public class
View
extends Object
implements Drawable.Callback KeyEvent.Callback AccessibilityEventSource

Summary: Nested Classes | XML Attrs | Constants | Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]
Added in API level 1

java.lang.Object
↳android.view.View

Known Direct Subclasses
AnalogClock, ImageView, KeyboardView, MediaRouteButton, ProgressBar, Space, SurfaceView, TextView, TextureView, ViewGroup, ViewStub

Known Indirect Subclasses
AbsListView, AbsSeekBar, AbsSpinner, AbsoluteLayout, AdapterView<T extends Adapter>, AdapterViewAnimator, AdapterViewFlipper, and 56 others.

Class Overview

This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, which are used to create interactive UI components (buttons, text fields, etc.). The `ViewGroup` (/reference/android/view/ViewGroup.html) subclass is the base class for *layouts*, which are invisible containers that hold other Views (or other ViewGroups) and define their layout properties.

Developer Guides

For information about using this class to develop your application's user interface, read the User Interface (/guide/topics/ui/index.html) developer guide.

Using Views

All of the views in a window are arranged in a single tree. You can add views either from code or by specifying a tree of views in one or more XML layout files. There are many specialized subclasses of views that act as controls or are capable of displaying text, images, or other content.

Once you have created a tree of views, there are typically a few types of common operations you may wish to perform:

- **Set properties:** for example setting the text of a `TextView`. The available properties and the methods that set them will vary among the different subclasses of views. Note that properties that are known at build time can be set in the XML layout files.
- **Set focus:** The framework will handled moving focus in response to user input. To force focus to a specific view, call `requestFocus()`.
- **Set up listeners:** Views allow clients to set listeners that will be notified when something interesting happens to the view. For example, all views will let you set a listener to be notified when the view gains or loses focus. You can register such a listener using `setOnFocusChangeListener(android.view.View.OnFocusChangeListener)`. Other view subclasses offer more specialized listeners. For example, a Button exposes a listener to notify clients when the button is clicked.
- **Set visibility:** You can hide or show views using `setVisibility(int)`.

Note: The Android framework is responsible for measuring, laying out and drawing views. You should not call methods that perform these actions on views yourself unless you are actually implementing a `ViewGroup` (/reference/android/view/ViewGroup.html).

Implementing a Custom View

To implement a custom view, you will usually begin by providing overrides for some of the standard methods that the framework calls on all views. You do not need to override all of these methods. In fact, you can start by just overriding `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)).

Category	Methods	Description
Creation	Constructors	There is a form of the constructor that are called when the view is created from code and a form that is called when the view is inflated from a layout file. The second form should parse and apply any attributes defined in the layout file.
	`onFinishInflate()`	Called after a view and all of its children has been inflated from XML.
Layout	`onMeasure(int, int)`	Called to determine the size requirements for this view and all of its children.
	`onLayout(boolean, int, int, int, int)`	Called when this view should assign a size and position to all of its children.
	`onSizeChanged(int, int, int, int)`	Called when the size of this view has changed.
Drawing	`onDraw(android.graphics.Canvas)`	Called when the view should render its content.
Event processing	`onKeyDown(int, KeyEvent)`	Called when a new hardware key event occurs.
	`onKeyUp(int, KeyEvent)`	Called when a hardware key up event occurs.
	`onTrackballEvent(MotionEvent)`	Called when a trackball motion event occurs.
	`onTouchEvent(MotionEvent)`	Called when a touch screen motion event occurs.
Focus	`onFocusChanged(boolean, int, android.graphics.Rect)`	Called when the view gains or loses focus.
	`onWindowFocusChanged(boolean)`	Called when the window containing the view gains or loses focus.
Attaching	`onAttachedToWindow()`	Called when the view is attached to a window.
	`onDetachedFromWindow()`	Called when the view is detached from its window.
	`onWindowVisibilityChanged(int)`	Called when the visibility of the window containing the view has changed.

IDs

Views may have an integer id associated with them. These ids are typically assigned in the layout XML files, and are used to find specific views within the view tree. A common pattern is to:

- Define a Button in the layout file and assign it a unique ID.

```
<Button
    android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_button_text"/>
```

- From the onCreate method of an Activity, find the Button

```
     Button myButton = (Button) findViewById(R.id.my_button);
```

View IDs need not be unique throughout the tree, but it is good practice to ensure that they are at least unique within the part of the tree you are searching.

Position

The geometry of a view is that of a rectangle. A view has a location, expressed as a pair of *left* and *top* coordinates, and two dimensions, expressed as a width and a height. The unit for location and dimensions is the pixel.

It is possible to retrieve the location of a view by invoking the methods `getLeft()` (/reference/android/view/View.html#getLeft()) and `getTop()` (/reference/android/view/View.html#getTop()). The former returns the left, or X, coordinate of the rectangle representing the view. The latter returns the top, or Y, coordinate of the rectangle representing the view. These methods both return the location of the view relative to its parent. For instance, when getLeft() returns 20, that means the view is located 20 pixels to the right of the left edge of its direct parent.

In addition, several convenience methods are offered to avoid unnecessary computations, namely `getRight()` (/reference/android/view/View.html#getRight()) and `getBottom()` (/reference/android/view/View.html#getBottom()). These methods return the coordinates of the right and bottom edges of the rectangle representing the view. For instance, calling `getRight()` (/reference/android/view/View.html#getRight()) is similar to the following computation: `getLeft() + getWidth()` (see Size (#SizePaddingMargins) for more information about the width.)

Size, padding and margins

The size of a view is expressed with a width and a height. A view actually possess two pairs of width and height values.

The first pair is known as *measured width* and *measured height*. These dimensions define how big a view wants to be within its parent (see Layout (#Layout) for more details.) The measured dimensions can be obtained by calling `getMeasuredWidth()` (/reference/android/view/View.html#getMeasuredWidth()) and `getMeasuredHeight()` (/reference/android/view/View.html#getMeasuredHeight()).

The second pair is simply known as *width* and *height*, or sometimes *drawing width* and *drawing height*. These dimensions define the actual size of the view on screen, at drawing time and after layout. These values may, but do not have to, be different from the measured width and height. The width and height can be obtained by calling `getWidth()` (/reference/android/view/View.html#getWidth()) and `getHeight()` (/reference/android/view/View.html#getHeight()).

To measure its dimensions, a view takes into account its padding. The padding is expressed in pixels for the left, top, right and bottom parts of the view. Padding can be used to offset the content of the view by a specific amount of pixels. For instance, a left padding of 2 will push the view's content by 2 pixels to the right of the left edge. Padding can be set using the `setPadding(int, int, int, int)` (/reference/android/view/View.html#setPadding(int, int, int, int)) or `setPaddingRelative(int, int, int, int)` (/reference/android/view/View.html#setPaddingRelative(int, int, int, int)) method and queried by calling `getPaddingLeft()` (/reference/android/view/View.html#getPaddingLeft()), `getPaddingTop()` (/reference/android/view/View.html#getPaddingTop()), `getPaddingRight()` (/reference/android/view/View.html#getPaddingRight()), `getPaddingBottom()` (/reference/android/view/View.html#getPaddingBottom()), `getPaddingStart()` (/reference/android/view/View.html#getPaddingStart()), `getPaddingEnd()` (/reference/android/view/View.html#getPaddingEnd()).

Even though a view can define a padding, it does not provide any support for margins. However, view groups provide such a support. Refer to `ViewGroup` (/reference/android/view/ViewGroup.html) and `ViewGroup.MarginLayoutParams` (/reference/android/view/ViewGroup.MarginLayoutParams.html) for further information.

Layout

Layout is a two pass process: a measure pass and a layout pass. The measuring pass is implemented in `measure(int, int)` (/reference/android/view/View.html#measure(int, int)) and is a top-down traversal of the view tree. Each view pushes dimension specifications down the tree during the recursion. At the end of the measure pass, every view has stored its measurements. The second pass happens in `layout(int, int, int, int)` (/reference/android/view/View.html#layout(int, int, int, int)) and is also top-down. During this pass each parent is responsible for positioning all of its children using the sizes computed in the measure pass.

When a view's measure() method returns, its `getMeasuredWidth()` (/reference/android/view/View.html#getMeasuredWidth()) and `getMeasuredHeight()` (/reference/android/view/View.html#getMeasuredHeight()) values must be set, along with those for all of that view's descendants. A view's measured width and measured height values must respect the constraints imposed by the view's parents. This guarantees that at the end of the measure pass, all parents accept all of their children's measurements. A parent view may call measure() more than once on its children. For example, the parent may measure each child once with unspecified dimensions to find out how big they want to be, then call measure() on them again with actual numbers if the sum of all the children's unconstrained sizes is too big or too small.

The measure pass uses two classes to communicate dimensions. The `View.MeasureSpec` (/reference/android/view/View.MeasureSpec.html) class is used by views to tell their parents how they want to be measured and positioned. The base LayoutParams class just describes how big the view wants to be for both width and height. For each dimension, it can specify one of:

- an exact number
- MATCH_PARENT, which means the view wants to be as big as its parent (minus padding)
- WRAP_CONTENT, which means that the view wants to be just big enough to enclose its content (plus padding).
  There are subclasses of LayoutParams for different subclasses of ViewGroup. For example, AbsoluteLayout has its own subclass of LayoutParams which adds an X and Y value.

MeasureSpecs are used to push requirements down the tree from parent to child. A MeasureSpec can be in one of three modes:

- UNSPECIFIED: This is used by a parent to determine the desired dimension of a child view. For example, a LinearLayout may call measure() on its child with the height set to UNSPECIFIED and a width of EXACTLY 240 to find out how tall the child view wants to be given a width of 240 pixels.
- EXACTLY: This is used by the parent to impose an exact size on the child. The child must use this size, and guarantee that all of its descendants will fit within this size.
- AT_MOST: This is used by the parent to impose a maximum size on the child. The child must gurantee that it and all of its descendants will fit within this size.

To intiate a layout, call `requestLayout()` (/reference/android/view/View.html#requestLayout()). This method is typically called by a view on itself when it believes that is can no longer fit within its current bounds.

Drawing

Drawing is handled by walking the tree and rendering each view that intersects the invalid region. Because the tree is traversed in-order, this means that parents will draw before (i.e., behind) their children, with siblings drawn in the order they appear in the tree. If you set a background drawable for a View, then the View will draw it for you before calling back to its `onDraw()` method.

Note that the framework will not draw views that are not in the invalid region.

To force a view to draw, call `invalidate()` (/reference/android/view/View.html#invalidate()).

Event Handling and Threading

The basic cycle of a view is as follows:

1. An event comes in and is dispatched to the appropriate view. The view handles the event and notifies any listeners.
2. If in the course of processing the event, the view's bounds may need to be changed, the view will call `requestLayout()`.
3. Similarly, if in the course of processing the event the view's appearance may need to be changed, the view will call `invalidate()`.
4. If either `requestLayout()` or `invalidate()` were called, the framework will take care of measuring, laying out, and drawing the tree as appropriate.

Note: The entire view tree is single threaded. You must always be on the UI thread when calling any method on any view. If you are doing work on other threads and want to update the state of a view from that thread, you should use a `Handler` (/reference/android/os/Handler.html).

Focus Handling

The framework will handle routine focus movement in response to user input. This includes changing the focus as views are removed or hidden, or as new views become available. Views indicate their willingness to take focus through the `isFocusable()` (/reference/android/view/View.html#isFocusable()) method. To change whether a view can take focus, call `setFocusable(boolean)` (/reference/android/view/View.html#setFocusable(boolean)). When in touch mode (see notes below) views indicate whether they still would like focus via `isFocusableInTouchMode()` (/reference/android/view/View.html#isFocusableInTouchMode()) and can change this via `setFocusableInTouchMode(boolean)` (/reference/android/view/View.html#setFocusableInTouchMode(boolean)).

Focus movement is based on an algorithm which finds the nearest neighbor in a given direction. In rare cases, the default algorithm may not match the intended behavior of the developer. In these situations, you can provide explicit overrides by using these XML attributes in the layout file:

```
nextFocusDown
nextFocusLeft
nextFocusRight
nextFocusUp
```

To get a particular view to take focus, call `requestFocus()` (/reference/android/view/View.html#requestFocus()).

Touch Mode

When a user is navigating a user interface via directional keys such as a D-pad, it is necessary to give focus to actionable items such as buttons so the user can see what will take input. If the device has touch capabilities, however, and the user begins interacting with the interface by touching it, it is no longer necessary to always highlight, or give focus to, a particular view. This motivates a mode for interaction named 'touch mode'.

For a touch capable device, once the user touches the screen, the device will enter touch mode. From this point onward, only views for which `isFocusableInTouchMode()` (/reference/android/view/View.html#isFocusableInTouchMode()) is true will be focusable, such as text editing widgets. Other views that are touchable, like buttons, will not take focus when touched; they will only fire the on click listeners.

Any time a user hits a directional key, such as a D-pad direction, the view device will exit touch mode, and find a view to take focus, so that the user may resume interacting with the user interface without touching the screen again.

The touch mode state is maintained across `Activity` (/reference/android/app/Activity.html)s. Call `isInTouchMode()` (/reference/android/view/View.html#isInTouchMode()) to see whether the device is currently in touch mode.

Scrolling

The framework provides basic support for views that wish to internally scroll their content. This includes keeping track of the X and Y scroll offset as well as mechanisms for drawing scrollbars. See `scrollBy(int, int)` (/reference/android/view/View.html#scrollBy(int, int)), `scrollTo(int, int)` (/reference/android/view/View.html#scrollTo(int, int)), and `awakenScrollBars()` (/reference/android/view/View.html#awakenScrollBars()) for more details.

Tags

Unlike IDs, tags are not used to identify views. Tags are essentially an extra piece of information that can be associated with a view. They are most often used as a convenience to store data related to views in the views themselves rather than by putting them in a separate structure.

Properties

The View class exposes an `ALPHA` (/reference/android/view/View.html#ALPHA) property, as well as several transform-related properties, such as `TRANSLATION_X` (/reference/android/view/View.html#TRANSLATION_X) and `TRANSLATION_Y` (/reference/android/view/View.html#TRANSLATION_Y). These properties are available both in the `Property` (/reference/android/util/Property.html) form as well as in similarly-named setter/getter methods (such as `setAlpha(float)` (/reference/android/view/View.html#setAlpha(float)) for `ALPHA` (/reference/android/view/View.html#ALPHA)). These properties can be used to set persistent state associated with these rendering-related properties on the view. The properties and methods can also be used in conjunction with `Animator` (/reference/android/animation/Animator.html)-based animations, described more in the Animation (#Animation) section.

Animation

Starting with Android 3.0, the preferred way of animating views is to use the `android.animation` (/reference/android/animation/package-summary.html) package APIs. These `Animator` (/reference/android/animation/Animator.html)-based classes change actual properties of the View object, such as `alpha` (/reference/android/view/View.html#setAlpha(float)) and `translationX` (/reference/android/view/View.html#setTranslationX(float)). This behavior is contrasted to that of the pre-3.0 `Animation` (/reference/android/view/animation/Animation.html)-based classes, which instead animate only how the view is drawn on the display. In particular, the `ViewPropertyAnimator` (/reference/android/view/ViewPropertyAnimator.html) class makes animating these View properties particularly easy and efficient.

Alternatively, you can use the pre-3.0 animation classes to animate how Views are rendered. You can attach an `Animation` (/reference/android/view/animation/Animation.html) object to a view using `setAnimation(Animation)` (/reference/android/view/View.html#setAnimation(android.view.animation.Animation)) or `startAnimation(Animation)` (/reference/android/view/View.html#startAnimation(android.view.animation.Animation)). The animation can alter the scale, rotation, translation and alpha of a view over time. If the animation is attached to a view that has children, the animation will affect the entire subtree rooted by that node. When an animation is started, the framework will take care of redrawing the appropriate views until the animation completes.

Security

Sometimes it is essential that an application be able to verify that an action is being performed with the full knowledge and consent of the user, such as granting a permission request, making a purchase or clicking on an advertisement. Unfortunately, a malicious application could try to spoof the user into performing these actions, unaware, by concealing the intended purpose of the view. As a remedy, the framework offers a touch filtering mechanism that can be used to improve the security of views that provide access to sensitive functionality.

To enable touch filtering, call `setFilterTouchesWhenObscured(boolean)` (/reference/android/view/View.html#setFilterTouchesWhenObscured(boolean)) or set the android:filterTouchesWhenObscured layout attribute to true. When enabled, the framework will discard touches that are received whenever the view's window is obscured by another visible window. As a result, the view will not receive touches whenever a toast, dialog or other window appears above the view's window.

For more fine-grained control over security, consider overriding the `onFilterTouchEventForSecurity(MotionEvent)` (/reference/android/view/View.html#onFilterTouchEventForSecurity(android.view.MotionEvent)) method to implement your own security policy. See also `FLAG_WINDOW_IS_OBSCURED` (/reference/android/view/MotionEvent.html#FLAG_WINDOW_IS_OBSCURED).

See Also
`ViewGroup`

Summary

Nested Classes

class View.AccessibilityDelegate	This class represents a delegate that can be registered in a `View` (/reference/android/view/View.html) to enhance accessibility support via composition rather via inheritance.
class View.BaseSavedState	Base class for derived classes that want to save and restore their own state in `onSaveInstanceState()`.
class View.DragShadowBuilder	Creates an image that the system displays during the drag and drop operation.
class View.MeasureSpec	A MeasureSpec encapsulates the layout requirements passed from parent to child.
interface View.OnAttachStateChangeListener	Interface definition for a callback to be invoked when this view is attached or detached from its window.
interface View.OnClickListener	Interface definition for a callback to be invoked when a view is clicked.
interface View.OnCreateContextMenuListener	Interface definition for a callback to be invoked when the context menu for this view is being built.
interface View.OnDragListener	Interface definition for a callback to be invoked when a drag is being dispatched to this view.
interface View.OnFocusChangeListener	Interface definition for a callback to be invoked when the focus state of a view changed.
interface View.OnGenericMotionListener	Interface definition for a callback to be invoked when a generic motion event is dispatched to this view.
interface View.OnHoverListener	Interface definition for a callback to be invoked when a hover event is dispatched to this view.
interface View.OnKeyListener	Interface definition for a callback to be invoked when a hardware key event is dispatched to this view.
interface View.OnLayoutChangeListener	Interface definition for a callback to be invoked when the layout bounds of a view changes due to layout processing.
interface View.OnLongClickListener	Interface definition for a callback to be invoked when a view has been clicked and held.
interface View.OnSystemUiVisibilityChangeListener	Interface definition for a callback to be invoked when the status bar changes visibility.
interface View.OnTouchListener	Interface definition for a callback to be invoked when a touch event is dispatched to this view.

XML Attributes

Attribute Name	*Related Method*	*Description*
android:accessibilityLiveRegion	setAccessibilityLiveRegion(int)	Indicates to accessibility services whether the user should be notified when this view changes.
android:alpha	setAlpha(float)	alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque).
android:background	setBackgroundResource(int)	A drawable to use as the background.
android:clickable	setClickable(boolean)	Defines whether this view reacts to click events.
android:contentDescription	setContentDescription(CharSequence)	Defines text that briefly describes content of the view.
android:drawingCacheQuality	setDrawingCacheQuality(int)	Defines the quality of translucent drawing caches.
android:duplicateParentState		When this attribute is set to true, the view gets its drawable state (focused, pressed, etc.) from its direct parent rather than from itself.
android:fadeScrollbars	setScrollbarFadingEnabled(boolean)	Defines whether to fade out scrollbars when they are not in use.
android:fadingEdgeLength	getVerticalFadingEdgeLength()	Defines the length of the fading edges.
android:filterTouchesWhenObscured	setFilterTouchesWhenObscured(boolean)	Specifies whether to filter touches when the view's window is obscured by another visible window.
android:fitsSystemWindows	setFitsSystemWindows(boolean)	Boolean internal attribute to adjust view layout based on system windows such as the status bar.
android:focusable	setFocusable(boolean)	Boolean that controls whether a view can take focus.
android:focusableInTouchMode	setFocusableInTouchMode(boolean)	Boolean that controls whether a view can take focus while in touch mode.
android:hapticFeedbackEnabled	setHapticFeedbackEnabled(boolean)	Boolean that controls whether a view should have haptic feedback enabled for events such as long presses.
android:id	setId(int)	Supply an identifier name for this view, to later retrieve it with `View.findViewById()` or `Activity.findViewById()`.
android:importantForAccessibility	setImportantForAccessibility(int)	Controls how this View is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen.
android:isScrollContainer	setScrollContainer(boolean)	Set this if the view will serve as a scrolling container, meaing that it can be resized to shrink its overall window so that there will be space for an input method.
android:keepScreenOn	setKeepScreenOn(boolean)	Controls whether the view's window should keep the screen on while visible.
android:layerType	setLayerType(int,Paint)	Specifies the type of layer backing this view.
android:layoutDirection	setLayoutDirection(int)	Defines the direction of layout drawing.
android:longClickable	setLongClickable(boolean)	Defines whether this view reacts to long click events.
android:minHeight	setMinimumHeight(int)	Defines the minimum height of the view.
android:minWidth	setMinimumWidth(int)	Defines the minimum width of the view.

android:nextFocusDown	setNextFocusDownId(int)	Defines the next view to give focus to when the next focus is `FOCUS_DOWN` If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` will result when the reference is accessed.
android:nextFocusForward	setNextFocusForwardId(int)	Defines the next view to give focus to when the next focus is `FOCUS_FORWARD` If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` will result when the reference is accessed.
android:nextFocusLeft	setNextFocusLeftId(int)	Defines the next view to give focus to when the next focus is `FOCUS_LEFT`.
android:nextFocusRight	setNextFocusRightId(int)	Defines the next view to give focus to when the next focus is `FOCUS_RIGHT` If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` will result when the reference is accessed.
android:nextFocusUp	setNextFocusUpId(int)	Defines the next view to give focus to when the next focus is `FOCUS_UP` If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` will result when the reference is accessed.
android:onClick		Name of the method in this View's context to invoke when the view is clicked.
android:padding	setPaddingRelative(int,int,int,int)	Sets the padding, in pixels, of all four edges.
android:paddingBottom	setPaddingRelative(int,int,int,int)	Sets the padding, in pixels, of the bottom edge; see `padding`.
android:paddingEnd	setPaddingRelative(int,int,int,int)	Sets the padding, in pixels, of the end edge; see `padding`.
android:paddingLeft	setPadding(int,int,int,int)	Sets the padding, in pixels, of the left edge; see `padding`.
android:paddingRight	setPadding(int,int,int,int)	Sets the padding, in pixels, of the right edge; see `padding`.
android:paddingStart	setPaddingRelative(int,int,int,int)	Sets the padding, in pixels, of the start edge; see `padding`.
android:paddingTop	setPaddingRelative(int,int,int,int)	Sets the padding, in pixels, of the top edge; see `padding`.
android:requiresFadingEdge	setVerticalFadingEdgeEnabled(boolean)	Defines which edges should be faded on scrolling.
android:rotation	setRotation(float)	rotation of the view, in degrees.
android:rotationX	setRotationX(float)	rotation of the view around the x axis, in degrees.
android:rotationY	setRotationY(float)	rotation of the view around the y axis, in degrees.
android:saveEnabled	setSaveEnabled(boolean)	If unset, no state will be saved for this view when it is being frozen.
android:scaleX	setScaleX(float)	scale of the view in the x direction.
android:scaleY	setScaleY(float)	scale of the view in the y direction.
android:scrollX		The initial horizontal scroll offset, in pixels.
android:scrollY		The initial vertical scroll offset, in pixels.
android:scrollbarAlwaysDrawHorizontalTrack		Defines whether the horizontal scrollbar track should always be drawn.
android:scrollbarAlwaysDrawVerticalTrack		Defines whether the vertical scrollbar track should always be drawn.
android:scrollbarDefaultDelayBeforeFade	setScrollBarDefaultDelayBeforeFade(int)	Defines the delay in milliseconds that a scrollbar waits before fade out.
android:scrollbarFadeDuration	setScrollBarFadeDuration(int)	Defines the delay in milliseconds that a scrollbar takes to fade out.
android:scrollbarSize	setScrollBarSize(int)	Sets the width of vertical scrollbars and height of horizontal scrollbars.
android:scrollbarStyle	setScrollBarStyle(int)	Controls the scrollbar style and position.
android:scrollbarThumbHorizontal		Defines the horizontal scrollbar thumb drawable.
android:scrollbarThumbVertical		Defines the vertical scrollbar thumb drawable.
android:scrollbarTrackHorizontal		Defines the horizontal scrollbar track drawable.
android:scrollbarTrackVertical		Defines the vertical scrollbar track drawable.
android:scrollbars		Defines which scrollbars should be displayed on scrolling or not.
android:soundEffectsEnabled	setSoundEffectsEnabled(boolean)	Boolean that controls whether a view should have sound effects enabled for events such as clicking and touching.
android:tag		Supply a tag for this view containing a String, to be retrieved later with `View.getTag()` or searched for with `View.findViewWithTag()`.
android:textAlignment	setTextAlignment(int)	Defines the alignment of the text.
android:textDirection	setTextDirection(int)	Defines the direction of the text.
android:transformPivotX	setPivotX(float)	x location of the pivot point around which the view will rotate and scale.
android:transformPivotY	setPivotY(float)	y location of the pivot point around which the view will rotate and scale.
android:translationX	setTranslationX(float)	translation in x of the view.
android:translationY	setTranslationY(float)	translation in y of the view.
android:visibility	setVisibility(int)	Controls the initial visibility of the view.

