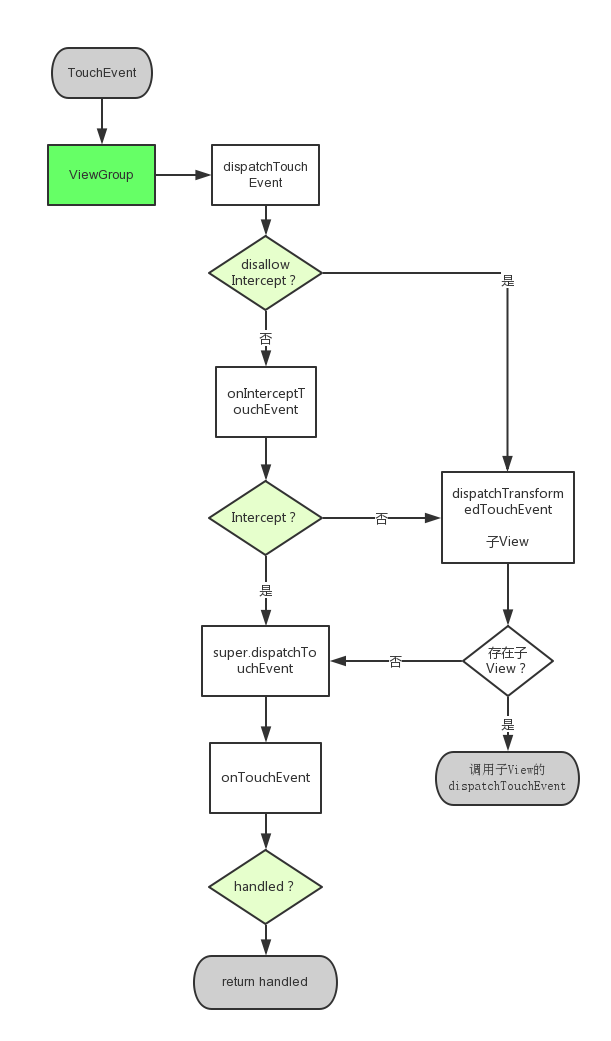
2. onInterceptTouchEvent如果返回true，自身的onInterceptTouchEvent不再执行，onInterceptTouchEvent由diapstchTouchEvent内部执行

返回false/super都会继续向下传递

3. onTouchEvent返回true，事件由该view处理，至此不再向上传递

其他：

1. requestDisallowInterceptTouchEvent 方法：



disallowIntercept的作用  
ViewGroup有一个disallowIntercept开关，可以设置此ViewGroup是否屏蔽onInterceptTouchEvent事件。如果开启此开关，则此ViewGroup跳过自身的onInterceptTouchEvent事件，直接dispatchTouchEvent到子View。  
重置disallowIntercept  
disallowIntercept，会在每次ACTION\_DOWN被重置，默认为允许调用onInterceptTouchEvent

滑动冲突事件：

自定义ViewGroup中嵌套ListView导致重写的onInterceptTouchEvent只能接收到一下ACTION\_DOWN事件，后续该方法不会回调，为什么？

ListView在收到ACTION\_DOWN事件之后判断是否可以滑动，调用requestDisallowInterceptTouchEvent阻止了父类后续调用onInterceptTouchEvent，导致父类无法直接拦截后续方法，这个主要描述为：ListView需要处理后续事件，父类不可以再拦截后续事件，直到一次完整的滑动结束，如果ListView处理不完，会考虑onTouchEvent继续向上传递剩余的滑动量

2. 嵌套滚动NestedScroll

一般的触摸消息的分发都是从外向内的,由外层的ViewGroup的dispatchTouchEvent方法调用到内层的View的dispatchTouchEvent方法.

而NestedScroll提供了一个反向的机制,内层的view在接收到ACTION\_MOVE的时候,将滚动消息先传回给外层的ViewGroup,看外层的ViewGroup是不是需要消耗一部分的移动,然后内层的View再去消耗剩下的移动.内层view可以消耗剩下的滚动的一部分,如果还没有消耗完,外层的view可以再选择把最后剩下的滚动消耗掉.