Constants

int ACCESSIBILITY_LIVE_REGION_ASSERTIVE	Live region mode specifying that accessibility services should interrupt ongoing speech to immediately announce changes to this view.
int ACCESSIBILITY_LIVE_REGION_NONE	Live region mode specifying that accessibility services should not automatically announce changes to this view.
int ACCESSIBILITY_LIVE_REGION_POLITE	Live region mode specifying that accessibility services should announce changes to this view.
int DRAWING_CACHE_QUALITY_AUTO	Enables automatic quality mode for the drawing cache.

int DRAWING_CACHE_QUALITY_HIGH	Enables high quality mode for the drawing cache.
int DRAWING_CACHE_QUALITY_LOW	Enables low quality mode for the drawing cache.
int FIND_VIEWS_WITH_CONTENT_DESCRIPTION	Find find views that contain the specified content description.
int FIND_VIEWS_WITH_TEXT	Find views that render the specified text.
int FOCUSABLES_ALL	View flag indicating whether `addFocusables(ArrayList, int, int)` should add all focusable Views regardless if they are focusable in touch mode.
int FOCUSABLES_TOUCH_MODE	View flag indicating whether `addFocusables(ArrayList, int, int)` should add only Views focusable in touch mode.
int FOCUS_BACKWARD	Use with `focusSearch(int)`.
int FOCUS_DOWN	Use with `focusSearch(int)`.
int FOCUS_FORWARD	Use with `focusSearch(int)`.
int FOCUS_LEFT	Use with `focusSearch(int)`.
int FOCUS_RIGHT	Use with `focusSearch(int)`.
int FOCUS_UP	Use with `focusSearch(int)`.
int GONE	This view is invisible, and it doesn't take any space for layout purposes.
int HAPTIC_FEEDBACK_ENABLED	View flag indicating whether this view should have haptic feedback enabled for events such as long presses.
int IMPORTANT_FOR_ACCESSIBILITY_AUTO	Automatically determine whether a view is important for accessibility.
int IMPORTANT_FOR_ACCESSIBILITY_NO	The view is not important for accessibility.
int IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS	The view is not important for accessibility, nor are any of its descendant views.
int IMPORTANT_FOR_ACCESSIBILITY_YES	The view is important for accessibility.
int INVISIBLE	This view is invisible, but it still takes up space for layout purposes.
int KEEP_SCREEN_ON	View flag indicating that the screen should remain on while the window containing this view is visible to the user.
int LAYER_TYPE_HARDWARE	Indicates that the view has a hardware layer.
int LAYER_TYPE_NONE	Indicates that the view does not have a layer.
int LAYER_TYPE_SOFTWARE	Indicates that the view has a software layer.
int LAYOUT_DIRECTION_INHERIT	Horizontal layout direction of this view is inherited from its parent.
int LAYOUT_DIRECTION_LOCALE	Horizontal layout direction of this view is from deduced from the default language script for the locale.
int LAYOUT_DIRECTION_LTR	Horizontal layout direction of this view is from Left to Right.
int LAYOUT_DIRECTION_RTL	Horizontal layout direction of this view is from Right to Left.
int MEASURED_HEIGHT_STATE_SHIFT	Bit shift of `MEASURED_STATE_MASK` to get to the height bits for functions that combine both width and height into a single int, such as `getMeasuredState()` and the childState argument of `resolveSizeAndState(int, int, int)`.
int MEASURED_SIZE_MASK	Bits of `getMeasuredWidthAndState()` and `getMeasuredWidthAndState()` that provide the actual measured size.
int MEASURED_STATE_MASK	Bits of `getMeasuredWidthAndState()` and `getMeasuredWidthAndState()` that provide the additional state bits.
int MEASURED_STATE_TOO_SMALL	Bit of `getMeasuredWidthAndState()` and `getMeasuredWidthAndState()` that indicates the measured size is smaller that the space the view would like to have.
int NO_ID	Used to mark a View that has no ID.
int OVER_SCROLL_ALWAYS	Always allow a user to over-scroll this view, provided it is a view that can scroll.
int OVER_SCROLL_IF_CONTENT_SCROLLS	Allow a user to over-scroll this view only if the content is large enough to meaningfully scroll, provided it is a view that can scroll.
int OVER_SCROLL_NEVER	Never allow a user to over-scroll this view.
int SCREEN_STATE_OFF	Indicates that the screen has changed state and is now off.
int SCREEN_STATE_ON	Indicates that the screen has changed state and is now on.
int SCROLLBARS_INSIDE_INSET	The scrollbar style to display the scrollbars inside the padded area, increasing the padding of the view.
int SCROLLBARS_INSIDE_OVERLAY	The scrollbar style to display the scrollbars inside the content area, without increasing the padding.
int SCROLLBARS_OUTSIDE_INSET	The scrollbar style to display the scrollbars at the edge of the view, increasing the padding of the view.
int SCROLLBARS_OUTSIDE_OVERLAY	The scrollbar style to display the scrollbars at the edge of the view, without increasing the padding.
int SCROLLBAR_POSITION_DEFAULT	Position the scroll bar at the default position as determined by the system.
int SCROLLBAR_POSITION_LEFT	Position the scroll bar along the left edge.
int SCROLLBAR_POSITION_RIGHT	Position the scroll bar along the right edge.
int SOUND_EFFECTS_ENABLED	View flag indicating whether this view should have sound effects enabled for events such as clicking and touching.
int STATUS_BAR_HIDDEN	*This constant was deprecated in API level 14. Use `SYSTEM_UI_FLAG_LOW_PROFILE` instead.*
int STATUS_BAR_VISIBLE	*This constant was deprecated in API level 14. Use `SYSTEM_UI_FLAG_VISIBLE` instead.*
int SYSTEM_UI_FLAG_FULLSCREEN	Flag for `setSystemUiVisibility(int)`: View has requested to go into the normal fullscreen mode so that its content can take over the screen while still allowing the user to interact with the application.
int SYSTEM_UI_FLAG_HIDE_NAVIGATION	Flag for `setSystemUiVisibility(int)`: View has requested that the system navigation be temporarily hidden.
int SYSTEM_UI_FLAG_IMMERSIVE	Flag for `setSystemUiVisibility(int)`: View would like to remain interactive when hiding the navigation bar with `SYSTEM_UI_FLAG_HIDE_NAVIGATION`.

int SYSTEM_UI_FLAG_IMMERSIVE_STICKY	Flag for `setSystemUiVisibility(int)`: View would like to remain interactive when hiding the status bar with `SYSTEM_UI_FLAG_FULLSCREEN` and/or hiding the navigation bar with `SYSTEM_UI_FLAG_HIDE_NAVIGATION`.
int SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN	Flag for `setSystemUiVisibility(int)`: View would like its window to be layed out as if it has requested `SYSTEM_UI_FLAG_FULLSCREEN`, even if it currently hasn't.
int SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION	Flag for `setSystemUiVisibility(int)`: View would like its window to be layed out as if it has requested `SYSTEM_UI_FLAG_HIDE_NAVIGATION`, even if it currently hasn't.
int SYSTEM_UI_FLAG_LAYOUT_STABLE	Flag for `setSystemUiVisibility(int)`: When using other layout flags, we would like a stable view of the content insets given to `fitSystemWindows(Rect)`.
int SYSTEM_UI_FLAG_LOW_PROFILE	Flag for `setSystemUiVisibility(int)`: View has requested the system UI to enter an unobtrusive "low profile" mode.
int SYSTEM_UI_FLAG_VISIBLE	Special constant for `setSystemUiVisibility(int)`: View has requested the system UI (status bar) to be visible (the default).
int SYSTEM_UI_LAYOUT_FLAGS	Flags that can impact the layout in relation to system UI.
int TEXT_ALIGNMENT_CENTER	Center the paragraph, e.g.
int TEXT_ALIGNMENT_GRAVITY	Default for the root view.
int TEXT_ALIGNMENT_INHERIT	
int TEXT_ALIGNMENT_TEXT_END	Align to the end of the paragraph, e.g.
int TEXT_ALIGNMENT_TEXT_START	Align to the start of the paragraph, e.g.
int TEXT_ALIGNMENT_VIEW_END	Align to the end of the view, which is ALIGN_RIGHT if the view's resolved layoutDirection is LTR, and ALIGN_LEFT otherwise.
int TEXT_ALIGNMENT_VIEW_START	Align to the start of the view, which is ALIGN_LEFT if the view's resolved layoutDirection is LTR, and ALIGN_RIGHT otherwise.
int TEXT_DIRECTION_ANY_RTL	Text direction is using "any-RTL" algorithm.
int TEXT_DIRECTION_FIRST_STRONG	Text direction is using "first strong algorithm".
int TEXT_DIRECTION_INHERIT	Text direction is inherited thru `ViewGroup`
int TEXT_DIRECTION_LOCALE	Text direction is coming from the system Locale.
int TEXT_DIRECTION_LTR	Text direction is forced to LTR.
int TEXT_DIRECTION_RTL	Text direction is forced to RTL.
String VIEW_LOG_TAG	The logging tag used by this class with android.util.Log.
int VISIBLE	This view is visible.

Fields

public static final Property<View, Float> ALPHA	A Property wrapper around the `alpha` functionality handled by the `setAlpha(float)` and `getAlpha()` methods.
protected static final int[] EMPTY_STATE_SET	Indicates the view has no states set.
protected static final int[] ENABLED_FOCUSED_SELECTED_STATE_SET	Indicates the view is enabled, focused and selected.
protected static final int[] ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is enabled, focused, selected and its window has the focus.
protected static final int[] ENABLED_FOCUSED_STATE_SET	Indicates the view is enabled and has the focus.
protected static final int[] ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET	Indicates the view is enabled, focused and its window has the focus.
protected static final int[] ENABLED_SELECTED_STATE_SET	Indicates the view is enabled and selected.
protected static final int[] ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is enabled, selected and its window has the focus.
protected static final int[] ENABLED_STATE_SET	Indicates the view is enabled.
protected static final int[] ENABLED_WINDOW_FOCUSED_STATE_SET	Indicates the view is enabled and that its window has focus.
protected static final int[] FOCUSED_SELECTED_STATE_SET	Indicates the view is focused and selected.
protected static final int[] FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is focused, selected and its window has the focus.
protected static final int[] FOCUSED_STATE_SET	Indicates the view is focused.
protected static final int[] FOCUSED_WINDOW_FOCUSED_STATE_SET	Indicates the view has the focus and that its window has the focus.
protected static final int[] PRESSED_ENABLED_FOCUSED_SELECTED_STATE_SET	Indicates the view is pressed, enabled, focused and selected.
protected static final int[] PRESSED_ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed, enabled, focused, selected and its window has the focus.
protected static final int[] PRESSED_ENABLED_FOCUSED_STATE_SET	Indicates the view is pressed, enabled and focused.
protected static final int[] PRESSED_ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed, enabled, focused and its window has the focus.
protected static final int[] PRESSED_ENABLED_SELECTED_STATE_SET	Indicates the view is pressed, enabled and selected.

protected static final int[] PRESSED_ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed, enabled, selected and its window has the focus.
protected static final int[] PRESSED_ENABLED_STATE_SET	Indicates the view is pressed and enabled.
protected static final int[] PRESSED_ENABLED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed, enabled and its window has the focus.
protected static final int[] PRESSED_FOCUSED_SELECTED_STATE_SET	Indicates the view is pressed, focused and selected.
protected static final int[] PRESSED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed, focused, selected and its window has the focus.
protected static final int[] PRESSED_FOCUSED_STATE_SET	Indicates the view is pressed and focused.
protected static final int[] PRESSED_FOCUSED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed, focused and its window has the focus.
protected static final int[] PRESSED_SELECTED_STATE_SET	Indicates the view is pressed and selected.
protected static final int[] PRESSED_SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed, selected and its window has the focus.
protected static final int[] PRESSED_STATE_SET	Indicates the view is pressed.
protected static final int[] PRESSED_WINDOW_FOCUSED_STATE_SET	Indicates the view is pressed and its window has the focus.
public static final Property<View, Float> ROTATION	A Property wrapper around the `rotation` functionality handled by the `setRotation(float)` and `getRotation()` methods.
public static final Property<View, Float> ROTATION_X	A Property wrapper around the `rotationX` functionality handled by the `setRotationX(float)` and `getRotationX()` methods.
public static final Property<View, Float> ROTATION_Y	A Property wrapper around the `rotationY` functionality handled by the `setRotationY(float)` and `getRotationY()` methods.
public static final Property<View, Float> SCALE_X	A Property wrapper around the `scaleX` functionality handled by the `setScaleX(float)` and `getScaleX()` methods.
public static final Property<View, Float> SCALE_Y	A Property wrapper around the `scaleY` functionality handled by the `setScaleY(float)` and `getScaleY()` methods.
protected static final int[] SELECTED_STATE_SET	Indicates the view is selected.
protected static final int[] SELECTED_WINDOW_FOCUSED_STATE_SET	Indicates the view is selected and that its window has the focus.
public static final Property<View, Float> TRANSLATION_X	A Property wrapper around the `translationX` functionality handled by the `setTranslationX(float)` and `getTranslationX()` methods.
public static final Property<View, Float> TRANSLATION_Y	A Property wrapper around the `translationY` functionality handled by the `setTranslationY(float)` and `getTranslationY()` methods.
protected static final int[] WINDOW_FOCUSED_STATE_SET	Indicates the view's window has focus.
public static final Property<View, Float> X	A Property wrapper around the `x` functionality handled by the `setX(float)` and `getX()` methods.
public static final Property<View, Float> Y	A Property wrapper around the `y` functionality handled by the `setY(float)` and `getY()` methods.

Public Constructors

View (Context context)
Simple constructor to use when creating a view from code.
View (Context context, AttributeSet attrs)
Constructor that is called when inflating a view from XML.
View (Context context, AttributeSet attrs, int defStyleAttr)
Perform inflation from XML and apply a class-specific base style.

Public Methods	
void	addChildrenForAccessibility (ArrayList<View> children) Adds the children of a given View for accessibility.
void	addFocusables (ArrayList<View> views, int direction, int focusableMode) Adds any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views.
void	addFocusables (ArrayList<View> views, int direction) Add any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views.
void	addOnAttachStateChangeListener (View.OnAttachStateChangeListener listener) Add a listener for attach state changes.
void	addOnLayoutChangeListener (View.OnLayoutChangeListener listener) Add a listener that will be called when the bounds of the view change due to layout processing.
void	addTouchables (ArrayList<View> views) Add any touchable views that are descendants of this view (possibly including this view if it is touchable itself) to views.
ViewPropertyAnimator	animate () This method returns a ViewPropertyAnimator object, which can be used to animate specific properties on this View.
void	announceForAccessibility (CharSequence text) Convenience method for sending a `TYPE_ANNOUNCEMENT` `AccessibilityEvent` to make an announcement which is related to some sort of a context change for which none of the events representing UI transitions is a good fit.
void	bringToFront () Change the view's z order in the tree, so it's on top of other sibling views.
void	buildDrawingCache () Calling this method is equivalent to calling `buildDrawingCache(false)`.
void	buildDrawingCache (boolean autoScale) Forces the drawing cache to be built if the drawing cache is invalid.
void	buildLayer () Forces this view's layer to be created and this view to be rendered into its layer.
boolean	callOnClick () Directly call any attached OnClickListener.
boolean	canResolveLayoutDirection () Check if layout direction resolution can be done.
boolean	canResolveTextAlignment () Check if text alignment resolution can be done.
boolean	canResolveTextDirection () Check if text direction resolution can be done.
boolean	canScrollHorizontally (int direction) Check if this view can be scrolled horizontally in a certain direction.
boolean	canScrollVertically (int direction) Check if this view can be scrolled vertically in a certain direction.
void	cancelLongPress () Cancels a pending long press.
final void	cancelPendingInputEvents () Cancel any deferred high-level input events that were previously posted to the event queue.
boolean	checkInputConnectionProxy (View view) Called by the `InputMethodManager` when a view who is not the current input connection target is trying to make a call on the manager.
void	clearAnimation () Cancels any animations for this view.
void	clearFocus () Called when this view wants to give up focus.
static int	combineMeasuredStates (int curState, int newState) Merge two states as returned by `getMeasuredState()`.
void	computeScroll () Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary.
AccessibilityNodeInfo	createAccessibilityNodeInfo () Returns an `AccessibilityNodeInfo` representing this view from the point of view of an `AccessibilityService`.
void	createContextMenu (ContextMenu menu) Show the context menu for this view.
void	destroyDrawingCache () Frees the resources used by the drawing cache.
void	dispatchConfigurationChanged (Configuration newConfig) Dispatch a notification about a resource configuration change down the view hierarchy.
void	dispatchDisplayHint (int hint) Dispatch a hint about whether this view is displayed.
boolean	dispatchDragEvent (DragEvent event) Detects if this View is enabled and has a drag event listener.
boolean	dispatchGenericMotionEvent (MotionEvent event) Dispatch a generic motion event.
boolean	dispatchKeyEvent (KeyEvent event) Dispatch a key event to the next view on the focus path.
boolean	dispatchKeyEventPreIme (KeyEvent event) Dispatch a key event before it is processed by any input method associated with the view hierarchy.
boolean	dispatchKeyShortcutEvent (KeyEvent event) Dispatches a key shortcut event.

boolean	dispatchPopulateAccessibilityEvent (AccessibilityEvent event) Dispatches an `AccessibilityEvent` to the `View` first and then to its children for adding their text content to the event.
void	dispatchSystemUiVisibilityChanged (int visibility) Dispatch callbacks to `setOnSystemUiVisibilityChangeListener(View.OnSystemUiVisibilityChangeListener)` down the view hierarchy.
boolean	dispatchTouchEvent (MotionEvent event) Pass the touch screen motion event down to the target view, or this view if it is the target.
boolean	dispatchTrackballEvent (MotionEvent event) Pass a trackball motion event down to the focused view.
boolean	dispatchUnhandledMove (View focused, int direction) This method is the last chance for the focused view and its ancestors to respond to an arrow key.
void	dispatchWindowFocusChanged (boolean hasFocus) Called when the window containing this view gains or loses window focus.
void	dispatchWindowSystemUiVisiblityChanged (int visible) Dispatch callbacks to `onWindowSystemUiVisibilityChanged(int)` down the view hierarchy.
void	dispatchWindowVisibilityChanged (int visibility) Dispatch a window visibility change down the view hierarchy.
void	draw (Canvas canvas) Manually render this view (and all of its children) to the given Canvas.
View	findFocus () Find the view in the hierarchy rooted at this view that currently has focus.
final View	findViewById (int id) Look for a child view with the given id.
final View	findViewWithTag (Object tag) Look for a child view with the given tag.
void	findViewsWithText (ArrayList<View> outViews, CharSequence searched, int flags) Finds the Views that contain given text.
View	focusSearch (int direction) Find the nearest view in the specified direction that can take focus.
void	forceLayout () Forces this view to be laid out during the next layout pass.
static int	generateViewId () Generate a value suitable for use in `setId(int)`.
int	getAccessibilityLiveRegion () Gets the live region mode for this View.
AccessibilityNodeProvider	getAccessibilityNodeProvider () Gets the provider for managing a virtual view hierarchy rooted at this View and reported to `AccessibilityServices` that explore the window content.
float	getAlpha () The opacity of the view.
Animation	getAnimation () Get the animation currently associated with this view.
IBinder	getApplicationWindowToken () Retrieve a unique token identifying the top-level "real" window of the window that this view is attached to.
Drawable	getBackground () Gets the background drawable
int	getBaseline () Return the offset of the widget's text baseline from the widget's top boundary.
final int	getBottom () Bottom position of this view relative to its parent.
float	getCameraDistance () Gets the distance along the Z axis from the camera to this view.
Rect	getClipBounds () Returns a copy of the current `clipBounds`.
CharSequence	getContentDescription () Gets the `View` description.
final Context	getContext () Returns the context the view is running in, through which it can access the current theme, resources, etc.
static int	getDefaultSize (int size, int measureSpec) Utility to return a default size.
Display	getDisplay () Gets the logical display to which the view's window has been attached.
final int[]	getDrawableState () Return an array of resource IDs of the drawable states representing the current state of the view.
Bitmap	getDrawingCache (boolean autoScale) Returns the bitmap in which this view drawing is cached.
Bitmap	getDrawingCache () Calling this method is equivalent to calling `getDrawingCache(false)`.
int	getDrawingCacheBackgroundColor ()
int	getDrawingCacheQuality () Returns the quality of the drawing cache.
void	getDrawingRect (Rect outRect) Return the visible drawing bounds of your view.
long	getDrawingTime () Return the time at which the drawing of the view hierarchy started.

boolean	getFilterTouchesWhenObscured () Gets whether the framework should discard touches when the view's window is obscured by another visible window.
boolean	getFitsSystemWindows () Check for state of `setFitsSystemWindows(boolean)`.
ArrayList<View>	getFocusables (int direction) Find and return all focusable views that are descendants of this view, possibly including this view if it is focusable itself.
void	getFocusedRect (Rect r) When a view has focus and the user navigates away from it, the next view is searched for starting from the rectangle filled in by this method.
boolean	getGlobalVisibleRect (Rect r, Point globalOffset) If some part of this view is not clipped by any of its parents, then return that area in r in global (root) coordinates.
final boolean	getGlobalVisibleRect (Rect r)
Handler	getHandler ()
final int	getHeight () Return the height of your view.
void	getHitRect (Rect outRect) Hit rectangle in parent's coordinates
int	getHorizontalFadingEdgeLength () Returns the size of the horizontal faded edges used to indicate that more content in this view is visible.
int	getId () Returns this view's identifier.
int	getImportantForAccessibility () Gets the mode for determining whether this View is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen.
boolean	getKeepScreenOn () Returns whether the screen should remain on, corresponding to the current value of `KEEP_SCREEN_ON`.
KeyEvent.DispatcherState	getKeyDispatcherState () Return the global `KeyEvent.DispatcherState` for this view's window.
int	getLabelFor () Gets the id of a view for which this view serves as a label for accessibility purposes.
int	getLayerType () Indicates what type of layer is currently associated with this view.
int	getLayoutDirection () Returns the resolved layout direction for this view.
ViewGroup.LayoutParams	getLayoutParams () Get the LayoutParams associated with this view.
final int	getLeft () Left position of this view relative to its parent.
final boolean	getLocalVisibleRect (Rect r)
void	getLocationInWindow (int[] location) Computes the coordinates of this view in its window.
void	getLocationOnScreen (int[] location) Computes the coordinates of this view on the screen.
Matrix	getMatrix () The transform matrix of this view, which is calculated based on the current roation, scale, and pivot properties.
final int	getMeasuredHeight () Like `getMeasuredHeightAndState()`, but only returns the raw width component (that is the result is masked by `MEASURED_SIZE_MASK`).
final int	getMeasuredHeightAndState () Return the full height measurement information for this view as computed by the most recent call to `measure(int, int)`.
final int	getMeasuredState () Return only the state bits of `getMeasuredWidthAndState()` and `getMeasuredHeightAndState()`, combined into one integer.
final int	getMeasuredWidth () Like `getMeasuredWidthAndState()`, but only returns the raw width component (that is the result is masked by `MEASURED_SIZE_MASK`).
final int	getMeasuredWidthAndState () Return the full width measurement information for this view as computed by the most recent call to `measure(int, int)`.
int	getMinimumHeight () Returns the minimum height of the view.
int	getMinimumWidth () Returns the minimum width of the view.
int	getNextFocusDownId () Gets the id of the view to use when the next focus is `FOCUS_DOWN`.
int	getNextFocusForwardId () Gets the id of the view to use when the next focus is `FOCUS_FORWARD`.
int	getNextFocusLeftId () Gets the id of the view to use when the next focus is `FOCUS_LEFT`.
int	getNextFocusRightId () Gets the id of the view to use when the next focus is `FOCUS_RIGHT`.
int	getNextFocusUpId () Gets the id of the view to use when the next focus is `FOCUS_UP`.
View.OnFocusChangeListener	getOnFocusChangeListener () Returns the focus-change callback registered for this view.

int	getOverScrollMode () Returns the over-scroll mode for this view.
ViewOverlay	getOverlay () Returns the overlay for this view, creating it if it does not yet exist.
int	getPaddingBottom () Returns the bottom padding of this view.
int	getPaddingEnd () Returns the end padding of this view depending on its resolved layout direction.
int	getPaddingLeft () Returns the left padding of this view.
int	getPaddingRight () Returns the right padding of this view.
int	getPaddingStart () Returns the start padding of this view depending on its resolved layout direction.
int	getPaddingTop () Returns the top padding of this view.
final ViewParent	getParent () Gets the parent of this view.
ViewParent	getParentForAccessibility () Gets the parent for accessibility purposes.
float	getPivotX () The x location of the point around which the view is `rotated` and `scaled`.
float	getPivotY () The y location of the point around which the view is `rotated` and `scaled`.
Resources	getResources () Returns the resources associated with this view.
final int	getRight () Right position of this view relative to its parent.
View	getRootView () Finds the topmost view in the current view hierarchy.
float	getRotation () The degrees that the view is rotated around the pivot point.
float	getRotationX () The degrees that the view is rotated around the horizontal axis through the pivot point.
float	getRotationY () The degrees that the view is rotated around the vertical axis through the pivot point.
float	getScaleX () The amount that the view is scaled in x around the pivot point, as a proportion of the view's unscaled width.
float	getScaleY () The amount that the view is scaled in y around the pivot point, as a proportion of the view's unscaled height.
int	getScrollBarDefaultDelayBeforeFade () Returns the delay before scrollbars fade.
int	getScrollBarFadeDuration () Returns the scrollbar fade duration.
int	getScrollBarSize () Returns the scrollbar size.
int	getScrollBarStyle () Returns the current scrollbar style.
final int	getScrollX () Return the scrolled left position of this view.
final int	getScrollY () Return the scrolled top position of this view.
int	getSolidColor () Override this if your view is known to always be drawn on top of a solid color background, and needs to draw fading edges.
int	getSystemUiVisibility () Returns the last `setSystemUiVisibility(int)` that this view has requested.
Object	getTag (int key) Returns the tag associated with this view and the specified key.
Object	getTag () Returns this view's tag.
int	getTextAlignment () Return the resolved text alignment.
int	getTextDirection () Return the resolved text direction.
final int	getTop () Top position of this view relative to its parent.
TouchDelegate	getTouchDelegate () Gets the TouchDelegate for this View.
ArrayList<View>	getTouchables () Find and return all touchable views that are descendants of this view, possibly including this view if it is touchable itself.
float	getTranslationX () The horizontal location of this view relative to its `left` position.
float	getTranslationY () The vertical location of this view relative to its `top` position.
int	getVerticalFadingEdgeLength () Returns the size of the vertical faded edges used to indicate that more content in this view is visible.
int	getVerticalScrollbarPosition ()

int	getVerticalScrollbarWidth () Returns the width of the vertical scrollbar.
ViewTreeObserver	getViewTreeObserver () Returns the ViewTreeObserver for this view's hierarchy.
int	getVisibility () Returns the visibility status for this view.
final int	getWidth () Return the width of the your view.
WindowId	getWindowId () Retrieve the `WindowId` for the window this view is currently attached to.
int	getWindowSystemUiVisibility () Returns the current system UI visibility that is currently set for the entire window.
IBinder	getWindowToken () Retrieve a unique token identifying the window this view is attached to.
int	getWindowVisibility () Returns the current visibility of the window this view is attached to (either `GONE`, `INVISIBLE`, or `VISIBLE`).
void	getWindowVisibleDisplayFrame (Rect outRect) Retrieve the overall visible display size in which the window this view is attached to has been positioned in.
float	getX () The visual x position of this view, in pixels.
float	getY () The visual y position of this view, in pixels.
boolean	hasFocus () Returns true if this view has focus iteself, or is the ancestor of the view that has focus.
boolean	hasFocusable () Returns true if this view is focusable or if it contains a reachable View for which `hasFocusable()` returns true.
boolean	hasOnClickListeners () Return whether this view has an attached OnClickListener.
boolean	hasOverlappingRendering () Returns whether this View has content which overlaps.
boolean	hasTransientState () Indicates whether the view is currently tracking transient state that the app should not need to concern itself with saving and restoring, but that the framework should take special note to preserve when possible.
boolean	hasWindowFocus () Returns true if this view is in a window that currently has window focus.
static View	inflate (Context context, int resource, ViewGroup root) Inflate a view from an XML resource.
void	invalidate (Rect dirty) Mark the area defined by dirty as needing to be drawn.
void	invalidate (int l, int t, int r, int b) Mark the area defined by the rect (l,t,r,b) as needing to be drawn.
void	invalidate () Invalidate the whole view.
void	invalidateDrawable (Drawable drawable) Invalidates the specified Drawable.
boolean	isActivated () Indicates the activation state of this view.
boolean	isAttachedToWindow () Returns true if this view is currently attached to a window.
boolean	isClickable () Indicates whether this view reacts to click events or not.
boolean	isDirty () True if this view has changed since the last time being drawn.
boolean	isDrawingCacheEnabled () Indicates whether the drawing cache is enabled for this view.
boolean	isDuplicateParentStateEnabled () Indicates whether this duplicates its drawable state from its parent.
boolean	isEnabled () Returns the enabled status for this view.
final boolean	isFocusable () Returns whether this View is able to take focus.
final boolean	isFocusableInTouchMode () When a view is focusable, it may not want to take focus when in touch mode.
boolean	isFocused () Returns true if this view has focus
boolean	isHapticFeedbackEnabled ()
boolean	isHardwareAccelerated () Indicates whether this view is attached to a hardware accelerated window or not.
boolean	isHorizontalFadingEdgeEnabled () Indicate whether the horizontal edges are faded when the view is scrolled horizontally.
boolean	isHorizontalScrollBarEnabled () Indicate whether the horizontal scrollbar should be drawn or not.
boolean	isHovered () Returns true if the view is currently hovered.

boolean	isInEditMode () Indicates whether this View is currently in edit mode.
boolean	isInLayout () Returns whether the view hierarchy is currently undergoing a layout pass.
boolean	isInTouchMode () Returns whether the device is currently in touch mode.
boolean	isLaidOut () Returns true if this view has been through at least one layout since it was last attached to or detached from a window.
boolean	isLayoutDirectionResolved ()
boolean	isLayoutRequested () Indicates whether or not this view's layout will be requested during the next hierarchy layout pass.
boolean	isLongClickable () Indicates whether this view reacts to long click events or not.
boolean	isOpaque () Indicates whether this View is opaque.
boolean	isPaddingRelative () Return if the padding as been set thru relative values `setPaddingRelative(int, int, int, int)` or thru
boolean	isPressed () Indicates whether the view is currently in pressed state.
boolean	isSaveEnabled () Indicates whether this view will save its state (that is, whether its `onSaveInstanceState()` method will be called).
boolean	isSaveFromParentEnabled () Indicates whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent.
boolean	isScrollContainer () Indicates whether this view is one of the set of scrollable containers in its window.
boolean	isScrollbarFadingEnabled () Returns true if scrollbars will fade when this view is not scrolling
boolean	isSelected () Indicates the selection state of this view.
boolean	isShown () Returns the visibility of this view and all of its ancestors
boolean	isSoundEffectsEnabled ()
boolean	isTextAlignmentResolved ()
boolean	isTextDirectionResolved ()
boolean	isVerticalFadingEdgeEnabled () Indicate whether the vertical edges are faded when the view is scrolled horizontally.
boolean	isVerticalScrollBarEnabled () Indicate whether the vertical scrollbar should be drawn or not.
void	jumpDrawablesToCurrentState () Call `Drawable.jumpToCurrentState()` on all Drawable objects associated with this view.
void	layout (int l, int t, int r, int b) Assign a size and position to a view and all of its descendants This is the second phase of the layout mechanism.
final void	measure (int widthMeasureSpec, int heightMeasureSpec) This is called to find out how big a view should be.
void	offsetLeftAndRight (int offset) Offset this view's horizontal location by the specified amount of pixels.
void	offsetTopAndBottom (int offset) Offset this view's vertical location by the specified number of pixels.
void	onCancelPendingInputEvents () Called as the result of a call to `cancelPendingInputEvents()` on this view or a parent view.
boolean	onCheckIsTextEditor () Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it.
InputConnection	onCreateInputConnection (EditorInfo outAttrs) Create a new InputConnection for an InputMethod to interact with the view.
boolean	onDragEvent (DragEvent event) Handles drag events sent by the system following a call to `startDrag()`.
boolean	onFilterTouchEventForSecurity (MotionEvent event) Filter the touch event to apply security policies.
void	onFinishTemporaryDetach () Called after `onStartTemporaryDetach()` when the container is done changing the view.
boolean	onGenericMotionEvent (MotionEvent event) Implement this method to handle generic motion events.
void	onHoverChanged (boolean hovered) Implement this method to handle hover state changes.
boolean	onHoverEvent (MotionEvent event) Implement this method to handle hover events.
void	onInitializeAccessibilityEvent (AccessibilityEvent event) Initializes an `AccessibilityEvent` with information about this View which is the event source.
void	onInitializeAccessibilityNodeInfo (AccessibilityNodeInfo info) Initializes an `AccessibilityNodeInfo` with information about this view.

boolean	onKeyDown (int keyCode, KeyEvent event) Default implementation of `KeyEvent.Callback.onKeyDown()`: perform press of the view when `KEYCODE_DPAD_CENTER` or `KEYCODE_ENTER` is released, if the view is enabled and clickable.
boolean	onKeyLongPress (int keyCode, KeyEvent event) Default implementation of `KeyEvent.Callback.onKeyLongPress()`: always returns false (doesn't handle the event).
boolean	onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) Default implementation of `KeyEvent.Callback.onKeyMultiple()`: always returns false (doesn't handle the event).
boolean	onKeyPreIme (int keyCode, KeyEvent event) Handle a key event before it is processed by any input method associated with the view hierarchy.
boolean	onKeyShortcut (int keyCode, KeyEvent event) Called on the focused view when a key shortcut event is not handled.
boolean	onKeyUp (int keyCode, KeyEvent event) Default implementation of `KeyEvent.Callback.onKeyUp()`: perform clicking of the view when `KEYCODE_DPAD_CENTER` or `KEYCODE_ENTER` is released.
void	onPopulateAccessibilityEvent (AccessibilityEvent event) Called from `dispatchPopulateAccessibilityEvent(AccessibilityEvent)` giving a chance to this View to populate the accessibility event with its text content.
void	onRtlPropertiesChanged (int layoutDirection) Called when any RTL property (layout direction or text direction or text alignment) has been changed.
void	onScreenStateChanged (int screenState) This method is called whenever the state of the screen this view is attached to changes.
void	onStartTemporaryDetach () This is called when a container is going to temporarily detach a child, with `ViewGroup.detachViewFromParent`.
boolean	onTouchEvent (MotionEvent event) Implement this method to handle touch screen motion events.
boolean	onTrackballEvent (MotionEvent event) Implement this method to handle trackball motion events.
void	onWindowFocusChanged (boolean hasWindowFocus) Called when the window containing this view gains or loses focus.
void	onWindowSystemUiVisibilityChanged (int visible) Override to find out when the window's requested system UI visibility has changed, that is the value returned by `getWindowSystemUiVisibility()`.
boolean	performAccessibilityAction (int action, Bundle arguments) Performs the specified accessibility action on the view.
boolean	performClick () Call this view's OnClickListener, if it is defined.
boolean	performHapticFeedback (int feedbackConstant) BZZZTT!!1! Provide haptic feedback to the user for this view.
boolean	performHapticFeedback (int feedbackConstant, int flags) BZZZTT!!1! Like `performHapticFeedback(int)` (/reference/android/view/View.html#performHapticFeedback(int)), with additional options.
boolean	performLongClick () Call this view's OnLongClickListener, if it is defined.
void	playSoundEffect (int soundConstant) Play a sound effect for this view.
boolean	post (Runnable action) Causes the Runnable to be added to the message queue.
boolean	postDelayed (Runnable action, long delayMillis) Causes the Runnable to be added to the message queue, to be run after the specified amount of time elapses.
void	postInvalidate (int left, int top, int right, int bottom) Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop.
void	postInvalidate () Cause an invalidate to happen on a subsequent cycle through the event loop.
void	postInvalidateDelayed (long delayMilliseconds, int left, int top, int right, int bottom) Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop.
void	postInvalidateDelayed (long delayMilliseconds) Cause an invalidate to happen on a subsequent cycle through the event loop.
void	postInvalidateOnAnimation (int left, int top, int right, int bottom) Cause an invalidate of the specified area to happen on the next animation time step, typically the next display frame.
void	postInvalidateOnAnimation () Cause an invalidate to happen on the next animation time step, typically the next display frame.
void	postOnAnimation (Runnable action) Causes the Runnable to execute on the next animation time step.

void	postOnAnimationDelayed (Runnable action, long delayMillis) Causes the Runnable to execute on the next animation time step, after the specified amount of time elapses.
void	refreshDrawableState () Call this to force a view to update its drawable state.
boolean	removeCallbacks (Runnable action) Removes the specified Runnable from the message queue.
void	removeOnAttachStateChangeListener (View.OnAttachStateChangeListener listener) Remove a listener for attach state changes.
void	removeOnLayoutChangeListener (View.OnLayoutChangeListener listener) Remove a listener for layout changes.
void	requestFitSystemWindows () Ask that a new dispatch of `fitSystemWindows(Rect)` be performed.
boolean	requestFocus (int direction, Rect previouslyFocusedRect) Call this to try to give focus to a specific view or to one of its descendants and give it hints about the direction and a specific rectangle that the focus is coming from.
final boolean	requestFocus (int direction) Call this to try to give focus to a specific view or to one of its descendants and give it a hint about what direction focus is heading.
final boolean	requestFocus () Call this to try to give focus to a specific view or to one of its descendants.
final boolean	requestFocusFromTouch () Call this to try to give focus to a specific view or to one of its descendants.
void	requestLayout () Call this when something has changed which has invalidated the layout of this view.
boolean	requestRectangleOnScreen (Rect rectangle) Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough.
boolean	requestRectangleOnScreen (Rect rectangle, boolean immediate) Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough.
static int	resolveSize (int size, int measureSpec) Version of `resolveSizeAndState(int, int, int)` returning only the `MEASURED_SIZE_MASK` bits of the result.
static int	resolveSizeAndState (int size, int measureSpec, int childMeasuredState) Utility to reconcile a desired size and state, with constraints imposed by a MeasureSpec.
void	restoreHierarchyState (SparseArray<Parcelable> container) Restore this view hierarchy's frozen state from the given container.
void	saveHierarchyState (SparseArray<Parcelable> container) Store this view hierarchy's frozen state into the given container.
void	scheduleDrawable (Drawable who, Runnable what, long when) Schedules an action on a drawable to occur at a specified time.
void	scrollBy (int x, int y) Move the scrolled position of your view.
void	scrollTo (int x, int y) Set the scrolled position of your view.
void	sendAccessibilityEvent (int eventType) Sends an accessibility event of the given type.
void	sendAccessibilityEventUnchecked (AccessibilityEvent event) This method behaves exactly as `sendAccessibilityEvent(int)` but takes as an argument an empty `AccessibilityEvent` and does not perform a check whether accessibility is enabled.
void	setAccessibilityDelegate (View.AccessibilityDelegate delegate) Sets a delegate for implementing accessibility support via composition as opposed to inheritance.
void	setAccessibilityLiveRegion (int mode) Sets the live region mode for this view.
void	setActivated (boolean activated) Changes the activated state of this view.
void	setAlpha (float alpha) Sets the opacity of the view.
void	setAnimation (Animation animation) Sets the next animation to play for this view.
void	setBackground (Drawable background) Set the background to a given Drawable, or remove the background.
void	setBackgroundColor (int color) Sets the background color for this view.
void	setBackgroundDrawable (Drawable background) *This method was deprecated in API level 16. use `setBackground(Drawable)` instead*
void	setBackgroundResource (int resid) Set the background to a given resource.
final void	setBottom (int bottom) Sets the bottom position of this view relative to its parent.
void	setCameraDistance (float distance) Sets the distance along the Z axis (orthogonal to the X/Y plane on which views are drawn) from the camera to this view.
void	setClickable (boolean clickable) Enables or disables click events for this view.
void	setClipBounds (Rect clipBounds) Sets a rectangular area on this view to which the view will be clipped when it is drawn.
void	setContentDescription (CharSequence contentDescription) Sets the `View` description.

void	setDrawingCacheBackgroundColor (int color) Setting a solid background color for the drawing cache's bitmaps will improve performance and memory usage.
void	setDrawingCacheEnabled (boolean enabled) Enables or disables the drawing cache.
void	setDrawingCacheQuality (int quality) Set the drawing cache quality of this view.
void	setDuplicateParentStateEnabled (boolean enabled) Enables or disables the duplication of the parent's state into this view.
void	setEnabled (boolean enabled) Set the enabled state of this view.
void	setFadingEdgeLength (int length) Set the size of the faded edge used to indicate that more content in this view is available.
void	setFilterTouchesWhenObscured (boolean enabled) Sets whether the framework should discard touches when the view's window is obscured by another visible window.
void	setFitsSystemWindows (boolean fitSystemWindows) Sets whether or not this view should account for system screen decorations such as the status bar and inset its content; that is, controlling whether the default implementation of `fitSystemWindows(Rect)` will be executed.
void	setFocusable (boolean focusable) Set whether this view can receive the focus.
void	setFocusableInTouchMode (boolean focusableInTouchMode) Set whether this view can receive focus while in touch mode.
void	setHapticFeedbackEnabled (boolean hapticFeedbackEnabled) Set whether this view should have haptic feedback for events such as long presses.
void	setHasTransientState (boolean hasTransientState) Set whether this view is currently tracking transient state that the framework should attempt to preserve when possible.
void	setHorizontalFadingEdgeEnabled (boolean horizontalFadingEdgeEnabled) Define whether the horizontal edges should be faded when this view is scrolled horizontally.
void	setHorizontalScrollBarEnabled (boolean horizontalScrollBarEnabled) Define whether the horizontal scrollbar should be drawn or not.
void	setHovered (boolean hovered) Sets whether the view is currently hovered.
void	setId (int id) Sets the identifier for this view.
void	setImportantForAccessibility (int mode) Sets how to determine whether this view is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen.
void	setKeepScreenOn (boolean keepScreenOn) Controls whether the screen should remain on, modifying the value of `KEEP_SCREEN_ON`.
void	setLabelFor (int id) Sets the id of a view for which this view serves as a label for accessibility purposes.
void	setLayerPaint (Paint paint) Updates the `Paint` object used with the current layer (used only if the current layer type is not set to `LAYER_TYPE_NONE`).
void	setLayerType (int layerType, Paint paint) Specifies the type of layer backing this view.
void	setLayoutDirection (int layoutDirection) Set the layout direction for this view.
void	setLayoutParams (ViewGroup.LayoutParams params) Set the layout parameters associated with this view.
final void	setLeft (int left) Sets the left position of this view relative to its parent.
void	setLongClickable (boolean longClickable) Enables or disables long click events for this view.
void	setMinimumHeight (int minHeight) Sets the minimum height of the view.
void	setMinimumWidth (int minWidth) Sets the minimum width of the view.
void	setNextFocusDownId (int nextFocusDownId) Sets the id of the view to use when the next focus is `FOCUS_DOWN`.
void	setNextFocusForwardId (int nextFocusForwardId) Sets the id of the view to use when the next focus is `FOCUS_FORWARD`.
void	setNextFocusLeftId (int nextFocusLeftId) Sets the id of the view to use when the next focus is `FOCUS_LEFT`.
void	setNextFocusRightId (int nextFocusRightId) Sets the id of the view to use when the next focus is `FOCUS_RIGHT`.
void	setNextFocusUpId (int nextFocusUpId) Sets the id of the view to use when the next focus is `FOCUS_UP`.
void	setOnClickListener (View.OnClickListener l) Register a callback to be invoked when this view is clicked.
void	setOnCreateContextMenuListener (View.OnCreateContextMenuListener l) Register a callback to be invoked when the context menu for this view is being built.
void	setOnDragListener (View.OnDragListener l) Register a drag event listener callback object for this View.

void	setOnFocusChangeListener (View.OnFocusChangeListener l) Register a callback to be invoked when focus of this view changed.
void	setOnGenericMotionListener (View.OnGenericMotionListener l) Register a callback to be invoked when a generic motion event is sent to this view.
void	setOnHoverListener (View.OnHoverListener l) Register a callback to be invoked when a hover event is sent to this view.
void	setOnKeyListener (View.OnKeyListener l) Register a callback to be invoked when a hardware key is pressed in this view.
void	setOnLongClickListener (View.OnLongClickListener l) Register a callback to be invoked when this view is clicked and held.
void	setOnSystemUiVisibilityChangeListener (View.OnSystemUiVisibilityChangeListener l) Set a listener to receive callbacks when the visibility of the system bar changes.
void	setOnTouchListener (View.OnTouchListener l) Register a callback to be invoked when a touch event is sent to this view.
void	setOverScrollMode (int overScrollMode) Set the over-scroll mode for this view.
void	setPadding (int left, int top, int right, int bottom) Sets the padding.
void	setPaddingRelative (int start, int top, int end, int bottom) Sets the relative padding.
void	setPivotX (float pivotX) Sets the x location of the point around which the view is `rotated` and `scaled`.
void	setPivotY (float pivotY) Sets the y location of the point around which the view is `rotated` and `scaled`.
void	setPressed (boolean pressed) Sets the pressed state for this view.
final void	setRight (int right) Sets the right position of this view relative to its parent.
void	setRotation (float rotation) Sets the degrees that the view is rotated around the pivot point.
void	setRotationX (float rotationX) Sets the degrees that the view is rotated around the horizontal axis through the pivot point.
void	setRotationY (float rotationY) Sets the degrees that the view is rotated around the vertical axis through the pivot point.
void	setSaveEnabled (boolean enabled) Controls whether the saving of this view's state is enabled (that is, whether its `onSaveInstanceState()` method will be called).
void	setSaveFromParentEnabled (boolean enabled) Controls whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent.
void	setScaleX (float scaleX) Sets the amount that the view is scaled in x around the pivot point, as a proportion of the view's unscaled width.
void	setScaleY (float scaleY) Sets the amount that the view is scaled in Y around the pivot point, as a proportion of the view's unscaled width.
void	setScrollBarDefaultDelayBeforeFade (int scrollBarDefaultDelayBeforeFade) Define the delay before scrollbars fade.
void	setScrollBarFadeDuration (int scrollBarFadeDuration) Define the scrollbar fade duration.
void	setScrollBarSize (int scrollBarSize) Define the scrollbar size.
void	setScrollBarStyle (int style) Specify the style of the scrollbars.
void	setScrollContainer (boolean isScrollContainer) Change whether this view is one of the set of scrollable containers in its window.
void	setScrollX (int value) Set the horizontal scrolled position of your view.
void	setScrollY (int value) Set the vertical scrolled position of your view.
void	setScrollbarFadingEnabled (boolean fadeScrollbars) Define whether scrollbars will fade when the view is not scrolling.
void	setSelected (boolean selected) Changes the selection state of this view.
void	setSoundEffectsEnabled (boolean soundEffectsEnabled) Set whether this view should have sound effects enabled for events such as clicking and touching.
void	setSystemUiVisibility (int visibility) Request that the visibility of the status bar or other screen/window decorations be changed.
void	setTag (int key, Object tag) Sets a tag associated with this view and a key.
void	setTag (Object tag) Sets the tag associated with this view.
void	setTextAlignment (int textAlignment) Set the text alignment.
void	setTextDirection (int textDirection) Set the text direction.
final void	setTop (int top) Sets the top position of this view relative to its parent.
void	setTouchDelegate (TouchDelegate delegate) Sets the TouchDelegate for this View.
void	setTranslationX (float translationX) Sets the horizontal location of this view relative to its `left` position.

void	setTranslationY (float translationY) Sets the vertical location of this view relative to its `top` position.
void	setVerticalFadingEdgeEnabled (boolean verticalFadingEdgeEnabled) Define whether the vertical edges should be faded when this view is scrolled vertically.
void	setVerticalScrollBarEnabled (boolean verticalScrollBarEnabled) Define whether the vertical scrollbar should be drawn or not.
void	setVerticalScrollbarPosition (int position) Set the position of the vertical scroll bar.
void	setVisibility (int visibility) Set the enabled state of this view.
void	setWillNotCacheDrawing (boolean willNotCacheDrawing) When a View's drawing cache is enabled, drawing is redirected to an offscreen bitmap.
void	setWillNotDraw (boolean willNotDraw) If this view doesn't do any drawing on its own, set this flag to allow further optimizations.
void	setX (float x) Sets the visual x position of this view, in pixels.
void	setY (float y) Sets the visual y position of this view, in pixels.
boolean	showContextMenu () Bring up the context menu for this view.
ActionMode	startActionMode (ActionMode.Callback callback) Start an action mode.
void	startAnimation (Animation animation) Start the specified animation now.
final boolean	startDrag (ClipData data, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) Starts a drag and drop operation.
String	toString () Returns a string containing a concise, human-readable description of this object.
void	unscheduleDrawable (Drawable who) Unschedule any events associated with the given Drawable.
void	unscheduleDrawable (Drawable who, Runnable what) Cancels a scheduled action on a drawable.
boolean	willNotCacheDrawing () Returns whether or not this View can cache its drawing or not.
boolean	willNotDraw () Returns whether or not this View draws on its own.

Protected Methods

boolean	awakenScrollBars (int startDelay) Trigger the scrollbars to draw.
boolean	awakenScrollBars (int startDelay, boolean invalidate) Trigger the scrollbars to draw.
boolean	awakenScrollBars () Trigger the scrollbars to draw.
int	computeHorizontalScrollExtent () Compute the horizontal extent of the horizontal scrollbar's thumb within the horizontal range.
int	computeHorizontalScrollOffset () Compute the horizontal offset of the horizontal scrollbar's thumb within the horizontal range.
int	computeHorizontalScrollRange () Compute the horizontal range that the horizontal scrollbar represents.
int	computeVerticalScrollExtent () Compute the vertical extent of the horizontal scrollbar's thumb within the vertical range.
int	computeVerticalScrollOffset () Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range.
int	computeVerticalScrollRange () Compute the vertical range that the vertical scrollbar represents.
void	dispatchDraw (Canvas canvas) Called by draw to draw the child views.
boolean	dispatchGenericFocusedEvent (MotionEvent event) Dispatch a generic motion event to the currently focused view.
boolean	dispatchGenericPointerEvent (MotionEvent event) Dispatch a generic motion event to the view under the first pointer.
boolean	dispatchHoverEvent (MotionEvent event) Dispatch a hover event.
void	dispatchRestoreInstanceState (SparseArray<Parcelable> container) Called by `restoreHierarchyState(android.util.SparseArray)` to retrieve the state for this view and its children.
void	dispatchSaveInstanceState (SparseArray<Parcelable> container) Called by `saveHierarchyState(android.util.SparseArray)` to store the state for this view and its children.

void	dispatchSetActivated (boolean activated) Dispatch setActivated to all of this View's children.
void	dispatchSetPressed (boolean pressed) Dispatch setPressed to all of this View's children.
void	dispatchSetSelected (boolean selected) Dispatch setSelected to all of this View's children.
void	dispatchVisibilityChanged (View changedView, int visibility) Dispatch a view visibility change down the view hierarchy.
void	drawableStateChanged () This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown.
boolean	fitSystemWindows (Rect insets) Called by the view hierarchy when the content insets for a window have changed, to allow it to adjust its content to fit within those windows.
float	getBottomFadingEdgeStrength () Returns the strength, or intensity, of the bottom faded edge.
int	getBottomPaddingOffset () Amount by which to extend the bottom fading region.
ContextMenu.ContextMenuInfo	getContextMenuInfo () Views should implement this if they have extra information to associate with the context menu.
int	getHorizontalScrollbarHeight () Returns the height of the horizontal scrollbar.
float	getLeftFadingEdgeStrength () Returns the strength, or intensity, of the left faded edge.
int	getLeftPaddingOffset () Amount by which to extend the left fading region.
float	getRightFadingEdgeStrength () Returns the strength, or intensity, of the right faded edge.
int	getRightPaddingOffset () Amount by which to extend the right fading region.
int	getSuggestedMinimumHeight () Returns the suggested minimum height that the view should use.
int	getSuggestedMinimumWidth () Returns the suggested minimum width that the view should use.
float	getTopFadingEdgeStrength () Returns the strength, or intensity, of the top faded edge.
int	getTopPaddingOffset () Amount by which to extend the top fading region.
int	getWindowAttachCount ()
void	initializeFadingEdge (TypedArray a) Initializes the fading edges from a given set of styled attributes.
void	initializeScrollbars (TypedArray a) Initializes the scrollbars from a given set of styled attributes.
boolean	isPaddingOffsetRequired () If the View draws content inside its padding and enables fading edges, it needs to support padding offsets.
static int[]	mergeDrawableStates (int[] baseState, int[] additionalState) Merge your own state values in *additionalState* into the base state values *baseState* that were returned by `onCreateDrawableState(int)`.
void	onAnimationEnd () Invoked by a parent ViewGroup to notify the end of the animation currently associated with this view.
void	onAnimationStart () Invoked by a parent ViewGroup to notify the start of the animation currently associated with this view.
void	onAttachedToWindow () This is called when the view is attached to a window.
void	onConfigurationChanged (Configuration newConfig) Called when the current configuration of the resources being used by the application have changed.
void	onCreateContextMenu (ContextMenu menu) Views should implement this if the view itself is going to add items to the context menu.
int[]	onCreateDrawableState (int extraSpace) Generate the new `Drawable` state for this view.
void	onDetachedFromWindow () This is called when the view is detached from a window.
void	onDisplayHint (int hint) Gives this view a hint about whether is displayed or not.
void	onDraw (Canvas canvas) Implement this to do your drawing.
final void	onDrawScrollBars (Canvas canvas) Request the drawing of the horizontal and the vertical scrollbar.
void	onFinishInflate () Finalize inflating a view from XML.
void	onFocusChanged (boolean gainFocus, int direction, Rect previouslyFocusedRect) Called by the view system when the focus state of this view changes.
void	onLayout (boolean changed, int left, int top, int right, int bottom) Called from layout when this view should assign a size and position to each of its children.
void	onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height.
void	onOverScrolled (int scrollX, int scrollY, boolean clampedX, boolean clampedY) Called by `overScrollBy(int, int, int, int, int, int, int, int, boolean)` to respond to the results of an over-scroll operation.

void	onRestoreInstanceState (Parcelable state) Hook allowing a view to re-apply a representation of its internal state that had previously been generated by `onSaveInstanceState()`.
Parcelable	onSaveInstanceState () Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state.
void	onScrollChanged (int l, int t, int oldl, int oldt) This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents).
boolean	onSetAlpha (int alpha) Invoked if there is a Transform that involves alpha.
void	onSizeChanged (int w, int h, int oldw, int oldh) This is called during layout when the size of this view has changed.
void	onVisibilityChanged (View changedView, int visibility) Called when the visibility of the view or an ancestor of the view is changed.
void	onWindowVisibilityChanged (int visibility) Called when the window containing has change its visibility (between `GONE`, `INVISIBLE`, and `VISIBLE`).
boolean	overScrollBy (int deltaX, int deltaY, int scrollX, int scrollY, int scrollRangeX, int scrollRangeY, int maxOverScrollX, int maxOverScrollY, boolean isTouchEvent) Scroll the view with standard behavior for scrolling beyond the normal content boundaries.
final void	setMeasuredDimension (int measuredWidth, int measuredHeight) This method must be called by `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)) to store the measured width and measured height.
boolean	verifyDrawable (Drawable who) If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying.

Inherited Methods [Expand]

- From class java.lang.Object
- From interface android.graphics.drawable.Drawable.Callback
- From interface android.view.KeyEvent.Callback
- From interface android.view.accessibility.AccessibilityEventSource

XML Attributes

android:accessibilityLiveRegion

Indicates to accessibility services whether the user should be notified when this view changes.

May be an integer value, such as "`100`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

May be one of the following constant values.

Constant	Value	Description
`none`	0	Accessibility services should not announce changes to this view.
`polite`	1	Accessibility services should announce changes to this view.
`assertive`	2	Accessibility services should interrupt ongoing speech to immediately announce changes to this view.

This corresponds to the global attribute resource symbol `accessibilityLiveRegion` (/reference/android/R.attr.html#accessibilityLiveRegion).

Related Methods
setAccessibilityLiveRegion(int)

android:alpha

alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque).

Must be a floating point value, such as "`1.2`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `alpha` (/reference/android/R.attr.html#alpha).

Related Methods
setAlpha(float)

android:background

A drawable to use as the background. This can be either a reference to a full drawable resource (such as a PNG image, 9-patch, XML state list description, etc), or a solid color such as "#ff000000" (black).

May be a reference to another resource, in the form "`@[+][`*`package`*`:]`*`type`*`:`*`name`*" or to a theme attribute in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*".

May be a color value, in the form of "`#`*`rgb`*", "`#`*`argb`*", "`#`*`rrggbb`*", or "`#`*`aarrggbb`*".

This corresponds to the global attribute resource symbol `background` (/reference/android/R.attr.html#background).

Related Methods
setBackgroundResource(int)

android:clickable

Defines whether this view reacts to click events.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `clickable` (/reference/android/R.attr.html#clickable).

Related Methods
setClickable(boolean)

android:contentDescription

Defines text that briefly describes content of the view. This property is used primarily for accessibility. Since some views do not have textual representation this attribute can be used for providing such.

Must be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character.

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:]` `[`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `contentDescription` (/reference/android/R.attr.html#contentDescription).

Related Methods
setContentDescription(CharSequence)

android:drawingCacheQuality

Defines the quality of translucent drawing caches. This property is used only when the drawing cache is enabled and translucent. The default value is auto.

Must be one of the following constant values.

Constant	Value	Description
`auto`	0	Lets the framework decide what quality level should be used for the drawing cache.
`low`	1	Low quality. When set to low quality, the drawing cache uses a lower color depth, thus losing precision in rendering gradients, but uses less memory.
`high`	2	High quality. When set to high quality, the drawing cache uses a higher color depth but uses more memory.

This corresponds to the global attribute resource symbol `drawingCacheQuality` (/reference/android/R.attr.html#drawingCacheQuality).

Related Methods
setDrawingCacheQuality(int)

android:duplicateParentState

When this attribute is set to true, the view gets its drawable state (focused, pressed, etc.) from its direct parent rather than from itself.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:]` `[`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `duplicateParentState` (/reference/android/R.attr.html#duplicateParentState).

Related Methods

android:fadeScrollbars

Defines whether to fade out scrollbars when they are not in use.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:]` `[`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `fadeScrollbars` (/reference/android/R.attr.html#fadeScrollbars).

Related Methods
setScrollbarFadingEnabled(boolean)

android:fadingEdgeLength

Defines the length of the fading edges.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:]` `[`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `fadingEdgeLength` (/reference/android/R.attr.html#fadingEdgeLength).

Related Methods
getVerticalFadingEdgeLength()

android:filterTouchesWhenObscured

Specifies whether to filter touches when the view's window is obscured by another visible window. When set to true, the view will not receive touches whenever a toast, dialog or other window appears above the view's window. Refer to the `View` (/reference/android/view/View.html) security documentation for more details.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:]` `[`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `filterTouchesWhenObscured` (/reference/android/R.attr.html#filterTouchesWhenObscured).

Related Methods
setFilterTouchesWhenObscured(boolean)

android:fitsSystemWindows

Boolean internal attribute to adjust view layout based on system windows such as the status bar. If true, adjusts the padding of this view to leave space for the system windows. Will only take effect if this view is in a non-embedded activity.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `fitsSystemWindows` (/reference/android/R.attr.html#fitsSystemWindows).

Related Methods
setFitsSystemWindows(boolean)

android:focusable

Boolean that controls whether a view can take focus. By default the user can not move focus to a view; by setting this attribute to true the view is allowed to take focus. This value does not impact the behavior of directly calling `requestFocus()` (/reference/android/view/View.html#requestFocus()), which will always request focus regardless of this view. It only impacts where focus navigation will try to move focus.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `focusable` (/reference/android/R.attr.html#focusable).

Related Methods
setFocusable(boolean)

android:focusableInTouchMode

Boolean that controls whether a view can take focus while in touch mode. If this is true for a view, that view can gain focus when clicked on, and can keep focus if another view is clicked on that doesn't have this attribute set to true.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `focusableInTouchMode` (/reference/android/R.attr.html#focusableInTouchMode).

Related Methods
setFocusableInTouchMode(boolean)

android:hapticFeedbackEnabled

Boolean that controls whether a view should have haptic feedback enabled for events such as long presses.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `hapticFeedbackEnabled` (/reference/android/R.attr.html#hapticFeedbackEnabled).

Related Methods
setHapticFeedbackEnabled(boolean)

android:id

Supply an identifier name for this view, to later retrieve it with `View.findViewById()` (/reference/android/view/View.html#findViewById(int)) or `Activity.findViewById()` (/reference/android/app/Activity.html#findViewById(int)). This must be a resource reference; typically you set this using the @+ syntax to create a new ID resources. For example: `android:id="@+id/my_id"` which allows you to later retrieve the view with `findViewById(R.id.my_id)`.

Must be a reference to another resource, in the form "`@[+][`*package*`:]`*type*`:`*name*" or to a theme attribute in the form "`?[`*package*`:]` `[`*type*`:]`*name*".

This corresponds to the global attribute resource symbol `id` (/reference/android/R.attr.html#id).

Related Methods
setId(int)

android:importantForAccessibility

Controls how this View is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen. Note: While not recommended, an accessibility service may decide to ignore this attribute and operate on all views in the view tree.

May be an integer value, such as "`100`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

May be one of the following constant values.

Constant	Value	Description
`auto`	0	The system determines whether the view is important for accessibility - default (recommended).
`yes`	1	The view is important for accessibility.
`no`	2	The view is not important for accessibility.
`noHideDescendants`	4	The view is not important for accessibility, nor are any of its descendant views.

This corresponds to the global attribute resource symbol `importantForAccessibility` (/reference/android/R.attr.html#importantForAccessibility).

Related Methods
setImportantForAccessibility(int)

android:isScrollContainer

Set this if the view will serve as a scrolling container, meaing that it can be resized to shrink its overall window so that there will be space for an input method. If not set, the default value will be true if "scrollbars" has the vertical scrollbar set, else it will be false.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `isScrollContainer` (/reference/android/R.attr.html#isScrollContainer).

Related Methods
setScrollContainer(boolean)

android:keepScreenOn

Controls whether the view's window should keep the screen on while visible.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `keepScreenOn` (/reference/android/R.attr.html#keepScreenOn).

Related Methods
setKeepScreenOn(boolean)

android:layerType

Specifies the type of layer backing this view. The default value is none. Refer to `setLayerType(int, android.graphics.Paint)` (/reference/android/view/View.html#setLayerType(int, android.graphics.Paint)) for more information.

Must be one of the following constant values.

Constant	Value	Description
`none`	0	Don't use a layer.
`software`	1	Use a software layer. Refer to `setLayerType(int, android.graphics.Paint)` for more information.
`hardware`	2	Use a hardware layer. Refer to `setLayerType(int, android.graphics.Paint)` for more information.

This corresponds to the global attribute resource symbol `layerType` (/reference/android/R.attr.html#layerType).

Related Methods
setLayerType(int,Paint)

android:layoutDirection

Defines the direction of layout drawing. This typically is associated with writing direction of the language script used. The possible values are "ltr" for Left-to-Right, "rtl" for Right-to-Left, "locale" and "inherit" from parent view. If there is nothing to inherit, "locale" is used. "locale" falls back to "en-US". "ltr" is the direction used in "en-US". The default for this attribute is "inherit".

Must be one of the following constant values.

Constant	Value	Description
`ltr`	0	Left-to-Right
`rtl`	1	Right-to-Left
`inherit`	2	Inherit from parent
`locale`	3	Locale

This corresponds to the global attribute resource symbol `layoutDirection` (/reference/android/R.attr.html#layoutDirection).

Related Methods
setLayoutDirection(int)

android:longClickable

Defines whether this view reacts to long click events.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `longClickable` (/reference/android/R.attr.html#longClickable).

Related Methods
setLongClickable(boolean)

android:minHeight

Defines the minimum height of the view. It is not guaranteed the view will be able to achieve this minimum height (for example, if its parent layout constrains it with less available height).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `minHeight` (/reference/android/R.attr.html#minHeight).

Related Methods
setMinimumHeight(int)

android:minWidth

Defines the minimum width of the view. It is not guaranteed the view will be able to achieve this minimum width (for example, if its parent layout constrains it with less available width).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `minWidth` (/reference/android/R.attr.html#minWidth).

Related Methods
setMinimumWidth(int)

android:nextFocusDown

Defines the next view to give focus to when the next focus is `FOCUS_DOWN` (/reference/android/view/View.html#FOCUS_DOWN) If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` (/reference/java/lang/RuntimeException.html) will result when the reference is accessed.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `nextFocusDown` (/reference/android/R.attr.html#nextFocusDown).

Related Methods
setNextFocusDownId(int)

android:nextFocusForward

Defines the next view to give focus to when the next focus is `FOCUS_FORWARD` (/reference/android/view/View.html#FOCUS_FORWARD) If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` (/reference/java/lang/RuntimeException.html) will result when the reference is accessed.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `nextFocusForward` (/reference/android/R.attr.html#nextFocusForward).

Related Methods
setNextFocusForwardId(int)

android:nextFocusLeft

Defines the next view to give focus to when the next focus is `FOCUS_LEFT` (/reference/android/view/View.html#FOCUS_LEFT). If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` (/reference/java/lang/RuntimeException.html) will result when the reference is accessed.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `nextFocusLeft` (/reference/android/R.attr.html#nextFocusLeft).

Related Methods
setNextFocusLeftId(int)

android:nextFocusRight

Defines the next view to give focus to when the next focus is `FOCUS_RIGHT` (/reference/android/view/View.html#FOCUS_RIGHT) If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` (/reference/java/lang/RuntimeException.html) will result when the reference is accessed.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `nextFocusRight` (/reference/android/R.attr.html#nextFocusRight).

Related Methods
setNextFocusRightId(int)

android:nextFocusUp

Defines the next view to give focus to when the next focus is `FOCUS_UP` (/reference/android/view/View.html#FOCUS_UP) If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a `RuntimeException` (/reference/java/lang/RuntimeException.html) will result when the reference is accessed.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `nextFocusUp` (/reference/android/R.attr.html#nextFocusUp).

Related Methods
setNextFocusUpId(int)

android:onClick

Name of the method in this View's context to invoke when the view is clicked. This name must correspond to a public method that takes exactly one parameter of type View. For instance, if you specify `android:onClick="sayHello"`, you must declare a `public void sayHello(View v)` method of your context (typically, your Activity).

Must be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character.

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `onClick` (/reference/android/R.attr.html#onClick).

Related Methods

android:padding

Sets the padding, in pixels, of all four edges. Padding is defined as space between the edges of the view and the view's content. A views size will include it's padding. If a `background` (/reference/android/R.attr.html#background) is provided, the padding will initially be set to that (0 if the drawable does not have padding). Explicitly setting a padding value will override the corresponding padding found in the background.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `padding` (/reference/android/R.attr.html#padding).

Related Methods
setPaddingRelative(int,int,int,int)

android:paddingBottom

Sets the padding, in pixels, of the bottom edge; see `padding` (/reference/android/R.attr.html#padding).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `paddingBottom` (/reference/android/R.attr.html#paddingBottom).

Related Methods
setPaddingRelative(int,int,int,int)

android:paddingEnd

Sets the padding, in pixels, of the end edge; see `padding` (/reference/android/R.attr.html#padding).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `paddingEnd` (/reference/android/R.attr.html#paddingEnd).

Related Methods
setPaddingRelative(int,int,int,int)

android:paddingLeft

Sets the padding, in pixels, of the left edge; see `padding` (/reference/android/R.attr.html#padding).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `paddingLeft` (/reference/android/R.attr.html#paddingLeft).

Related Methods
setPadding(int,int,int,int)

android:paddingRight

Sets the padding, in pixels, of the right edge; see `padding` (/reference/android/R.attr.html#padding).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `paddingRight` (/reference/android/R.attr.html#paddingRight).

Related Methods
setPadding(int,int,int,int)

android:paddingStart

Sets the padding, in pixels, of the start edge; see `padding` (/reference/android/R.attr.html#padding).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `paddingStart` (/reference/android/R.attr.html#paddingStart).

Related Methods
setPaddingRelative(int,int,int,int)

android:paddingTop

Sets the padding, in pixels, of the top edge; see `padding` (/reference/android/R.attr.html#padding).

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `paddingTop` (/reference/android/R.attr.html#paddingTop).

Related Methods
setPaddingRelative(int,int,int,int)

android:requiresFadingEdge

Defines which edges should be faded on scrolling.

Must be one or more (separated by '|') of the following constant values.

Constant	Value	Description
`none`	0x00000000	No edge is faded.
`horizontal`	0x00001000	Fades horizontal edges only.
`vertical`	0x00002000	Fades vertical edges only.

This corresponds to the global attribute resource symbol `requiresFadingEdge` (/reference/android/R.attr.html#requiresFadingEdge).

Related Methods
setVerticalFadingEdgeEnabled(boolean)

android:rotation

rotation of the view, in degrees.

Must be a floating point value, such as "`1.2`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `rotation` (/reference/android/R.attr.html#rotation).

Related Methods
setRotation(float)

android:rotationX

rotation of the view around the x axis, in degrees.

Must be a floating point value, such as "`1.2`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `rotationX` (/reference/android/R.attr.html#rotationX).

Related Methods
setRotationX(float)

android:rotationY

rotation of the view around the y axis, in degrees.

Must be a floating point value, such as "`1.2`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `rotationY` (/reference/android/R.attr.html#rotationY).

Related Methods
setRotationY(float)

android:saveEnabled

If unset, no state will be saved for this view when it is being frozen. The default is true, allowing the view to be saved (however it also must have an ID assigned to it for its state to be saved). Setting this to false only disables the state for this view, not for its children which may still be saved.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `saveEnabled` (/reference/android/R.attr.html#saveEnabled).

Related Methods
setSaveEnabled(boolean)

android:scaleX

scale of the view in the x direction.

Must be a floating point value, such as "`1.2`".

This may also be a reference to a resource (in the form "`@[`*`package`*`:]`*`type`*`:`*`name`*") or theme attribute (in the form "`?[`*`package`*`:][`*`type`*`:]`*`name`*") containing a value of this type.

This corresponds to the global attribute resource symbol `scaleX` (/reference/android/R.attr.html#scaleX).

Related Methods
setScaleX(float)

android:scaleY

scale of the view in the y direction.

Must be a floating point value, such as "`1.2`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scaleY` (/reference/android/R.attr.html#scaleY).

Related Methods
setScaleY(float)

android:scrollX

The initial horizontal scroll offset, in pixels.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scrollX` (/reference/android/R.attr.html#scrollX).

Related Methods

android:scrollY

The initial vertical scroll offset, in pixels.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scrollY` (/reference/android/R.attr.html#scrollY).

Related Methods

android:scrollbarAlwaysDrawHorizontalTrack

Defines whether the horizontal scrollbar track should always be drawn.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scrollbarAlwaysDrawHorizontalTrack` (/reference/android/R.attr.html#scrollbarAlwaysDrawHorizontalTrack).

Related Methods

android:scrollbarAlwaysDrawVerticalTrack

Defines whether the vertical scrollbar track should always be drawn.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scrollbarAlwaysDrawVerticalTrack` (/reference/android/R.attr.html#scrollbarAlwaysDrawVerticalTrack).

Related Methods

android:scrollbarDefaultDelayBeforeFade

Defines the delay in milliseconds that a scrollbar waits before fade out.

Must be an integer value, such as "`100`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scrollbarDefaultDelayBeforeFade` (/reference/android/R.attr.html#scrollbarDefaultDelayBeforeFade).

Related Methods
setScrollBarDefaultDelayBeforeFade(int)

android:scrollbarFadeDuration

Defines the delay in milliseconds that a scrollbar takes to fade out.

Must be an integer value, such as "`100`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scrollbarFadeDuration` (/reference/android/R.attr.html#scrollbarFadeDuration).

Related Methods
setScrollBarFadeDuration(int)

android:scrollbarSize

Sets the width of vertical scrollbars and height of horizontal scrollbars.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `scrollbarSize` (/reference/android/R.attr.html#scrollbarSize).

Related Methods
setScrollBarSize(int)

android:scrollbarStyle

Controls the scrollbar style and position. The scrollbars can be overlaid or inset. When inset, they add to the padding of the view. And the scrollbars can be drawn inside the padding area or on the edge of the view. For example, if a view has a background drawable and you want to draw the scrollbars inside the padding specified by the drawable, you can use insideOverlay or insideInset. If you want them to appear at the edge of the view, ignoring the padding, then you can use outsideOverlay or outsideInset.

Must be one of the following constant values.

Constant	Value	Description
`insideOverlay`	0x0	Inside the padding and overlaid
`insideInset`	0x01000000	Inside the padding and inset
`outsideOverlay`	0x02000000	Edge of the view and overlaid
`outsideInset`	0x03000000	Edge of the view and inset

This corresponds to the global attribute resource symbol `scrollbarStyle` (/reference/android/R.attr.html#scrollbarStyle).

Related Methods
setScrollBarStyle(int)

android:scrollbarThumbHorizontal

Defines the horizontal scrollbar thumb drawable.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `scrollbarThumbHorizontal` (/reference/android/R.attr.html#scrollbarThumbHorizontal).

Related Methods

android:scrollbarThumbVertical

Defines the vertical scrollbar thumb drawable.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `scrollbarThumbVertical` (/reference/android/R.attr.html#scrollbarThumbVertical).

Related Methods

android:scrollbarTrackHorizontal

Defines the horizontal scrollbar track drawable.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `scrollbarTrackHorizontal` (/reference/android/R.attr.html#scrollbarTrackHorizontal).

Related Methods

android:scrollbarTrackVertical

Defines the vertical scrollbar track drawable.

Must be a reference to another resource, in the form "`@[+][package:]type:name`" or to a theme attribute in the form "`?[package:][type:]name`".

This corresponds to the global attribute resource symbol `scrollbarTrackVertical` (/reference/android/R.attr.html#scrollbarTrackVertical).

Related Methods

android:scrollbars

Defines which scrollbars should be displayed on scrolling or not.

Must be one or more (separated by '|') of the following constant values.

Constant	Value	Description
`none`	0x00000000	No scrollbar is displayed.
`horizontal`	0x00000100	Displays horizontal scrollbar only.
`vertical`	0x00000200	Displays vertical scrollbar only.

This corresponds to the global attribute resource symbol `scrollbars` (/reference/android/R.attr.html#scrollbars).

Related Methods

android:soundEffectsEnabled

Boolean that controls whether a view should have sound effects enabled for events such as clicking and touching.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `soundEffectsEnabled` (/reference/android/R.attr.html#soundEffectsEnabled).

Related Methods
setSoundEffectsEnabled(boolean)

android:tag

Supply a tag for this view containing a String, to be retrieved later with `View.getTag()` (/reference/android/view/View.html#getTag()) or searched for with `View.findViewWithTag()` (/reference/android/view/View.html#findViewWithTag(java.lang.Object)). It is generally preferable to use IDs (through the android:id attribute) instead of tags because they are faster and allow for compile-time type checking.

Must be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character.

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `tag` (/reference/android/R.attr.html#tag).

Related Methods

android:textAlignment

Defines the alignment of the text. A heuristic is used to determine the resolved text alignment.

May be an integer value, such as "`100`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

May be one of the following constant values.

Constant	Value	Description
`inherit`	0	Default
`gravity`	1	Default for the root view. The gravity determines the alignment, ALIGN_NORMAL, ALIGN_CENTER, or ALIGN_OPPOSITE, which are relative to each paragraph's text direction
`textStart`	2	Align to the start of the paragraph, e.g. ALIGN_NORMAL.
`textEnd`	3	Align to the end of the paragraph, e.g. ALIGN_OPPOSITE.
`center`	4	Center the paragraph, e.g. ALIGN_CENTER.
`viewStart`	5	Align to the start of the view, which is ALIGN_LEFT if the view's resolved layoutDirection is LTR, and ALIGN_RIGHT otherwise.
`viewEnd`	6	Align to the end of the view, which is ALIGN_RIGHT if the view's resolved layoutDirection is LTR, and ALIGN_LEFT otherwise

This corresponds to the global attribute resource symbol `textAlignment` (/reference/android/R.attr.html#textAlignment).

Related Methods
setTextAlignment(int)

android:textDirection

Defines the direction of the text. A heuristic is used to determine the resolved text direction of paragraphs.

May be an integer value, such as "`100`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

May be one of the following constant values.

Constant	Value	Description
`inherit`	0	Default
`firstStrong`	1	Default for the root view. The first strong directional character determines the paragraph direction. If there is no strong directional character, the paragraph direction is the view's resolved layout direction.
`anyRtl`	2	The paragraph direction is RTL if it contains any strong RTL character, otherwise it is LTR if it contains any strong LTR characters. If there are neither, the paragraph direction is the view's resolved layout direction.
`ltr`	3	The paragraph direction is left to right.
`rtl`	4	The paragraph direction is right to left.
`locale`	5	The paragraph direction is coming from the system Locale.

This corresponds to the global attribute resource symbol `textDirection` (/reference/android/R.attr.html#textDirection).

Related Methods
setTextDirection(int)

android:transformPivotX

x location of the pivot point around which the view will rotate and scale. This xml attribute sets the pivotX property of the View.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `transformPivotX` (/reference/android/R.attr.html#transformPivotX).

Related Methods
setPivotX(float)

android:transformPivotY

y location of the pivot point around which the view will rotate and scale. This xml attribute sets the pivotY property of the View.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:]` `[`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `transformPivotY` (/reference/android/R.attr.html#transformPivotY).

Related Methods
setPivotY(float)

android:translationX

translation in x of the view. This value is added post-layout to the left property of the view, which is set by its layout.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `translationX` (/reference/android/R.attr.html#translationX).

Related Methods
setTranslationX(float)

android:translationY

translation in y of the view. This value is added post-layout to the left property of the view, which is set by its layout.

Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters).

This may also be a reference to a resource (in the form "`@[package:]type:name`") or theme attribute (in the form "`?[package:][type:]name`") containing a value of this type.

This corresponds to the global attribute resource symbol `translationY` (/reference/android/R.attr.html#translationY).

Related Methods
setTranslationY(float)

android:visibility

Controls the initial visibility of the view.

Must be one of the following constant values.

Constant	Value	Description
`visible`	0	Visible on screen; the default value.
`invisible`	1	Not displayed, but taken into account during layout (space is left for it).
`gone`	2	Completely hidden, as if the view had not been added.

This corresponds to the global attribute resource symbol `visibility` (/reference/android/R.attr.html#visibility).

Related Methods
setVisibility(int)

Constants

public static final int **ACCESSIBILITY_LIVE_REGION_ASSERTIVE** Added in API level 19

Live region mode specifying that accessibility services should interrupt ongoing speech to immediately announce changes to this view.

Use with `setAccessibilityLiveRegion(int)` (/reference/android/view/View.html#setAccessibilityLiveRegion(int)).

Constant Value: 2 (0x00000002)

public static final int **ACCESSIBILITY_LIVE_REGION_NONE** Added in API level 19

Live region mode specifying that accessibility services should not automatically announce changes to this view. This is the default live region mode for most views.

Use with `setAccessibilityLiveRegion(int)` (/reference/android/view/View.html#setAccessibilityLiveRegion(int)).

Constant Value: 0 (0x00000000)

public static final int **ACCESSIBILITY_LIVE_REGION_POLITE** Added in API level 19

Live region mode specifying that accessibility services should announce changes to this view.

Use with `setAccessibilityLiveRegion(int)` (/reference/android/view/View.html#setAccessibilityLiveRegion(int)).

Constant Value: 1 (0x00000001)

public static final int **DRAWING_CACHE_QUALITY_AUTO** Added in API level 1

Enables automatic quality mode for the drawing cache.

Constant Value: 0 (0x00000000)

public static final int **DRAWING_CACHE_QUALITY_HIGH** Added in API level 1

Enables high quality mode for the drawing cache.

Constant Value: 1048576 (0x00100000)

public static final int **DRAWING_CACHE_QUALITY_LOW** Added in API level 1

Enables low quality mode for the drawing cache.

Constant Value: 524288 (0x00080000)

public static final int **FIND_VIEWS_WITH_CONTENT_DESCRIPTION** Added in API level 14

Find find views that contain the specified content description.

See Also
`findViewsWithText(ArrayList, CharSequence, int)`

Constant Value: 2 (0x00000002)

public static final int **FIND_VIEWS_WITH_TEXT** Added in API level 14

Find views that render the specified text.

See Also
`findViewsWithText(ArrayList, CharSequence, int)`

Constant Value: 1 (0x00000001)

public static final int **FOCUSABLES_ALL** Added in API level 4

View flag indicating whether `addFocusables(ArrayList, int, int)` (/reference/android/view/View.html#addFocusables(java.util.ArrayList<android.view.View>, int, int)) should add all focusable Views regardless if they are focusable in touch mode.

Constant Value: 0 (0x00000000)

public static final int **FOCUSABLES_TOUCH_MODE** Added in API level 4

View flag indicating whether `addFocusables(ArrayList, int, int)` (/reference/android/view/View.html#addFocusables(java.util.ArrayList<android.view.View>, int, int)) should add only Views focusable in touch mode.

Constant Value: 1 (0x00000001)

public static final int **FOCUS_BACKWARD** Added in API level 1

Use with `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)). Move focus to the previous selectable item.

Constant Value: 1 (0x00000001)

public static final int **FOCUS_DOWN** Added in API level 1

Use with `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)). Move focus down.

Constant Value: 130 (0x00000082)

public static final int **FOCUS_FORWARD** Added in API level 1

Use with `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)). Move focus to the next selectable item.

Constant Value: 2 (0x00000002)

public static final int **FOCUS_LEFT** Added in API level 1

Use with `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)). Move focus to the left.

Constant Value: 17 (0x00000011)

public static final int **FOCUS_RIGHT** Added in API level 1

Use with `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)). Move focus to the right.

Constant Value: 66 (0x00000042)

public static final int **FOCUS_UP** Added in API level 1

Use with `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)). Move focus up.

Constant Value: 33 (0x00000021)

public static final int **GONE** Added in API level 1

This view is invisible, and it doesn't take any space for layout purposes. Use with `setVisibility(int)` (/reference/android/view/View.html#setVisibility(int)) and `android:visibility`. (#attr_android:visibility)

Constant Value: 8 (0x00000008)

public static final int **HAPTIC_FEEDBACK_ENABLED** Added in API level 3

View flag indicating whether this view should have haptic feedback enabled for events such as long presses.

Constant Value: 268435456 (0x10000000)

public static final int **IMPORTANT_FOR_ACCESSIBILITY_AUTO** Added in API level 16

Automatically determine whether a view is important for accessibility.

Constant Value: 0 (0x00000000)

public static final int **IMPORTANT_FOR_ACCESSIBILITY_NO** Added in API level 16

The view is not important for accessibility.

Constant Value: 2 (0x00000002)

public static final int **IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS** Added in API level 19

The view is not important for accessibility, nor are any of its descendant views.

Constant Value: 4 (0x00000004)

public static final int **IMPORTANT_FOR_ACCESSIBILITY_YES**

Added in API level 16

The view is important for accessibility.

Constant Value: 1 (0x00000001)

public static final int **INVISIBLE**

Added in API level 1

This view is invisible, but it still takes up space for layout purposes. Use with `setVisibility(int)` (/reference/android/view/View.html#setVisibility(int)) and `android:visibility`. (#attr_android:visibility)

Constant Value: 4 (0x00000004)

public static final int **KEEP_SCREEN_ON**

Added in API level 1

View flag indicating that the screen should remain on while the window containing this view is visible to the user. This effectively takes care of automatically setting the WindowManager's `FLAG_KEEP_SCREEN_ON` (/reference/android/view/WindowManager.LayoutParams.html#FLAG_KEEP_SCREEN_ON).

Constant Value: 67108864 (0x04000000)

public static final int **LAYER_TYPE_HARDWARE**

Added in API level 11

Indicates that the view has a hardware layer. A hardware layer is backed by a hardware specific texture (generally Frame Buffer Objects or FBO on OpenGL hardware) and causes the view to be rendered using Android's hardware rendering pipeline, but only if hardware acceleration is turned on for the view hierarchy. When hardware acceleration is turned off, hardware layers behave exactly as `software layers` (/reference/android/view/View.html#LAYER_TYPE_SOFTWARE).

A hardware layer is useful to apply a specific color filter and/or blending mode and/or translucency to a view and all its children.

A hardware layer can be used to cache a complex view tree into a texture and reduce the complexity of drawing operations. For instance, when animating a complex view tree with a translation, a hardware layer can be used to render the view tree only once.

A hardware layer can also be used to increase the rendering quality when rotation transformations are applied on a view. It can also be used to prevent potential clipping issues when applying 3D transforms on a view.

See Also
`getLayerType()`
`setLayerType(int, android.graphics.Paint)`
`LAYER_TYPE_NONE`
`LAYER_TYPE_SOFTWARE`

Constant Value: 2 (0x00000002)

public static final int **LAYER_TYPE_NONE**

Added in API level 11

Indicates that the view does not have a layer.

See Also
`getLayerType()`
`setLayerType(int, android.graphics.Paint)`
`LAYER_TYPE_SOFTWARE`
`LAYER_TYPE_HARDWARE`

Constant Value: 0 (0x00000000)

public static final int **LAYER_TYPE_SOFTWARE**

Added in API level 11

Indicates that the view has a software layer. A software layer is backed by a bitmap and causes the view to be rendered using Android's software rendering pipeline, even if hardware acceleration is enabled.

Software layers have various usages:

When the application is not using hardware acceleration, a software layer is useful to apply a specific color filter and/or blending mode and/or translucency to a view and all its children.

When the application is using hardware acceleration, a software layer is useful to render drawing primitives not supported by the hardware accelerated pipeline. It can also be used to cache a complex view tree into a texture and reduce the complexity of drawing operations. For instance, when animating a complex view tree with a translation, a software layer can be used to render the view tree only once.

Software layers should be avoided when the affected view tree updates often. Every update will require to re-render the software layer, which can potentially be slow (particularly when hardware acceleration is turned on since the layer will have to be uploaded into a hardware texture after every update.)

See Also
`getLayerType()`
`setLayerType(int, android.graphics.Paint)`
`LAYER_TYPE_NONE`
`LAYER_TYPE_HARDWARE`

Constant Value: 1 (0x00000001)

public static final int **LAYOUT_DIRECTION_INHERIT**

Added in API level 17

Horizontal layout direction of this view is inherited from its parent. Use with `setLayoutDirection(int)` (/reference/android/view/View.html#setLayoutDirection(int)).

Constant Value: 2 (0x00000002)

public static final int **LAYOUT_DIRECTION_LOCALE**

Added in API level 17

Horizontal layout direction of this view is from deduced from the default language script for the locale. Use with `setLayoutDirection(int)` (/reference/android/view/View.html#setLayoutDirection(int)).

Constant Value: 3 (0x00000003)

public static final int **LAYOUT_DIRECTION_LTR**

Added in API level 17

Horizontal layout direction of this view is from Left to Right. Use with `setLayoutDirection(int)` (/reference/android

/view/View.html#setLayoutDirection(int)).

Constant Value: 0 (0x00000000)

public static final int **LAYOUT_DIRECTION_RTL** Added in API level 17

Horizontal layout direction of this view is from Right to Left. Use with `setLayoutDirection(int)` (/reference/android/view/View.html#setLayoutDirection(int)).

Constant Value: 1 (0x00000001)

public static final int **MEASURED_HEIGHT_STATE_SHIFT** Added in API level 11

Bit shift of `MEASURED_STATE_MASK` (/reference/android/view/View.html#MEASURED_STATE_MASK) to get to the height bits for functions that combine both width and height into a single int, such as `getMeasuredState()` (/reference/android/view/View.html#getMeasuredState()) and the childState argument of `resolveSizeAndState(int, int, int)` (/reference/android/view/View.html#resolveSizeAndState(int, int, int)).

Constant Value: 16 (0x00000010)

public static final int **MEASURED_SIZE_MASK** Added in API level 11

Bits of `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()) and `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()) that provide the actual measured size.

Constant Value: 16777215 (0x00ffffff)

public static final int **MEASURED_STATE_MASK** Added in API level 11

Bits of `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()) and `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()) that provide the additional state bits.

Constant Value: -16777216 (0xff000000)

public static final int **MEASURED_STATE_TOO_SMALL** Added in API level 11

Bit of `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()) and `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()) that indicates the measured size is smaller that the space the view would like to have.

Constant Value: 16777216 (0x01000000)

public static final int **NO_ID** Added in API level 1

Used to mark a View that has no ID.

Constant Value: -1 (0xffffffff)

public static final int **OVER_SCROLL_ALWAYS** Added in API level 9

Always allow a user to over-scroll this view, provided it is a view that can scroll.

See Also
`getOverScrollMode()`
`setOverScrollMode(int)`

Constant Value: 0 (0x00000000)

public static final int **OVER_SCROLL_IF_CONTENT_SCROLLS** Added in API level 9

Allow a user to over-scroll this view only if the content is large enough to meaningfully scroll, provided it is a view that can scroll.

See Also
`getOverScrollMode()`
`setOverScrollMode(int)`

Constant Value: 1 (0x00000001)

public static final int **OVER_SCROLL_NEVER** Added in API level 9

Never allow a user to over-scroll this view.

See Also
`getOverScrollMode()`
`setOverScrollMode(int)`

Constant Value: 2 (0x00000002)

public static final int **SCREEN_STATE_OFF** Added in API level 16

Indicates that the screen has changed state and is now off.

See Also
`onScreenStateChanged(int)`

Constant Value: 0 (0x00000000)

public static final int **SCREEN_STATE_ON** Added in API level 16

Indicates that the screen has changed state and is now on.

See Also
`onScreenStateChanged(int)`

Constant Value: 1 (0x00000001)

public static final int **SCROLLBARS_INSIDE_INSET** Added in API level 1

The scrollbar style to display the scrollbars inside the padded area, increasing the padding of the view. The scrollbars will not overlap the content area of the view.

Constant Value: 16777216 (0x01000000)

public static final int **SCROLLBARS_INSIDE_OVERLAY**

Added in API level 1

The scrollbar style to display the scrollbars inside the content area, without increasing the padding. The scrollbars will be overlaid with translucency on the view's content.

Constant Value: 0 (0x00000000)

public static final int **SCROLLBARS_OUTSIDE_INSET**

Added in API level 1

The scrollbar style to display the scrollbars at the edge of the view, increasing the padding of the view. The scrollbars will only overlap the background, if any.

Constant Value: 50331648 (0x03000000)

public static final int **SCROLLBARS_OUTSIDE_OVERLAY**

Added in API level 1

The scrollbar style to display the scrollbars at the edge of the view, without increasing the padding. The scrollbars will be overlaid with translucency.

Constant Value: 33554432 (0x02000000)

public static final int **SCROLLBAR_POSITION_DEFAULT**

Added in API level 11

Position the scroll bar at the default position as determined by the system.

Constant Value: 0 (0x00000000)

public static final int **SCROLLBAR_POSITION_LEFT**

Added in API level 11

Position the scroll bar along the left edge.

Constant Value: 1 (0x00000001)

public static final int **SCROLLBAR_POSITION_RIGHT**

Added in API level 11

Position the scroll bar along the right edge.

Constant Value: 2 (0x00000002)

public static final int **SOUND_EFFECTS_ENABLED**

Added in API level 1

View flag indicating whether this view should have sound effects enabled for events such as clicking and touching.

Constant Value: 134217728 (0x08000000)

public static final int **STATUS_BAR_HIDDEN**

Added in API level 11

This constant was deprecated in API level 14.
Use `SYSTEM_UI_FLAG_LOW_PROFILE` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LOW_PROFILE) instead.

Constant Value: 1 (0x00000001)

public static final int **STATUS_BAR_VISIBLE**

Added in API level 11

This constant was deprecated in API level 14.
Use `SYSTEM_UI_FLAG_VISIBLE` (/reference/android/view/View.html#SYSTEM_UI_FLAG_VISIBLE) instead.

Constant Value: 0 (0x00000000)

public static final int **SYSTEM_UI_FLAG_FULLSCREEN**

Added in API level 16

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): View has requested to go into the normal fullscreen mode so that its content can take over the screen while still allowing the user to interact with the application.

This has the same visual effect as `WindowManager.LayoutParams.FLAG_FULLSCREEN` (/reference/android/view/WindowManager.LayoutParams.html#FLAG_FULLSCREEN), meaning that non-critical screen decorations (such as the status bar) will be hidden while the user is in the View's window, focusing the experience on that content. Unlike the window flag, if you are using ActionBar in overlay mode with `Window.FEATURE_ACTION_BAR_OVERLAY` (/reference/android/view/Window.html#FEATURE_ACTION_BAR_OVERLAY), then enabling this flag will also hide the action bar.

This approach to going fullscreen is best used over the window flag when it is a transient state -- that is, the application does this at certain points in its user interaction where it wants to allow the user to focus on content, but not as a continuous state. For situations where the application would like to simply stay full screen the entire time (such as a game that wants to take over the screen), the `window flag` (/reference/android/view/WindowManager.LayoutParams.html#FLAG_FULLSCREEN) is usually a better approach. The state set here will be removed by the system in various situations (such as the user moving to another application) like the other system UI states.

When using this flag, the application should provide some easy facility for the user to go out of it. A common example would be in an e-book reader, where tapping on the screen brings back whatever screen and UI decorations that had been hidden while the user was immersed in reading the book.

See Also
`setSystemUiVisibility(int)`

Constant Value: 4 (0x00000004)

public static final int **SYSTEM_UI_FLAG_HIDE_NAVIGATION**

Added in API level 14

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): View has requested that the system navigation be temporarily hidden.

This is an even less obtrusive state than that called for by `SYSTEM_UI_FLAG_LOW_PROFILE` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LOW_PROFILE); on devices that draw essential navigation controls (Home, Back, and the like) on screen, `SYSTEM_UI_FLAG_HIDE_NAVIGATION` will cause those to disappear. This is useful (in conjunction with the `FLAG_FULLSCREEN` (/reference/android/view/WindowManager.LayoutParams.html#FLAG_FULLSCREEN) and `FLAG_LAYOUT_IN_SCREEN` (/reference/android/view/WindowManager.LayoutParams.html#FLAG_LAYOUT_IN_SCREEN) window flags) for displaying content using every last pixel on the display.

There is a limitation: because navigation controls are so important, the least user interaction will cause them to reappear immediately. When this happens, both this flag and `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN) will be cleared automatically, so that both elements reappear at the same time.

See Also
`setSystemUiVisibility(int)`

Constant Value: 2 (0x00000002)

public static final int **SYSTEM_UI_FLAG_IMMERSIVE**

Added in API level 19

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): View would like to remain interactive when hiding the navigation bar with `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION). If this flag is not set, `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION) will be force cleared by the system on any user interaction.

Since this flag is a modifier for `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION), it only has an effect when used in combination with that flag.

Constant Value: 2048 (0x00000800)

public static final int **SYSTEM_UI_FLAG_IMMERSIVE_STICKY**

Added in API level 19

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): View would like to remain interactive when hiding the status bar with `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN) and/or hiding the navigation bar with `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION). Use this flag to create an immersive experience while also hiding the system bars. If this flag is not set, `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION) will be force cleared by the system on any user interaction, and `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN) will be force-cleared by the system if the user swipes from the top of the screen.

When system bars are hidden in immersive mode, they can be revealed temporarily with system gestures, such as swiping from the top of the screen. These transient system bars will overlay app's content, may have some degree of transparency, and will automatically hide after a short timeout.

Since this flag is a modifier for `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN) and `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION), it only has an effect when used in combination with one or both of those flags.

Constant Value: 4096 (0x00001000)

public static final int **SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN**

Added in API level 16

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): View would like its window to be layed out as if it has requested `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN), even if it currently hasn't. This allows it to avoid artifacts when switching in and out of that mode, at the expense that some of its user interface may be covered by screen decorations when they are shown. You can perform layout of your inner UI elements to account for non-fullscreen system UI through the `fitSystemWindows(Rect)` (/reference/android/view/View.html#fitSystemWindows(android.graphics.Rect)) method.

Constant Value: 1024 (0x00000400)

public static final int **SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION**

Added in API level 16

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): View would like its window to be layed out as if it has requested `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION), even if it currently hasn't. This allows it to avoid artifacts when switching in and out of that mode, at the expense that some of its user interface may be covered by screen decorations when they are shown. You can perform layout of your inner UI elements to account for the navigation system UI through the `fitSystemWindows(Rect)` (/reference/android/view/View.html#fitSystemWindows(android.graphics.Rect)) method.

Constant Value: 512 (0x00000200)

public static final int **SYSTEM_UI_FLAG_LAYOUT_STABLE**

Added in API level 16

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): When using other layout flags, we would like a stable view of the content insets given to `fitSystemWindows(Rect)` (/reference/android/view/View.html#fitSystemWindows(android.graphics.Rect)). This means that the insets seen there will always represent the worst case that the application can expect as a continuous state. In the stock Android UI this is the space for the system bar, nav bar, and status bar, but not more transient elements such as an input method. The stable layout your UI sees is based on the system UI modes you can switch to. That is, if you specify `SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN) then you will get a stable layout for changes of the `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN) mode; if you specify `SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN) and `SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION), then you can transition to `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN) and `SYSTEM_UI_FLAG_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_HIDE_NAVIGATION) with a stable layout. (Note that you should avoid using `SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION) by itself.) If you have set the window flag `FLAG_FULLSCREEN` (/reference/android/view/WindowManager.LayoutParams.html#FLAG_FULLSCREEN) to hide the status bar (instead of using `SYSTEM_UI_FLAG_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_FULLSCREEN)), then a hidden status bar will be considered a "stable" state for purposes here. This allows your UI to continually hide the status bar, while still using the system UI flags to hide the action bar while still retaining a stable layout. Note that changing the window fullscreen flag will never provide a stable layout for a clean transition.

If you are using ActionBar in overlay mode with `Window.FEATURE_ACTION_BAR_OVERLAY` (/reference/android/view/Window.html#FEATURE_ACTION_BAR_OVERLAY), this flag will also impact the insets it adds to those given to the application.

Constant Value: 256 (0x00000100)

public static final int **SYSTEM_UI_FLAG_LOW_PROFILE**

Added in API level 14

Flag for `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)): View has requested the system UI to enter an unobtrusive "low profile" mode.

This is for use in games, book readers, video players, or any other "immersive" application where the usual system chrome is deemed too distracting.

In low profile mode, the status bar and/or navigation icons may dim.

See Also
setSystemUiVisibility(int)

Constant Value: 1 (0x00000001)

public static final int **SYSTEM_UI_FLAG_VISIBLE** Added in API level 14

Special constant for setSystemUiVisibility(int) (/reference/android/view/View.html#setSystemUiVisibility(int)): View has requested the system UI (status bar) to be visible (the default).

See Also
setSystemUiVisibility(int)

Constant Value: 0 (0x00000000)

public static final int **SYSTEM_UI_LAYOUT_FLAGS** Added in API level 16

Flags that can impact the layout in relation to system UI.

Constant Value: 1536 (0x00000600)

public static final int **TEXT_ALIGNMENT_CENTER** Added in API level 17

Center the paragraph, e.g. ALIGN_CENTER. Use with setTextAlignment(int) (/reference/android/view/View.html#setTextAlignment(int))

Constant Value: 4 (0x00000004)

public static final int **TEXT_ALIGNMENT_GRAVITY** Added in API level 17

Default for the root view. The gravity determines the text alignment, ALIGN_NORMAL, ALIGN_CENTER, or ALIGN_OPPOSITE, which are relative to each paragraph's text direction. Use with setTextAlignment(int) (/reference/android/view/View.html#setTextAlignment(int))

Constant Value: 1 (0x00000001)

public static final int **TEXT_ALIGNMENT_INHERIT** Added in API level 16

Constant Value: 0 (0x00000000)

public static final int **TEXT_ALIGNMENT_TEXT_END** Added in API level 17

Align to the end of the paragraph, e.g. ALIGN_OPPOSITE. Use with setTextAlignment(int) (/reference/android/view/View.html#setTextAlignment(int))

Constant Value: 3 (0x00000003)

public static final int **TEXT_ALIGNMENT_TEXT_START** Added in API level 17

Align to the start of the paragraph, e.g. ALIGN_NORMAL. Use with setTextAlignment(int) (/reference/android/view/View.html#setTextAlignment(int))

Constant Value: 2 (0x00000002)

public static final int **TEXT_ALIGNMENT_VIEW_END** Added in API level 17

Align to the end of the view, which is ALIGN_RIGHT if the view's resolved layoutDirection is LTR, and ALIGN_LEFT otherwise. Use with setTextAlignment(int) (/reference/android/view/View.html#setTextAlignment(int))

Constant Value: 6 (0x00000006)

public static final int **TEXT_ALIGNMENT_VIEW_START** Added in API level 17

Align to the start of the view, which is ALIGN_LEFT if the view's resolved layoutDirection is LTR, and ALIGN_RIGHT otherwise. Use with setTextAlignment(int) (/reference/android/view/View.html#setTextAlignment(int))

Constant Value: 5 (0x00000005)

public static final int **TEXT_DIRECTION_ANY_RTL** Added in API level 17

Text direction is using "any-RTL" algorithm. The paragraph direction is RTL if it contains any strong RTL character, otherwise it is LTR if it contains any strong LTR characters. If there are neither, the paragraph direction is the view's resolved layout direction.

Constant Value: 2 (0x00000002)

public static final int **TEXT_DIRECTION_FIRST_STRONG** Added in API level 17

Text direction is using "first strong algorithm". The first strong directional character determines the paragraph direction. If there is no strong directional character, the paragraph direction is the view's resolved layout direction.

Constant Value: 1 (0x00000001)

public static final int **TEXT_DIRECTION_INHERIT** Added in API level 17

Text direction is inherited thru ViewGroup (/reference/android/view/ViewGroup.html)

Constant Value: 0 (0x00000000)

public static final int **TEXT_DIRECTION_LOCALE** Added in API level 17

Text direction is coming from the system Locale.

Constant Value: 5 (0x00000005)

public static final int **TEXT_DIRECTION_LTR** Added in API level 17

Text direction is forced to LTR.

Constant Value: 3 (0x00000003)

public static final int **TEXT_DIRECTION_RTL** Added in API level 17

Text direction is forced to RTL.

Constant Value: 4 (0x00000004)

protected static final String **VIEW_LOG_TAG** Added in API level 1

The logging tag used by this class with android.util.Log.

Constant Value: "View"

public static final int **VISIBLE** Added in API level 1

This view is visible. Use with `setVisibility(int)` (/reference/android/view/View.html#setVisibility(int)) and `android:visibility` (#attr_android:visibility).

Constant Value: 0 (0x00000000)

Fields

public static final Property<View, Float> **ALPHA** Added in API level 14

A Property wrapper around the `alpha` functionality handled by the `setAlpha(float)` (/reference/android/view/View.html#setAlpha(float)) and `getAlpha()` (/reference/android/view/View.html#getAlpha()) methods.

protected static final int[] **EMPTY_STATE_SET** Added in API level 1

Indicates the view has no states set. States are used with `Drawable` (/reference/android/graphics/drawable/Drawable.html) to change the drawing of the view depending on its state.

See Also
`Drawable`
`getDrawableState()`

protected static final int[] **ENABLED_FOCUSED_SELECTED_STATE_SET** Added in API level 1

Indicates the view is enabled, focused and selected.

See Also
`ENABLED_STATE_SET`
`FOCUSED_STATE_SET`
`SELECTED_STATE_SET`

protected static final int[] **ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is enabled, focused, selected and its window has the focus.

See Also
`ENABLED_STATE_SET`
`FOCUSED_STATE_SET`
`SELECTED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **ENABLED_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is enabled and has the focus.

See Also
`ENABLED_STATE_SET`
`FOCUSED_STATE_SET`

protected static final int[] **ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is enabled, focused and its window has the focus.

See Also
`ENABLED_STATE_SET`
`FOCUSED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **ENABLED_SELECTED_STATE_SET** Added in API level 1

Indicates the view is enabled and selected.

See Also
`ENABLED_STATE_SET`
`SELECTED_STATE_SET`

protected static final int[] **ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is enabled, selected and its window has the focus.

See Also
`ENABLED_STATE_SET`
`SELECTED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **ENABLED_STATE_SET** Added in API level 1

Indicates the view is enabled. States are used with `Drawable` (/reference/android/graphics/drawable/Drawable.html) to change the drawing of the view depending on its state.

See Also

`Drawable`
`getDrawableState()`

protected static final int[] **ENABLED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is enabled and that its window has focus.

See Also
`ENABLED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **FOCUSED_SELECTED_STATE_SET** Added in API level 1

Indicates the view is focused and selected.

See Also
`FOCUSED_STATE_SET`
`SELECTED_STATE_SET`

protected static final int[] **FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is focused, selected and its window has the focus.

See Also
`FOCUSED_STATE_SET`
`SELECTED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **FOCUSED_STATE_SET** Added in API level 1

Indicates the view is focused. States are used with `Drawable` (/reference/android/graphics/drawable/Drawable.html) to change the drawing of the view depending on its state.

See Also
`Drawable`
`getDrawableState()`

protected static final int[] **FOCUSED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view has the focus and that its window has the focus.

See Also
`FOCUSED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_FOCUSED_SELECTED_STATE_SET** Added in API level 1

Indicates the view is pressed, enabled, focused and selected.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`
`SELECTED_STATE_SET`
`FOCUSED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, enabled, focused, selected and its window has the focus.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`
`SELECTED_STATE_SET`
`FOCUSED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, enabled and focused.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`
`FOCUSED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, enabled, focused and its window has the focus.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`
`FOCUSED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_SELECTED_STATE_SET** Added in API level 1

Indicates the view is pressed, enabled and selected.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`
`SELECTED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, enabled, selected and its window has the focus.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`
`SELECTED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_STATE_SET** Added in API level 1

Indicates the view is pressed and enabled.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`

protected static final int[] **PRESSED_ENABLED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, enabled and its window has the focus.

See Also
`PRESSED_STATE_SET`
`ENABLED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_FOCUSED_SELECTED_STATE_SET** Added in API level 1

Indicates the view is pressed, focused and selected.

See Also
`PRESSED_STATE_SET`
`SELECTED_STATE_SET`
`FOCUSED_STATE_SET`

protected static final int[] **PRESSED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, focused, selected and its window has the focus.

See Also
`PRESSED_STATE_SET`
`FOCUSED_STATE_SET`
`SELECTED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed and focused.

See Also
`PRESSED_STATE_SET`
`FOCUSED_STATE_SET`

protected static final int[] **PRESSED_FOCUSED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, focused and its window has the focus.

See Also
`PRESSED_STATE_SET`
`FOCUSED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_SELECTED_STATE_SET** Added in API level 1

Indicates the view is pressed and selected.

See Also
`PRESSED_STATE_SET`
`SELECTED_STATE_SET`

protected static final int[] **PRESSED_SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed, selected and its window has the focus.

See Also
`PRESSED_STATE_SET`
`SELECTED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

protected static final int[] **PRESSED_STATE_SET** Added in API level 19

Indicates the view is pressed. States are used with `Drawable` (/reference/android/graphics/drawable/Drawable.html) to change the drawing of the view depending on its state.

See Also
`Drawable`
`getDrawableState()`

protected static final int[] **PRESSED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is pressed and its window has the focus.

See Also
`PRESSED_STATE_SET`

WINDOW_FOCUSED_STATE_SET

public static final Property<View, Float> **ROTATION** Added in API level 14

A Property wrapper around the `rotation` functionality handled by the `setRotation(float)` (/reference/android/view/View.html#setRotation(float)) and `getRotation()` (/reference/android/view/View.html#getRotation()) methods.

public static final Property<View, Float> **ROTATION_X** Added in API level 14

A Property wrapper around the `rotationX` functionality handled by the `setRotationX(float)` (/reference/android/view/View.html#setRotationX(float)) and `getRotationX()` (/reference/android/view/View.html#getRotationX()) methods.

public static final Property<View, Float> **ROTATION_Y** Added in API level 14

A Property wrapper around the `rotationY` functionality handled by the `setRotationY(float)` (/reference/android/view/View.html#setRotationY(float)) and `getRotationY()` (/reference/android/view/View.html#getRotationY()) methods.

public static final Property<View, Float> **SCALE_X** Added in API level 14

A Property wrapper around the `scaleX` functionality handled by the `setScaleX(float)` (/reference/android/view/View.html#setScaleX(float)) and `getScaleX()` (/reference/android/view/View.html#getScaleX()) methods.

public static final Property<View, Float> **SCALE_Y** Added in API level 14

A Property wrapper around the `scaleY` functionality handled by the `setScaleY(float)` (/reference/android/view/View.html#setScaleY(float)) and `getScaleY()` (/reference/android/view/View.html#getScaleY()) methods.

protected static final int[] **SELECTED_STATE_SET** Added in API level 1

Indicates the view is selected. States are used with `Drawable` (/reference/android/graphics/drawable/Drawable.html) to change the drawing of the view depending on its state.

See Also
`Drawable`
`getDrawableState()`

protected static final int[] **SELECTED_WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view is selected and that its window has the focus.

See Also
`SELECTED_STATE_SET`
`WINDOW_FOCUSED_STATE_SET`

public static final Property<View, Float> **TRANSLATION_X** Added in API level 14

A Property wrapper around the `translationX` functionality handled by the `setTranslationX(float)` (/reference/android/view/View.html#setTranslationX(float)) and `getTranslationX()` (/reference/android/view/View.html#getTranslationX()) methods.

public static final Property<View, Float> **TRANSLATION_Y** Added in API level 14

A Property wrapper around the `translationY` functionality handled by the `setTranslationY(float)` (/reference/android/view/View.html#setTranslationY(float)) and `getTranslationY()` (/reference/android/view/View.html#getTranslationY()) methods.

protected static final int[] **WINDOW_FOCUSED_STATE_SET** Added in API level 1

Indicates the view's window has focus. States are used with `Drawable` (/reference/android/graphics/drawable/Drawable.html) to change the drawing of the view depending on its state.

See Also
`Drawable`
`getDrawableState()`

public static final Property<View, Float> **X** Added in API level 14

A Property wrapper around the `x` functionality handled by the `setX(float)` (/reference/android/view/View.html#setX(float)) and `getX()` (/reference/android/view/View.html#getX()) methods.

public static final Property<View, Float> **Y** Added in API level 14

A Property wrapper around the `y` functionality handled by the `setY(float)` (/reference/android/view/View.html#setY(float)) and `getY()` (/reference/android/view/View.html#getY()) methods.

Public Constructors

public **View** (Context context) Added in API level 1

Simple constructor to use when creating a view from code.

Parameters

context	The Context the view is running in, through which it can access the current theme, resources, etc.

public **View** (Context context, AttributeSet attrs) Added in API level 1

Constructor that is called when inflating a view from XML. This is called when a view is being constructed from an XML file, supplying attributes that were specified in the XML file. This version uses a default style of 0, so the only attribute values applied are those in the Context's Theme and the given AttributeSet.

The method onFinishInflate() will be called after all children have been added.

Parameters

context	The Context the view is running in, through which it can access the current theme, resources, etc.
attrs	The attributes of the XML tag that is inflating the view.

See Also
`View(Context, AttributeSet, int)`

public **View** (Context context, AttributeSet attrs, int defStyleAttr) Added in API level 1

Perform inflation from XML and apply a class-specific base style. This constructor of View allows subclasses to use their own base style when they are inflating. For example, a Button class's constructor would call this version of the super class constructor and supply `R.attr.buttonStyle` for *defStyle*; this allows the theme's button style to modify all of the base view attributes (in particular its background) as well as the Button class's attributes.

Parameters

context	The Context the view is running in, through which it can access the current theme, resources, etc.
attrs	The attributes of the XML tag that is inflating the view.
defStyleAttr	An attribute in the current theme that contains a reference to a style resource to apply to this view. If 0, no default style will be applied.

See Also
`View(Context, AttributeSet)`

Public Methods

public void **addChildrenForAccessibility** (ArrayList<View> children) Added in API level 16

Adds the children of a given View for accessibility. Since some Views are not important for accessibility the children for accessibility are not necessarily direct children of the view, rather they are the first level of descendants important for accessibility.

Parameters

children	The list of children for accessibility.

public void **addFocusables** (ArrayList<View> views, int direction, int focusableMode) Added in API level 4

Adds any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views. This method adds all focusable views regardless if we are in touch mode or only views focusable in touch mode if we are in touch mode or only views that can take accessibility focus if accessibility is enabeld depending on the focusable mode paramater.

Parameters

views	Focusable views found so far or null if all we are interested is the number of focusables.
direction	The direction of the focus.
focusableMode	The type of focusables to be added.

See Also
`FOCUSABLES_ALL`
`FOCUSABLES_TOUCH_MODE`

public void **addFocusables** (ArrayList<View> views, int direction) Added in API level 1

Add any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views. If we are in touch mode, only add views that are also focusable in touch mode.

Parameters

views	Focusable views found so far
direction	The direction of the focus

public void **addOnAttachStateChangeListener** (View.OnAttachStateChangeListener listener) Added in API level 12

Add a listener for attach state changes. This listener will be called whenever this view is attached or detached from a window. Remove the listener using `removeOnAttachStateChangeListener(OnAttachStateChangeListener)` (/reference/android/view/View.html#removeOnAttachStateChangeListener(android.view.View.OnAttachStateChangeListener)).

Parameters

listener	Listener to attach

See Also
`removeOnAttachStateChangeListener(OnAttachStateChangeListener)`

public void **addOnLayoutChangeListener** (View.OnLayoutChangeListener listener) Added in API level 11

Add a listener that will be called when the bounds of the view change due to layout processing.

Parameters

listener	The listener that will be called when layout bounds change.

public void **addTouchables** (ArrayList<View> views) Added in API level 1

Add any touchable views that are descendants of this view (possibly including this view if it is touchable itself) to views.

Parameters

views	Touchable views found so far

public ViewPropertyAnimator **animate** () Added in API level 12

This method returns a ViewPropertyAnimator object, which can be used to animate specific properties on this View.

Returns
ViewPropertyAnimator The ViewPropertyAnimator associated with this View.

public void **announceForAccessibility** (CharSequence text) Added in API level 16

Convenience method for sending a `TYPE_ANNOUNCEMENT` (/reference/android/view/accessibility/AccessibilityEvent.html#TYPE_ANNOUNCEMENT) `AccessibilityEvent` (/reference/android/view/accessibility/AccessibilityEvent.html) to make an announcement which is related to some sort of a context change for which none of the events representing UI transitions is a good fit. For example, announcing a new page in a book. If accessibility is not enabled this method does nothing.

Parameters

text	The announcement text.

public void **bringToFront** () Added in API level 1

Change the view's z order in the tree, so it's on top of other sibling views. This ordering change may affect layout, if the parent container uses an order-dependent layout scheme (e.g., LinearLayout). Prior to `KITKAT` (/reference/android/os/Build.VERSION_CODES.html#KITKAT) this method should be followed by calls to `requestLayout()` (/reference/android/view/View.html#requestLayout()) and `invalidate()` (/reference/android/view/View.html#invalidate()) on the view's parent to force the parent to redraw with the new child ordering.

See Also
`bringChildToFront(View)`

public void **buildDrawingCache** () Added in API level 1

Calling this method is equivalent to calling `buildDrawingCache(false)`.

See Also
`buildDrawingCache(boolean)`

public void **buildDrawingCache** (boolean autoScale) Added in API level 4

Forces the drawing cache to be built if the drawing cache is invalid.

If you call `buildDrawingCache()` (/reference/android/view/View.html#buildDrawingCache()) manually without calling `setDrawingCacheEnabled(true)` (/reference/android/view/View.html#setDrawingCacheEnabled(boolean)), you should cleanup the cache by calling `destroyDrawingCache()` (/reference/android/view/View.html#destroyDrawingCache()) afterwards.

Note about auto scaling in compatibility mode: When auto scaling is not enabled, this method will create a bitmap of the same size as this view. Because this bitmap will be drawn scaled by the parent ViewGroup, the result on screen might show scaling artifacts. To avoid such artifacts, you should call this method by setting the auto scaling to true. Doing so, however, will generate a bitmap of a different size than the view. This implies that your application must be able to handle this size.

You should avoid calling this method when hardware acceleration is enabled. If you do not need the drawing cache bitmap, calling this method will increase memory usage and cause the view to be rendered in software once, thus negatively impacting performance.

See Also
`getDrawingCache()`
`destroyDrawingCache()`

public void **buildLayer** () Added in API level 12

Forces this view's layer to be created and this view to be rendered into its layer. If this view's layer type is set to `LAYER_TYPE_NONE` (/reference/android/view/View.html#LAYER_TYPE_NONE), invoking this method will have no effect. This method can for instance be used to render a view into its layer before starting an animation. If this view is complex, rendering into the layer before starting the animation will avoid skipping frames.

Throws

IllegalStateException	If this view is not attached to a window

See Also
`setLayerType(int, android.graphics.Paint)`

public boolean **callOnClick** () Added in API level 15

Directly call any attached OnClickListener. Unlike `performClick()` (/reference/android/view/View.html#performClick()), this only calls the listener, and does not do any associated clicking actions like reporting an accessibility event.

Returns
True there was an assigned OnClickListener that was called, false otherwise is returned.

public boolean **canResolveLayoutDirection** () Added in API level 19

Check if layout direction resolution can be done.

Returns
true if layout direction resolution can be done otherwise return false.

public boolean **canResolveTextAlignment** () Added in API level 19

Check if text alignment resolution can be done.

Returns
true if text alignment resolution can be done otherwise return false.

public boolean **canResolveTextDirection** () Added in API level 19

Check if text direction resolution can be done.

Returns
true if text direction resolution can be done otherwise return false.

public boolean **canScrollHorizontally** (int direction) Added in API level 14

Check if this view can be scrolled horizontally in a certain direction.

Parameters

direction	Negative to check scrolling left, positive to check scrolling right.

Returns
true if this view can be scrolled in the specified direction, false otherwise.

public boolean **canScrollVertically** (int direction)

Added in API level 14

Check if this view can be scrolled vertically in a certain direction.

Parameters

direction	Negative to check scrolling up, positive to check scrolling down.

Returns

true if this view can be scrolled in the specified direction, false otherwise.

public void **cancelLongPress** ()

Added in API level 1

Cancels a pending long press. Your subclass can use this if you want the context menu to come up if the user presses and holds at the same place, but you don't want it to come up if they press and then move around enough to cause scrolling.

public final void **cancelPendingInputEvents** ()

Added in API level 19

Cancel any deferred high-level input events that were previously posted to the event queue.

Many views post high-level events such as click handlers to the event queue to run deferred in order to preserve a desired user experience - clearing visible pressed states before executing, etc. This method will abort any events of this nature that are currently in flight.

Custom views that generate their own high-level deferred input events should override `onCancelPendingInputEvents()` (/reference/android/view/View.html#onCancelPendingInputEvents()) and remove those pending events from the queue.

This will also cancel pending input events for any child views.

Note that this may not be sufficient as a debouncing strategy for clicks in all cases. This will not impact newer events posted after this call that may occur as a result of lower-level input events still waiting in the queue. If you are trying to prevent double-submitted events for the duration of some sort of asynchronous transaction you should also take other steps to protect against unexpected double inputs e.g. calling `setEnabled(false)` (/reference/android/view/View.html#setEnabled(boolean)) and re-enabling the view when the transaction completes, tracking already submitted transaction IDs, etc.

public boolean **checkInputConnectionProxy** (View view)

Added in API level 3

Called by the `InputMethodManager` (/reference/android/view/inputmethod/InputMethodManager.html) when a view who is not the current input connection target is trying to make a call on the manager. The default implementation returns false; you can override this to return true for certain views if you are performing InputConnection proxying to them.

Parameters

view	The View that is making the InputMethodManager call.

Returns

Return true to allow the call, false to reject.

public void **clearAnimation** ()

Added in API level 1

Cancels any animations for this view.

public void **clearFocus** ()

Added in API level 1

Called when this view wants to give up focus. If focus is cleared `onFocusChanged(boolean, int, android.graphics.Rect)` (/reference/android/view/View.html#onFocusChanged(boolean, int, android.graphics.Rect)) is called.

Note: When a View clears focus the framework is trying to give focus to the first focusable View from the top. Hence, if this View is the first from the top that can take focus, then all callbacks related to clearing focus will be invoked after wich the framework will give focus to this view.

public static int **combineMeasuredStates** (int curState, int newState)

Added in API level 11

Merge two states as returned by `getMeasuredState()` (/reference/android/view/View.html#getMeasuredState()).

Parameters

curState	The current state as returned from a view or the result of combining multiple views.
newState	The new view state to combine.

Returns

Returns a new integer reflecting the combination of the two states.

public void **computeScroll** ()

Added in API level 1

Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary. This will typically be done if the child is animating a scroll using a `Scroller` (/reference/android/widget/Scroller.html) object.

public AccessibilityNodeInfo **createAccessibilityNodeInfo** ()

Added in API level 14

Returns an `AccessibilityNodeInfo` (/reference/android/view/accessibility/AccessibilityNodeInfo.html) representing this view from the point of view of an `AccessibilityService` (/reference/android/accessibilityservice/AccessibilityService.html). This method is responsible for obtaining an accessibility node info from a pool of reusable instances and calling `onInitializeAccessibilityNodeInfo(AccessibilityNodeInfo)` (/reference/android/view/View.html#onInitializeAccessibilityNodeInfo(android.view.accessibility.AccessibilityNodeInfo)) on this view to initialize the former.

Note: The client is responsible for recycling the obtained instance by calling `recycle()` (/reference/android/view/accessibility/AccessibilityNodeInfo.html#recycle()) to minimize object creation.

Returns

A populated `AccessibilityNodeInfo`.

See Also

`AccessibilityNodeInfo`

public void **createContextMenu** (ContextMenu menu)

Added in API level 1

Show the context menu for this view. It is not safe to hold on to the menu after returning from this method. You should normally not overload this method. Overload `onCreateContextMenu(ContextMenu)` (/reference/android/view/View.html#onCreateContextMenu(android.view.ContextMenu)) or define an `View.OnCreateContextMenuListener` (/reference/android/view/View.OnCreateContextMenuListener.html) to add items to the context menu.

Parameters

menu	The context menu to populate

public void **destroyDrawingCache** ()

Added in API level 1

Frees the resources used by the drawing cache. If you call `buildDrawingCache()` (/reference/android/view/View.html#buildDrawingCache()) manually without calling `setDrawingCacheEnabled(true)` (/reference/android/view/View.html#setDrawingCacheEnabled(boolean)), you should cleanup the cache with this method afterwards.

See Also
`setDrawingCacheEnabled(boolean)`
`buildDrawingCache()`
`getDrawingCache()`

public void **dispatchConfigurationChanged** (Configuration newConfig)

Added in API level 8

Dispatch a notification about a resource configuration change down the view hierarchy. ViewGroups should override to route to their children.

Parameters

newConfig	The new resource configuration.

See Also
`onConfigurationChanged(android.content.res.Configuration)`

public void **dispatchDisplayHint** (int hint)

Added in API level 8

Dispatch a hint about whether this view is displayed. For instance, when a View moves out of the screen, it might receives a display hint indicating the view is not displayed. Applications should not *rely* on this hint as there is no guarantee that they will receive one.

Parameters

hint	A hint about whether or not this view is displayed: `VISIBLE` or `INVISIBLE`.

public boolean **dispatchDragEvent** (DragEvent event)

Added in API level 11

Detects if this View is enabled and has a drag event listener. If both are true, then it calls the drag event listener with the `DragEvent` (/reference/android/view/DragEvent.html) it received. If the drag event listener returns `true`, then dispatchDragEvent() returns `true`.

For all other cases, the method calls the `onDragEvent()` (/reference/android/view/View.html#onDragEvent(android.view.DragEvent)) drag event handler method and returns its result.

This ensures that a drag event is always consumed, even if the View does not have a drag event listener. However, if the View has a listener and the listener returns true, then onDragEvent() is not called.

public boolean **dispatchGenericMotionEvent** (MotionEvent event)

Added in API level 12

Dispatch a generic motion event.

Generic motion events with source class `SOURCE_CLASS_POINTER` (/reference/android/view/InputDevice.html#SOURCE_CLASS_POINTER) are delivered to the view under the pointer. All other generic motion events are delivered to the focused view. Hover events are handled specially and are delivered to `onHoverEvent(MotionEvent)` (/reference/android/view/View.html#onHoverEvent(android.view.MotionEvent)).

Parameters

event	The motion event to be dispatched.

Returns
True if the event was handled by the view, false otherwise.

public boolean **dispatchKeyEvent** (KeyEvent event)

Added in API level 1

Dispatch a key event to the next view on the focus path. This path runs from the top of the view tree down to the currently focused view. If this view has focus, it will dispatch to itself. Otherwise it will dispatch the next node down the focus path. This method also fires any key listeners.

Parameters

event	The key event to be dispatched.

Returns
True if the event was handled, false otherwise.

public boolean **dispatchKeyEventPreIme** (KeyEvent event)

Added in API level 3

Dispatch a key event before it is processed by any input method associated with the view hierarchy. This can be used to intercept key events in special situations before the IME consumes them; a typical example would be handling the BACK key to update the application's UI instead of allowing the IME to see it and close itself.

Parameters

event	The key event to be dispatched.

Returns
True if the event was handled, false otherwise.

public boolean **dispatchKeyShortcutEvent** (KeyEvent event)

Added in API level 1

Dispatches a key shortcut event.

Parameters

event	The key event to be dispatched.

Returns
True if the event was handled by the view, false otherwise.

public boolean **dispatchPopulateAccessibilityEvent** (AccessibilityEvent event)

Added in API level 4

Dispatches an `AccessibilityEvent` (/reference/android/view/accessibility/AccessibilityEvent.html) to the `View` (/reference/android/view/View.html) first and then to its children for adding their text content to the event. Note that the event text is populated in a separate dispatch path since we add to the event not only the text of the source but also the text of all its descendants. A typical implementation will call `onPopulateAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#onPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) on the this view and then call the `dispatchPopulateAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) on each child. Override this method if custom population of the event text content is required.

If an `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate)` (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `dispatchPopulateAccessibilityEvent(View, AccessibilityEvent)` (/reference/android/view/View.AccessibilityDelegate.html#dispatchPopulateAccessibilityEvent(android.view.View, android.view.accessibility.AccessibilityEvent)) is responsible for handling this call.

Note: Accessibility events of certain types are not dispatched for populating the event text via this method. For details refer to `AccessibilityEvent` (/reference/android/view/accessibility/AccessibilityEvent.html).

Parameters

event	The event.

Returns
True if the event population was completed.

public void **dispatchSystemUiVisibilityChanged** (int visibility) Added in API level 11

Dispatch callbacks to `setOnSystemUiVisibilityChangeListener(View.OnSystemUiVisibilityChangeListener)` (/reference/android/view/View.html#setOnSystemUiVisibilityChangeListener(android.view.View.OnSystemUiVisibilityChangeListener)) down the view hierarchy.

public boolean **dispatchTouchEvent** (MotionEvent event) Added in API level 1

Pass the touch screen motion event down to the target view, or this view if it is the target.

Parameters

event	The motion event to be dispatched.

Returns
True if the event was handled by the view, false otherwise.

public boolean **dispatchTrackballEvent** (MotionEvent event) Added in API level 1

Pass a trackball motion event down to the focused view.

Parameters

event	The motion event to be dispatched.

Returns
True if the event was handled by the view, false otherwise.

public boolean **dispatchUnhandledMove** (View focused, int direction) Added in API level 1

This method is the last chance for the focused view and its ancestors to respond to an arrow key. This is called when the focused view did not consume the key internally, nor could the view system find a new view in the requested direction to give focus to.

Parameters

focused	The currently focused view.
direction	The direction focus wants to move. One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT.

Returns
True if the this view consumed this unhandled move.

public void **dispatchWindowFocusChanged** (boolean hasFocus) Added in API level 1

Called when the window containing this view gains or loses window focus. ViewGroups should override to route to their children.

Parameters

hasFocus	True if the window containing this view now has focus, false otherwise.

public void **dispatchWindowSystemUiVisiblityChanged** (int visible) Added in API level 16

Dispatch callbacks to `onWindowSystemUiVisibilityChanged(int)` (/reference/android/view/View.html#onWindowSystemUiVisibilityChanged(int)) down the view hierarchy.

public void **dispatchWindowVisibilityChanged** (int visibility) Added in API level 1

Dispatch a window visibility change down the view hierarchy. ViewGroups should override to route to their children.

Parameters

visibility	The new visibility of the window.

See Also
`onWindowVisibilityChanged(int)`

public void **draw** (Canvas canvas) Added in API level 1

Manually render this view (and all of its children) to the given Canvas. The view must have already done a full layout before this function is called. When implementing a view, implement `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)) instead of overriding this method. If you do need to override this method, call the superclass version.

Parameters

canvas	The Canvas to which the View is rendered.

public View **findFocus** () Added in API level 1

Find the view in the hierarchy rooted at this view that currently has focus.

Returns
The view that currently has focus, or null if no focused view can be found.

public final View **findViewById** (int id) Added in API level 1

Look for a child view with the given id. If this view has the given id, return this view.

Parameters

id	The id to search for.

Returns
The view that has the given id in the hierarchy or null

public final View **findViewWithTag** (Object tag) Added in API level 1

Look for a child view with the given tag. If this view has the given tag, return this view.

Parameters

tag	The tag to search for, using "tag.equals(getTag())".

Returns
The View that has the given tag in the hierarchy or null

public void **findViewsWithText** (ArrayList<View> outViews, CharSequence searched, int flags) Added in API level 14

Finds the Views that contain given text. The containment is case insensitive. The search is performed by either the text that the View renders or the content description that describes the view for accessibility purposes and the view does not render or both. Clients can specify how the search is to be performed via passing the `FIND_VIEWS_WITH_TEXT` (/reference/android/view/View.html#FIND_VIEWS_WITH_TEXT) and `FIND_VIEWS_WITH_CONTENT_DESCRIPTION` (/reference/android/view/View.html#FIND_VIEWS_WITH_CONTENT_DESCRIPTION) flags.

Parameters

outViews	The output list of matching Views.
searched	The text to match against.

See Also
`FIND_VIEWS_WITH_TEXT`
`FIND_VIEWS_WITH_CONTENT_DESCRIPTION`
`setContentDescription(CharSequence)`

public View **focusSearch** (int direction) Added in API level 1

Find the nearest view in the specified direction that can take focus. This does not actually give focus to that view.

Parameters

direction	One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT

Returns
The nearest focusable in the specified direction, or null if none can be found.

public void **forceLayout** () Added in API level 1

Forces this view to be laid out during the next layout pass. This method does not call requestLayout() or forceLayout() on the parent.

public static int **generateViewId** () Added in API level 17

Generate a value suitable for use in `setId(int)` (/reference/android/view/View.html#setId(int)). This value will not collide with ID values generated at build time by aapt for R.id.

Returns
a generated ID value

public int **getAccessibilityLiveRegion** () Added in API level 19

Gets the live region mode for this View.

Related XML Attributes
android:accessibilityLiveRegion

Returns
The live region mode for the view.

See Also
`setAccessibilityLiveRegion(int)`

public AccessibilityNodeProvider **getAccessibilityNodeProvider** () Added in API level 16

Gets the provider for managing a virtual view hierarchy rooted at this View and reported to `AccessibilityService` (/reference/android/accessibilityservice/AccessibilityService.html)s that explore the window content.

If this method returns an instance, this instance is responsible for managing `AccessibilityNodeInfo` (/reference/android/view/accessibility/AccessibilityNodeInfo.html)s describing the virtual sub-tree rooted at this View including the one representing the View itself. Similarly the returned instance is responsible for performing accessibility actions on any virtual view or the root view itself.

If an `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate)` (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `getAccessibilityNodeProvider(View)` (/reference/android/view/View.AccessibilityDelegate.html#getAccessibilityNodeProvider(android.view.View)) is responsible for handling this call.

Returns
The provider.

See Also
`AccessibilityNodeProvider`

public float **getAlpha** () Added in API level 11

The opacity of the view. This is a value from 0 to 1, where 0 means the view is completely transparent and 1 means the view is completely opaque.

By default this is 1.0f.

Returns
The opacity of the view.

public Animation **getAnimation** () Added in API level 1

Get the animation currently associated with this view.

Returns
The animation that is currently playing or scheduled to play for this view.

public IBinder **getApplicationWindowToken** () Added in API level 1

Retrieve a unique token identifying the top-level "real" window of the window that this view is attached to. That is, this is like `getWindowToken()` (/reference/android/view/View.html#getWindowToken()), except if the window this view in is a panel window (attached to another containing window), then the token of the containing window is returned instead.

Returns
Returns the associated window token, either `getWindowToken()` or the containing window's token.

public Drawable **getBackground** () Added in API level 1

Gets the background drawable

Related XML Attributes
android:background

Returns
The drawable used as the background for this view, if any.

See Also
`setBackground(Drawable)`

public int **getBaseline** () Added in API level 1

Return the offset of the widget's text baseline from the widget's top boundary. If this widget does not support baseline alignment, this method returns -1.

Returns
the offset of the baseline within the widget's bounds or -1 if baseline alignment is not supported

public final int **getBottom** () Added in API level 1

Bottom position of this view relative to its parent.

Returns
The bottom of this view, in pixels.

public float **getCameraDistance** () Added in API level 16

Gets the distance along the Z axis from the camera to this view.

Returns
The distance along the Z axis.

See Also
`setCameraDistance(float)`

public Rect **getClipBounds** () Added in API level 18

Returns a copy of the current `clipBounds` (/reference/android/view/View.html#setClipBounds(android.graphics.Rect)).

Returns
A copy of the current clip bounds if clip bounds are set, otherwise null.

public CharSequence **getContentDescription** () Added in API level 4

Gets the `View` (/reference/android/view/View.html) description. It briefly describes the view and is primarily used for accessibility support. Set this property to enable better accessibility support for your application. This is especially true for views that do not have textual representation (For example, ImageButton).

Related XML Attributes
android:contentDescription

Returns
The content description.

public final Context **getContext** () Added in API level 1

Returns the context the view is running in, through which it can access the current theme, resources, etc.

Returns
The view's Context.

public static int **getDefaultSize** (int size, int measureSpec) Added in API level 1

Utility to return a default size. Uses the supplied size if the MeasureSpec imposed no constraints. Will get larger if allowed by the MeasureSpec.

Parameters

size	Default size for this view
measureSpec	Constraints imposed by the parent

Returns
The size this view should be.

public Display **getDisplay** () Added in API level 17

Gets the logical display to which the view's window has been attached.

Returns
The logical display, or null if the view is not currently attached to a window.

public final int[] **getDrawableState** () Added in API level 1

Return an array of resource IDs of the drawable states representing the current state of the view.

Returns
The current drawable state

See Also
`setState(int[])`
`drawableStateChanged()`
`onCreateDrawableState(int)`

public Bitmap **getDrawingCache** (boolean autoScale) Added in API level 4

Returns the bitmap in which this view drawing is cached. The returned bitmap is null when caching is disabled. If caching is enabled and the cache is not ready, this method will create it. Calling `draw(android.graphics.Canvas)` (/reference/android/view/View.html#draw(android.graphics.Canvas)) will not draw from the cache when the cache is enabled. To benefit from the cache, you must request the drawing cache by calling this method and draw it on screen if the returned bitmap is not null.

Note about auto scaling in compatibility mode: When auto scaling is not enabled, this method will create a bitmap of the same size as this view. Because this bitmap will be drawn scaled by the parent ViewGroup, the result on screen might show scaling artifacts. To avoid such artifacts, you should call this method by setting the auto scaling to true. Doing so, however, will generate a bitmap of a different size than the view. This implies that your application must be able to handle this size.

Parameters

autoScale	Indicates whether the generated bitmap should be scaled based on the current density of the screen when the application is in compatibility mode.

Returns
A bitmap representing this view or null if cache is disabled.

See Also
`setDrawingCacheEnabled(boolean)`
`isDrawingCacheEnabled()`
`buildDrawingCache(boolean)`
`destroyDrawingCache()`

public Bitmap **getDrawingCache** () Added in API level 1

Calling this method is equivalent to calling `getDrawingCache(false)`.

Returns
A non-scaled bitmap representing this view or null if cache is disabled.

See Also
`getDrawingCache(boolean)`

public int **getDrawingCacheBackgroundColor** () Added in API level 1

Returns
The background color to used for the drawing cache's bitmap

See Also
`setDrawingCacheBackgroundColor(int)`

public int **getDrawingCacheQuality** () Added in API level 1

Returns the quality of the drawing cache.

Related XML Attributes
android:drawingCacheQuality

Returns
One of `DRAWING_CACHE_QUALITY_AUTO`, `DRAWING_CACHE_QUALITY_LOW`, or `DRAWING_CACHE_QUALITY_HIGH`

See Also
`setDrawingCacheQuality(int)`
`setDrawingCacheEnabled(boolean)`
`isDrawingCacheEnabled()`

public void **getDrawingRect** (Rect outRect) Added in API level 1

Return the visible drawing bounds of your view. Fills in the output rectangle with the values from getScrollX(), getScrollY(), getWidth(), and getHeight(). These bounds do not account for any transformation properties currently set on the view, such as `setScaleX(float)` (/reference/android/view/View.html#setScaleX(float)) or `setRotation(float)` (/reference/android/view/View.html#setRotation(float)).

Parameters

outRect	The (scrolled) drawing bounds of the view.

public long **getDrawingTime** () Added in API level 1

Return the time at which the drawing of the view hierarchy started.

Returns
the drawing start time in milliseconds

public boolean **getFilterTouchesWhenObscured** () Added in API level 9

Gets whether the framework should discard touches when the view's window is obscured by another visible window. Refer to the `View` (/reference/android/view/View.html) security documentation for more details.

Related XML Attributes
android:filterTouchesWhenObscured

Returns
True if touch filtering is enabled.

See Also
`setFilterTouchesWhenObscured(boolean)`

public boolean **getFitsSystemWindows** () Added in API level 16

Check for state of `setFitsSystemWindows(boolean)` (/reference/android/view/View.html#setFitsSystemWindows(boolean)). If this method returns `true`, the default implementation of `fitSystemWindows(Rect)` (/reference/android/view/View.html#fitSystemWindows(android.graphics.Rect)) will be executed.

Related XML Attributes
android:fitsSystemWindows

Returns
`true` if the default implementation of `fitSystemWindows(Rect)` will be executed.

See Also
`setFitsSystemWindows(boolean)`
`fitSystemWindows(Rect)`
`setSystemUiVisibility(int)`

public ArrayList<View> **getFocusables** (int direction) Added in API level 1

Find and return all focusable views that are descendants of this view, possibly including this view if it is focusable itself.

Parameters

direction	The direction of the focus

Returns
A list of focusable views

public void **getFocusedRect** (Rect r) Added in API level 1

When a view has focus and the user navigates away from it, the next view is searched for starting from the rectangle filled in by this method. By default, the rectangle is the `getDrawingRect(android.graphics.Rect)` (/reference/android/view/View.html#getDrawingRect(android.graphics.Rect))) of the view. However, if your view maintains some idea of internal selection, such as a cursor, or a selected row or column, you should override this method and fill in a more specific rectangle.

Parameters

r	The rectangle to fill in, in this view's coordinates.

public boolean **getGlobalVisibleRect** (Rect r, Point globalOffset) Added in API level 1

If some part of this view is not clipped by any of its parents, then return that area in r in global (root) coordinates. To convert r to local coordinates (without taking possible View rotations into account), offset it by -globalOffset (e.g. r.offset(-globalOffset.x, -globalOffset.y)). If the view is completely clipped or translated out, return false.

Parameters

r	If true is returned, r holds the global coordinates of the visible portion of this view.
globalOffset	If true is returned, globalOffset holds the dx,dy between this view and its root. globalOffet may be null.

Returns
true if r is non-empty (i.e. part of the view is visible at the root level.

public final boolean **getGlobalVisibleRect** (Rect r) Added in API level 1

public Handler **getHandler** () Added in API level 1

Returns
A handler associated with the thread running the View. This handler can be used to pump events in the UI events queue.

public final int **getHeight** () Added in API level 1

Return the height of your view.

Returns
The height of your view, in pixels.

public void **getHitRect** (Rect outRect) Added in API level 1

Hit rectangle in parent's coordinates

Parameters

outRect	The hit rectangle of the view.

public int **getHorizontalFadingEdgeLength** () Added in API level 1

Returns the size of the horizontal faded edges used to indicate that more content in this view is visible.

Related XML Attributes
android:fadingEdgeLength

Returns
The size in pixels of the horizontal faded edge or 0 if horizontal faded edges are not enabled for this view.

public int **getId** () Added in API level 1

Returns this view's identifier.

Related XML Attributes
android:id

Returns
a positive integer used to identify the view or NO_ID if the view has no ID

See Also
setId(int)
findViewById(int)

public int **getImportantForAccessibility** ()
Added in API level 16

Gets the mode for determining whether this View is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen.

Related XML Attributes
android:importantForAccessibility

Returns
The mode for determining whether a View is important for accessibility.

See Also
IMPORTANT_FOR_ACCESSIBILITY_YES
IMPORTANT_FOR_ACCESSIBILITY_NO
IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS
IMPORTANT_FOR_ACCESSIBILITY_AUTO

public boolean **getKeepScreenOn** ()
Added in API level 1

Returns whether the screen should remain on, corresponding to the current value of KEEP_SCREEN_ON (/reference/android/view/View.html#KEEP_SCREEN_ON).

Related XML Attributes
android:keepScreenOn

Returns
Returns true if KEEP_SCREEN_ON is set.

See Also
setKeepScreenOn(boolean)

public KeyEvent.DispatcherState **getKeyDispatcherState** ()
Added in API level 5

Return the global KeyEvent.DispatcherState (/reference/android/view/KeyEvent.DispatcherState.html) for this view's window. Returns null if the view is not currently attached to the window. Normally you will not need to use this directly, but just use the standard high-level event callbacks like onKeyDown(int, KeyEvent) (/reference/android/view/View.html#onKeyDown(int, android.view.KeyEvent)).

public int **getLabelFor** ()
Added in API level 17

Gets the id of a view for which this view serves as a label for accessibility purposes.

Returns
The labeled view id.

public int **getLayerType** ()
Added in API level 11

Indicates what type of layer is currently associated with this view. By default a view does not have a layer, and the layer type is LAYER_TYPE_NONE (/reference/android/view/View.html#LAYER_TYPE_NONE). Refer to the documentation of setLayerType(int, android.graphics.Paint) (/reference/android/view/View.html#setLayerType(int, android.graphics.Paint)) for more information on the different types of layers.

Returns
LAYER_TYPE_NONE, LAYER_TYPE_SOFTWARE or LAYER_TYPE_HARDWARE

See Also
setLayerType(int, android.graphics.Paint)
buildLayer()
LAYER_TYPE_NONE
LAYER_TYPE_SOFTWARE
LAYER_TYPE_HARDWARE

public int **getLayoutDirection** ()
Added in API level 17

Returns the resolved layout direction for this view.

Related XML Attributes
android:layoutDirection

Returns
LAYOUT_DIRECTION_RTL if the layout direction is RTL or returns LAYOUT_DIRECTION_LTR if the layout direction is not RTL. For compatibility, this will return LAYOUT_DIRECTION_LTR if API version is lower than JELLY_BEAN_MR1.

public ViewGroup.LayoutParams **getLayoutParams** ()
Added in API level 1

Get the LayoutParams associated with this view. All views should have layout parameters. These supply parameters to the *parent* of this view specifying how it should be arranged. There are many subclasses of ViewGroup.LayoutParams, and these correspond to the different subclasses of ViewGroup that are responsible for arranging their children. This method may return null if this View is not attached to a parent ViewGroup or setLayoutParams(android.view.ViewGroup.LayoutParams) (/reference/android/view/View.html#setLayoutParams(android.view.ViewGroup.LayoutParams)) was not invoked successfully. When a View is attached to a parent ViewGroup, this method must not return null.

Returns
The LayoutParams associated with this view, or null if no parameters have been set yet

public final int **getLeft** ()
Added in API level 1

Left position of this view relative to its parent.

Returns
The left edge of this view, in pixels.

public final boolean **getLocalVisibleRect** (Rect r) Added in API level 1

public void **getLocationInWindow** (int[] location) Added in API level 1

Computes the coordinates of this view in its window. The argument must be an array of two integers. After the method returns, the array contains the x and y location in that order.

Parameters

location	an array of two integers in which to hold the coordinates

public void **getLocationOnScreen** (int[] location) Added in API level 1

Computes the coordinates of this view on the screen. The argument must be an array of two integers. After the method returns, the array contains the x and y location in that order.

Parameters

location	an array of two integers in which to hold the coordinates

public Matrix **getMatrix** () Added in API level 11

The transform matrix of this view, which is calculated based on the current roation, scale, and pivot properties.

Returns

The current transform matrix for the view

See Also

`getRotation()`
`getScaleX()`
`getScaleY()`
`getPivotX()`
`getPivotY()`

public final int **getMeasuredHeight** () Added in API level 1

Like `getMeasuredHeightAndState()` (/reference/android/view/View.html#getMeasuredHeightAndState()), but only returns the raw width component (that is the result is masked by `MEASURED_SIZE_MASK` (/reference/android/view/View.html#MEASURED_SIZE_MASK)).

Returns

The raw measured height of this view.

public final int **getMeasuredHeightAndState** () Added in API level 11

Return the full height measurement information for this view as computed by the most recent call to `measure(int, int)` (/reference/android/view/View.html#measure(int, int)). This result is a bit mask as defined by `MEASURED_SIZE_MASK` (/reference/android/view/View.html#MEASURED_SIZE_MASK) and `MEASURED_STATE_TOO_SMALL` (/reference/android/view/View.html#MEASURED_STATE_TOO_SMALL). This should be used during measurement and layout calculations only. Use `getHeight()` (/reference/android/view/View.html#getHeight()) to see how wide a view is after layout.

Returns

The measured width of this view as a bit mask.

public final int **getMeasuredState** () Added in API level 11

Return only the state bits of `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()) and `getMeasuredHeightAndState()` (/reference/android/view/View.html#getMeasuredHeightAndState()), combined into one integer. The width component is in the regular bits `MEASURED_STATE_MASK` (/reference/android/view/View.html#MEASURED_STATE_MASK) and the height component is at the shifted bits `MEASURED_HEIGHT_STATE_SHIFT` (/reference/android/view/View.html#MEASURED_HEIGHT_STATE_SHIFT)>>`MEASURED_STATE_MASK` (/reference/android/view/View.html#MEASURED_STATE_MASK).

public final int **getMeasuredWidth** () Added in API level 1

Like `getMeasuredWidthAndState()` (/reference/android/view/View.html#getMeasuredWidthAndState()), but only returns the raw width component (that is the result is masked by `MEASURED_SIZE_MASK` (/reference/android/view/View.html#MEASURED_SIZE_MASK)).

Returns

The raw measured width of this view.

public final int **getMeasuredWidthAndState** () Added in API level 11

Return the full width measurement information for this view as computed by the most recent call to `measure(int, int)` (/reference/android/view/View.html#measure(int, int)). This result is a bit mask as defined by `MEASURED_SIZE_MASK` (/reference/android/view/View.html#MEASURED_SIZE_MASK) and `MEASURED_STATE_TOO_SMALL` (/reference/android/view/View.html#MEASURED_STATE_TOO_SMALL). This should be used during measurement and layout calculations only. Use `getWidth()` (/reference/android/view/View.html#getWidth()) to see how wide a view is after layout.

Returns

The measured width of this view as a bit mask.

public int **getMinimumHeight** () Added in API level 16

Returns the minimum height of the view.

Related XML Attributes

android:minHeight

Returns

the minimum height the view will try to be.

See Also

`setMinimumHeight(int)`

public int **getMinimumWidth** () Added in API level 16

Returns the minimum width of the view.

Related XML Attributes
android:minWidth

Returns
the minimum width the view will try to be.

See Also
`setMinimumWidth(int)`

public int **getNextFocusDownId** ()

Added in API level 1

Gets the id of the view to use when the next focus is `FOCUS_DOWN` (/reference/android/view/View.html#FOCUS_DOWN).

Related XML Attributes
android:nextFocusDown

Returns
The next focus ID, or `NO_ID` if the framework should decide automatically.

public int **getNextFocusForwardId** ()

Added in API level 11

Gets the id of the view to use when the next focus is `FOCUS_FORWARD` (/reference/android/view/View.html#FOCUS_FORWARD).

Related XML Attributes
android:nextFocusForward

Returns
The next focus ID, or `NO_ID` if the framework should decide automatically.

public int **getNextFocusLeftId** ()

Added in API level 1

Gets the id of the view to use when the next focus is `FOCUS_LEFT` (/reference/android/view/View.html#FOCUS_LEFT).

Related XML Attributes
android:nextFocusLeft

Returns
The next focus ID, or `NO_ID` if the framework should decide automatically.

public int **getNextFocusRightId** ()

Added in API level 1

Gets the id of the view to use when the next focus is `FOCUS_RIGHT` (/reference/android/view/View.html#FOCUS_RIGHT).

Related XML Attributes
android:nextFocusRight

Returns
The next focus ID, or `NO_ID` if the framework should decide automatically.

public int **getNextFocusUpId** ()

Added in API level 1

Gets the id of the view to use when the next focus is `FOCUS_UP` (/reference/android/view/View.html#FOCUS_UP).

Related XML Attributes
android:nextFocusUp

Returns
The next focus ID, or `NO_ID` if the framework should decide automatically.

public View.OnFocusChangeListener **getOnFocusChangeListener** ()

Added in API level 1

Returns the focus-change callback registered for this view.

Returns
The callback, or null if one is not registered.

public int **getOverScrollMode** ()

Added in API level 9

Returns the over-scroll mode for this view. The result will be one of `OVER_SCROLL_ALWAYS` (/reference/android/view/View.html#OVER_SCROLL_ALWAYS) (default), `OVER_SCROLL_IF_CONTENT_SCROLLS` (/reference/android/view/View.html#OVER_SCROLL_IF_CONTENT_SCROLLS) (allow over-scrolling only if the view content is larger than the container), or `OVER_SCROLL_NEVER` (/reference/android/view/View.html#OVER_SCROLL_NEVER).

Returns
This view's over-scroll mode.

public ViewOverlay **getOverlay** ()

Added in API level 18

Returns the overlay for this view, creating it if it does not yet exist. Adding drawables to the overlay will cause them to be displayed whenever the view itself is redrawn. Objects in the overlay should be actively managed: remove them when they should not be displayed anymore. The overlay will always have the same size as its host view.

Note: Overlays do not currently work correctly with `SurfaceView` (/reference/android/view/SurfaceView.html) or `TextureView` (/reference/android/view/TextureView.html); contents in overlays for these types of views may not display correctly.

Returns
The ViewOverlay object for this view.

See Also
`ViewOverlay`

public int **getPaddingBottom** ()

Added in API level 1

Returns the bottom padding of this view. If there are inset and enabled scrollbars, this value may include the space required to display the scrollbars as well.

Returns
the bottom padding in pixels

public int **getPaddingEnd** () Added in API level 17

Returns the end padding of this view depending on its resolved layout direction. If there are inset and enabled scrollbars, this value may include the space required to display the scrollbars as well.

Returns
the end padding in pixels

public int **getPaddingLeft** () Added in API level 1

Returns the left padding of this view. If there are inset and enabled scrollbars, this value may include the space required to display the scrollbars as well.

Returns
the left padding in pixels

public int **getPaddingRight** () Added in API level 1

Returns the right padding of this view. If there are inset and enabled scrollbars, this value may include the space required to display the scrollbars as well.

Returns
the right padding in pixels

public int **getPaddingStart** () Added in API level 17

Returns the start padding of this view depending on its resolved layout direction. If there are inset and enabled scrollbars, this value may include the space required to display the scrollbars as well.

Returns
the start padding in pixels

public int **getPaddingTop** () Added in API level 1

Returns the top padding of this view.

Returns
the top padding in pixels

public final ViewParent **getParent** () Added in API level 1

Gets the parent of this view. Note that the parent is a ViewParent and not necessarily a View.

Returns
Parent of this view.

public ViewParent **getParentForAccessibility** () Added in API level 16

Gets the parent for accessibility purposes. Note that the parent for accessibility is not necessary the immediate parent. It is the first predecessor that is important for accessibility.

Returns
The parent for accessibility purposes.

public float **getPivotX** () Added in API level 11

The x location of the point around which the view is `rotated` (/reference/android/view/View.html#setRotation(float)) and `scaled` (/reference/android/view/View.html#setScaleX(float)).

Related XML Attributes
android:transformPivotX

Returns
The x location of the pivot point.

See Also
`getRotation()`
`getScaleX()`
`getScaleY()`
`getPivotY()`

public float **getPivotY** () Added in API level 11

The y location of the point around which the view is `rotated` (/reference/android/view/View.html#setRotation(float)) and `scaled` (/reference/android/view/View.html#setScaleY(float)).

Related XML Attributes
android:transformPivotY

Returns
The y location of the pivot point.

See Also
`getRotation()`
`getScaleX()`
`getScaleY()`
`getPivotY()`

public Resources **getResources** () Added in API level 1

Returns the resources associated with this view.

Returns
Resources object.

public final int **getRight** () Added in API level 1

Right position of this view relative to its parent.

Returns
The right edge of this view, in pixels.

public View **getRootView** () Added in API level 1

Finds the topmost view in the current view hierarchy.

Returns
the topmost view containing this view

public float **getRotation** () Added in API level 11

The degrees that the view is rotated around the pivot point.

Returns
The degrees of rotation.

See Also
`setRotation(float)`
`getPivotX()`
`getPivotY()`

public float **getRotationX** () Added in API level 11

The degrees that the view is rotated around the horizontal axis through the pivot point.

Returns
The degrees of X rotation.

See Also
`getPivotX()`
`getPivotY()`
`setRotationX(float)`

public float **getRotationY** () Added in API level 11

The degrees that the view is rotated around the vertical axis through the pivot point.

Returns
The degrees of Y rotation.

See Also
`getPivotX()`
`getPivotY()`
`setRotationY(float)`

public float **getScaleX** () Added in API level 11

The amount that the view is scaled in x around the pivot point, as a proportion of the view's unscaled width. A value of 1, the default, means that no scaling is applied.

By default, this is 1.0f.

Returns
The scaling factor.

See Also
`getPivotX()`
`getPivotY()`

public float **getScaleY** () Added in API level 11

The amount that the view is scaled in y around the pivot point, as a proportion of the view's unscaled height. A value of 1, the default, means that no scaling is applied.

By default, this is 1.0f.

Returns
The scaling factor.

See Also
`getPivotX()`
`getPivotY()`

public int **getScrollBarDefaultDelayBeforeFade** () Added in API level 16

Returns the delay before scrollbars fade.

Related XML Attributes
android:scrollbarDefaultDelayBeforeFade

Returns
the delay before scrollbars fade

public int **getScrollBarFadeDuration** () Added in API level 16

Returns the scrollbar fade duration.

Related XML Attributes
android:scrollbarFadeDuration

Returns
the scrollbar fade duration

public int **getScrollBarSize** () Added in API level 16

Returns the scrollbar size.

Related XML Attributes

android:scrollbarSize

Returns
the scrollbar size

public int **getScrollBarStyle** () Added in API level 1

Returns the current scrollbar style.

Related XML Attributes
android:scrollbarStyle

Returns
the current scrollbar style

See Also
`SCROLLBARS_INSIDE_OVERLAY`
`SCROLLBARS_INSIDE_INSET`
`SCROLLBARS_OUTSIDE_OVERLAY`
`SCROLLBARS_OUTSIDE_INSET`

public final int **getScrollX** () Added in API level 1

Return the scrolled left position of this view. This is the left edge of the displayed part of your view. You do not need to draw any pixels farther left, since those are outside of the frame of your view on screen.

Returns
The left edge of the displayed part of your view, in pixels.

public final int **getScrollY** () Added in API level 1

Return the scrolled top position of this view. This is the top edge of the displayed part of your view. You do not need to draw any pixels above it, since those are outside of the frame of your view on screen.

Returns
The top edge of the displayed part of your view, in pixels.

public int **getSolidColor** () Added in API level 1

Override this if your view is known to always be drawn on top of a solid color background, and needs to draw fading edges. Returning a non-zero color enables the view system to optimize the drawing of the fading edges. If you do return a non-zero color, the alpha should be set to 0xFF.

Returns
The known solid color background for this view, or 0 if the color may vary

See Also
`setVerticalFadingEdgeEnabled(boolean)`
`setHorizontalFadingEdgeEnabled(boolean)`

public int **getSystemUiVisibility** () Added in API level 11

Returns the last `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)) that this view has requested.

Returns
Bitwise-or of flags `SYSTEM_UI_FLAG_LOW_PROFILE`, `SYSTEM_UI_FLAG_HIDE_NAVIGATION`, `SYSTEM_UI_FLAG_FULLSCREEN`, `SYSTEM_UI_FLAG_LAYOUT_STABLE`, `SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION`, `SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN`, `SYSTEM_UI_FLAG_IMMERSIVE`, and `SYSTEM_UI_FLAG_IMMERSIVE_STICKY`.

public Object **getTag** (int key) Added in API level 4

Returns the tag associated with this view and the specified key.

Parameters

key	The key identifying the tag

Returns
the Object stored in this view as a tag

See Also
`setTag(int, Object)`
`getTag()`

public Object **getTag** () Added in API level 1

Returns this view's tag.

Returns
the Object stored in this view as a tag

See Also
`setTag(Object)`
`getTag(int)`

public int **getTextAlignment** () Added in API level 17

Return the resolved text alignment.

Related XML Attributes
android:textAlignment

Returns
the resolved text alignment. Returns one of: `TEXT_ALIGNMENT_GRAVITY`, `TEXT_ALIGNMENT_CENTER`, `TEXT_ALIGNMENT_TEXT_START`, `TEXT_ALIGNMENT_TEXT_END`, `TEXT_ALIGNMENT_VIEW_START`, `TEXT_ALIGNMENT_VIEW_END`

public int **getTextDirection** () Added in API level 17

Return the resolved text direction.

Related XML Attributes

android:textDirection

Returns
the resolved text direction. Returns one of: `TEXT_DIRECTION_FIRST_STRONG` `TEXT_DIRECTION_ANY_RTL`, `TEXT_DIRECTION_LTR`, `TEXT_DIRECTION_RTL`, `TEXT_DIRECTION_LOCALE`

public final int **getTop** () Added in API level 1

Top position of this view relative to its parent.

Returns
The top of this view, in pixels.

public TouchDelegate **getTouchDelegate** () Added in API level 1

Gets the TouchDelegate for this View.

public ArrayList<View> **getTouchables** () Added in API level 1

Find and return all touchable views that are descendants of this view, possibly including this view if it is touchable itself.

Returns
A list of touchable views

public float **getTranslationX** () Added in API level 11

The horizontal location of this view relative to its `left` (/reference/android/view/View.html#getLeft()) position. This position is post-layout, in addition to wherever the object's layout placed it.

Returns
The horizontal position of this view relative to its left position, in pixels.

public float **getTranslationY** () Added in API level 11

The vertical location of this view relative to its `top` (/reference/android/view/View.html#getTop()) position. This position is post-layout, in addition to wherever the object's layout placed it.

Returns
The vertical position of this view relative to its top position, in pixels.

public int **getVerticalFadingEdgeLength** () Added in API level 1

Returns the size of the vertical faded edges used to indicate that more content in this view is visible.

Related XML Attributes
android:fadingEdgeLength

Returns
The size in pixels of the vertical faded edge or 0 if vertical faded edges are not enabled for this view.

public int **getVerticalScrollbarPosition** () Added in API level 11

Returns
The position where the vertical scroll bar will show, if applicable.

See Also
`setVerticalScrollbarPosition(int)`

public int **getVerticalScrollbarWidth** () Added in API level 1

Returns the width of the vertical scrollbar.

Returns
The width in pixels of the vertical scrollbar or 0 if there is no vertical scrollbar.

public ViewTreeObserver **getViewTreeObserver** () Added in API level 1

Returns the ViewTreeObserver for this view's hierarchy. The view tree observer can be used to get notifications when global events, like layout, happen. The returned ViewTreeObserver observer is not guaranteed to remain valid for the lifetime of this View. If the caller of this method keeps a long-lived reference to ViewTreeObserver, it should always check for the return value of `isAlive()` (/reference/android/view/ViewTreeObserver.html#isAlive()).

Returns
The ViewTreeObserver for this view's hierarchy.

public int **getVisibility** () Added in API level 1

Returns the visibility status for this view.

Related XML Attributes
android:visibility

Returns
One of `VISIBLE`, `INVISIBLE`, or `GONE`.

public final int **getWidth** () Added in API level 1

Return the width of the your view.

Returns
The width of your view, in pixels.

public WindowId **getWindowId** () Added in API level 18

Retrieve the `WindowId` (/reference/android/view/WindowId.html) for the window this view is currently attached to.

public int **getWindowSystemUiVisibility** () Added in API level 16

Returns the current system UI visibility that is currently set for the entire window. This is the combination of the `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)) values supplied by all of the views in the window.

public IBinder **getWindowToken** () Added in API level 1

Retrieve a unique token identifying the window this view is attached to.

Returns
Return the window's token for use in `WindowManager.LayoutParams.token`.

public int **getWindowVisibility** () Added in API level 1

Returns the current visibility of the window this view is attached to (either `GONE` (/reference/android/view/View.html#GONE), `INVISIBLE` (/reference/android/view/View.html#INVISIBLE), or `VISIBLE` (/reference/android/view/View.html#VISIBLE)).

Returns
Returns the current visibility of the view's window.

public void **getWindowVisibleDisplayFrame** (Rect outRect) Added in API level 3

Retrieve the overall visible display size in which the window this view is attached to has been positioned in. This takes into account screen decorations above the window, for both cases where the window itself is being position inside of them or the window is being placed under then and covered insets are used for the window to position its content inside. In effect, this tells you the available area where content can be placed and remain visible to users.

This function requires an IPC back to the window manager to retrieve the requested information, so should not be used in performance critical code like drawing.

Parameters

outRect	Filled in with the visible display frame. If the view is not attached to a window, this is simply the raw display size.

public float **getX** () Added in API level 11

The visual x position of this view, in pixels. This is equivalent to the `translationX` (/reference/android/view/View.html#setTranslationX(float)) property plus the current `left` (/reference/android/view/View.html#getLeft()) property.

Returns
The visual x position of this view, in pixels.

public float **getY** () Added in API level 11

The visual y position of this view, in pixels. This is equivalent to the `translationY` (/reference/android/view/View.html#setTranslationY(float)) property plus the current `top` (/reference/android/view/View.html#getTop()) property.

Returns
The visual y position of this view, in pixels.

public boolean **hasFocus** () Added in API level 1

Returns true if this view has focus iteself, or is the ancestor of the view that has focus.

Returns
True if this view has or contains focus, false otherwise.

public boolean **hasFocusable** () Added in API level 1

Returns true if this view is focusable or if it contains a reachable View for which `hasFocusable()` (/reference/android/view/View.html#hasFocusable()) returns true. A "reachable hasFocusable()" is a View whose parents do not block descendants focus. Only `VISIBLE` (/reference/android/view/View.html#VISIBLE) views are considered focusable.

Returns
True if the view is focusable or if the view contains a focusable View, false otherwise.

See Also
`FOCUS_BLOCK_DESCENDANTS`

public boolean **hasOnClickListeners** () Added in API level 15

Return whether this view has an attached OnClickListener. Returns true if there is a listener, false if there is none.

public boolean **hasOverlappingRendering** () Added in API level 16

Returns whether this View has content which overlaps.

This function, intended to be overridden by specific View types, is an optimization when alpha is set on a view. If rendering overlaps in a view with alpha < 1, that view is drawn to an offscreen buffer and then composited into place, which can be expensive. If the view has no overlapping rendering, the view can draw each primitive with the appropriate alpha value directly. An example of overlapping rendering is a TextView with a background image, such as a Button. An example of non-overlapping rendering is a TextView with no background, or an ImageView with only the foreground image. The default implementation returns true; subclasses should override if they have cases which can be optimized.

The current implementation of the saveLayer and saveLayerAlpha methods in `Canvas` (/reference/android/graphics/Canvas.html) necessitates that a View return true if it uses the methods internally without passing the `CLIP_TO_LAYER_SAVE_FLAG` (/reference/android/graphics/Canvas.html#CLIP_TO_LAYER_SAVE_FLAG).

Returns
true if the content in this view might overlap, false otherwise.

public boolean **hasTransientState** () Added in API level 16

Indicates whether the view is currently tracking transient state that the app should not need to concern itself with saving and restoring, but that the framework should take special note to preserve when possible.

A view with transient state cannot be trivially rebound from an external data source, such as an adapter binding item views in a list. This may be because the view is performing an animation, tracking user selection of content, or similar.

Returns
true if the view has transient state

public boolean **hasWindowFocus** ()
Added in API level 1

Returns true if this view is in a window that currently has window focus. Note that this is not the same as the view itself having focus.

Returns
True if this view is in a window that currently has window focus.

public static View **inflate** (Context context, int resource, ViewGroup root)
Added in API level 1

Inflate a view from an XML resource. This convenience method wraps the `LayoutInflater` (/reference/android/view/LayoutInflater.html) class, which provides a full range of options for view inflation.

Parameters

context	The Context object for your activity or application.
resource	The resource ID to inflate
root	A view group that will be the parent. Used to properly inflate the layout_* parameters.

See Also
`LayoutInflater`

public void **invalidate** (Rect dirty)
Added in API level 1

Mark the area defined by dirty as needing to be drawn. If the view is visible, `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)) will be called at some point in the future. This must be called from a UI thread. To call from a non-UI thread, call `postInvalidate()` (/reference/android/view/View.html#postInvalidate()). WARNING: This method is destructive to dirty.

Parameters

dirty	the rectangle representing the bounds of the dirty region

public void **invalidate** (int l, int t, int r, int b)
Added in API level 1

Mark the area defined by the rect (l,t,r,b) as needing to be drawn. The coordinates of the dirty rect are relative to the view. If the view is visible, `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)) will be called at some point in the future. This must be called from a UI thread. To call from a non-UI thread, call `postInvalidate()` (/reference/android/view/View.html#postInvalidate()).

Parameters

l	the left position of the dirty region
t	the top position of the dirty region
r	the right position of the dirty region
b	the bottom position of the dirty region

public void **invalidate** ()
Added in API level 1

Invalidate the whole view. If the view is visible, `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)) will be called at some point in the future. This must be called from a UI thread. To call from a non-UI thread, call `postInvalidate()` (/reference/android/view/View.html#postInvalidate()).

public void **invalidateDrawable** (Drawable drawable)
Added in API level 1

Invalidates the specified Drawable.

Parameters

drawable	the drawable to invalidate

public boolean **isActivated** ()
Added in API level 11

Indicates the activation state of this view.

Returns
true if the view is activated, false otherwise

public boolean **isAttachedToWindow** ()
Added in API level 19

Returns true if this view is currently attached to a window.

public boolean **isClickable** ()
Added in API level 1

Indicates whether this view reacts to click events or not.

Related XML Attributes
android:clickable

Returns
true if the view is clickable, false otherwise

See Also
`setClickable(boolean)`

public boolean **isDirty** ()
Added in API level 11

True if this view has changed since the last time being drawn.

Returns
The dirty state of this view.

public boolean **isDrawingCacheEnabled** ()
Added in API level 1

Indicates whether the drawing cache is enabled for this view.

Returns
true if the drawing cache is enabled

See Also
`setDrawingCacheEnabled(boolean)`
`getDrawingCache()`

public boolean **isDuplicateParentStateEnabled** () Added in API level 1

Indicates whether this duplicates its drawable state from its parent.

Returns
True if this view's drawable state is duplicated from the parent, false otherwise

See Also
`getDrawableState()`
`setDuplicateParentStateEnabled(boolean)`

public boolean **isEnabled** () Added in API level 1

Returns the enabled status for this view. The interpretation of the enabled state varies by subclass.

Returns
True if this view is enabled, false otherwise.

public final boolean **isFocusable** () Added in API level 1

Returns whether this View is able to take focus.

Related XML Attributes
android:focusable

Returns
True if this view can take focus, or false otherwise.

public final boolean **isFocusableInTouchMode** () Added in API level 1

When a view is focusable, it may not want to take focus when in touch mode. For example, a button would like focus when the user is navigating via a D-pad so that the user can click on it, but once the user starts touching the screen, the button shouldn't take focus

Related XML Attributes
android:focusableInTouchMode

Returns
Whether the view is focusable in touch mode.

public boolean **isFocused** () Added in API level 1

Returns true if this view has focus

Returns
True if this view has focus, false otherwise.

public boolean **isHapticFeedbackEnabled** () Added in API level 3

Related XML Attributes
android:hapticFeedbackEnabled

Returns
whether this view should have haptic feedback enabled for events long presses.

See Also
`setHapticFeedbackEnabled(boolean)`
`performHapticFeedback(int)`

public boolean **isHardwareAccelerated** () Added in API level 11

Indicates whether this view is attached to a hardware accelerated window or not.

Even if this method returns true, it does not mean that every call to `draw(android.graphics.Canvas)` (/reference/android/view/View.html#draw(android.graphics.Canvas)) will be made with an hardware accelerated `Canvas` (/reference/android/graphics/Canvas.html). For instance, if this view is drawn onto an offscreen `Bitmap` (/reference/android/graphics/Bitmap.html) and its window is hardware accelerated, `isHardwareAccelerated()` (/reference/android/graphics/Canvas.html#isHardwareAccelerated()) will likely return false, and this method will return true.

Returns
True if the view is attached to a window and the window is hardware accelerated; false in any other case.

public boolean **isHorizontalFadingEdgeEnabled** () Added in API level 1

Indicate whether the horizontal edges are faded when the view is scrolled horizontally.

Related XML Attributes
android:requiresFadingEdge

Returns
true if the horizontal edges should are faded on scroll, false otherwise

See Also
`setHorizontalFadingEdgeEnabled(boolean)`

public boolean **isHorizontalScrollBarEnabled** () Added in API level 1

Indicate whether the horizontal scrollbar should be drawn or not. The scrollbar is not drawn by default.

Returns

true if the horizontal scrollbar should be painted, false otherwise

See Also

`setHorizontalScrollBarEnabled(boolean)`

public boolean **isHovered** () Added in API level 14

Returns true if the view is currently hovered.

Returns

True if the view is currently hovered.

See Also

`setHovered(boolean)`
`onHoverChanged(boolean)`

public boolean **isInEditMode** () Added in API level 3

Indicates whether this View is currently in edit mode. A View is usually in edit mode when displayed within a developer tool. For instance, if this View is being drawn by a visual user interface builder, this method should return true. Subclasses should check the return value of this method to provide different behaviors if their normal behavior might interfere with the host environment. For instance: the class spawns a thread in its constructor, the drawing code relies on device-specific features, etc. This method is usually checked in the drawing code of custom widgets.

Returns

True if this View is in edit mode, false otherwise.

public boolean **isInLayout** () Added in API level 18

Returns whether the view hierarchy is currently undergoing a layout pass. This information is useful to avoid situations such as calling `requestLayout()` (/reference/android/view/View.html#requestLayout()) during a layout pass.

Returns

whether the view hierarchy is currently undergoing a layout pass

public boolean **isInTouchMode** () Added in API level 1

Returns whether the device is currently in touch mode. Touch mode is entered once the user begins interacting with the device by touch, and affects various things like whether focus is always visible to the user.

Returns

Whether the device is in touch mode.

public boolean **isLaidOut** () Added in API level 19

Returns true if this view has been through at least one layout since it was last attached to or detached from a window.

public boolean **isLayoutDirectionResolved** () Added in API level 19

Returns

true if layout direction has been resolved.

public boolean **isLayoutRequested** () Added in API level 1

Indicates whether or not this view's layout will be requested during the next hierarchy layout pass.

Returns

true if the layout will be forced during next layout pass

public boolean **isLongClickable** () Added in API level 1

Indicates whether this view reacts to long click events or not.

Related XML Attributes

android:longClickable

Returns

true if the view is long clickable, false otherwise

See Also

`setLongClickable(boolean)`

public boolean **isOpaque** () Added in API level 7

Indicates whether this View is opaque. An opaque View guarantees that it will draw all the pixels overlapping its bounds using a fully opaque color. Subclasses of View should override this method whenever possible to indicate whether an instance is opaque. Opaque Views are treated in a special way by the View hierarchy, possibly allowing it to perform optimizations during invalidate/draw passes.

Returns

True if this View is guaranteed to be fully opaque, false otherwise.

public boolean **isPaddingRelative** () Added in API level 17

Return if the padding as been set thru relative values `setPaddingRelative(int, int, int, int)` (/reference/android/view/View.html#setPaddingRelative(int, int, int, int)) or thru

Related XML Attributes

android:paddingStart
android:paddingEnd

Returns

true if the padding is relative or false if it is not.

public boolean **isPressed** () Added in API level 1

Indicates whether the view is currently in pressed state. Unless `setPressed(boolean)` (/reference/android/view/View.html#setPressed(boolean)) is explicitly called, only clickable views can enter the pressed state.

Returns

true if the view is currently pressed, false otherwise

See Also

`setPressed(boolean)`
`isClickable()`
`setClickable(boolean)`

public boolean **isSaveEnabled** () Added in API level 1

Indicates whether this view will save its state (that is, whether its `onSaveInstanceState()` (/reference/android/view/View.html#onSaveInstanceState()) method will be called).

Related XML Attributes

android:saveEnabled

Returns

Returns true if the view state saving is enabled, else false.

See Also

`setSaveEnabled(boolean)`

public boolean **isSaveFromParentEnabled** () Added in API level 11

Indicates whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent. The default is true; if false, these views will not be saved unless `saveHierarchyState(SparseArray)` (/reference/android/view/View.html#saveHierarchyState(android.util.SparseArray<android.os.Parcelable>)) is called directly on this view.

Returns

Returns true if the view state saving from parent is enabled, else false.

See Also

`setSaveFromParentEnabled(boolean)`

public boolean **isScrollContainer** () Added in API level 16

Indicates whether this view is one of the set of scrollable containers in its window.

Related XML Attributes

android:isScrollContainer

Returns

whether this view is one of the set of scrollable containers in its window

public boolean **isScrollbarFadingEnabled** () Added in API level 5

Returns true if scrollbars will fade when this view is not scrolling

Related XML Attributes

android:fadeScrollbars

Returns

true if scrollbar fading is enabled

public boolean **isSelected** () Added in API level 1

Indicates the selection state of this view.

Returns

true if the view is selected, false otherwise

public boolean **isShown** () Added in API level 1

Returns the visibility of this view and all of its ancestors

Returns

True if this view and all of its ancestors are `VISIBLE`

public boolean **isSoundEffectsEnabled** () Added in API level 1

Related XML Attributes

android:soundEffectsEnabled

Returns

whether this view should have sound effects enabled for events such as clicking and touching.

See Also

`setSoundEffectsEnabled(boolean)`
`playSoundEffect(int)`

public boolean **isTextAlignmentResolved** () Added in API level 19

Returns

true if text alignment is resolved.

public boolean **isTextDirectionResolved** () Added in API level 19

Returns

true if text direction is resolved.

public boolean **isVerticalFadingEdgeEnabled** () Added in API level 1

Indicate whether the vertical edges are faded when the view is scrolled horizontally.

Related XML Attributes

android:requiresFadingEdge

Returns

true if the vertical edges should are faded on scroll, false otherwise

See Also
setVerticalFadingEdgeEnabled(boolean)

public boolean **isVerticalScrollBarEnabled** () Added in API level 1

Indicate whether the vertical scrollbar should be drawn or not. The scrollbar is not drawn by default.

Returns
true if the vertical scrollbar should be painted, false otherwise

See Also
setVerticalScrollBarEnabled(boolean)

public void **jumpDrawablesToCurrentState** () Added in API level 11

Call `Drawable.jumpToCurrentState()` (/reference/android/graphics/drawable/Drawable.html#jumpToCurrentState()) on all Drawable objects associated with this view.

public void **layout** (int l, int t, int r, int b) Added in API level 1

Assign a size and position to a view and all of its descendants

This is the second phase of the layout mechanism. (The first is measuring). In this phase, each parent calls layout on all of its children to position them. This is typically done using the child measurements that were stored in the measure pass().

Derived classes should not override this method. Derived classes with children should override onLayout. In that method, they should call layout on each of their children.

Parameters

l	Left position, relative to parent
t	Top position, relative to parent
r	Right position, relative to parent
b	Bottom position, relative to parent

public final void **measure** (int widthMeasureSpec, int heightMeasureSpec) Added in API level 1

This is called to find out how big a view should be. The parent supplies constraint information in the width and height parameters.

The actual measurement work of a view is performed in `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)), called by this method. Therefore, only `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)) can and must be overridden by subclasses.

Parameters

widthMeasureSpec	Horizontal space requirements as imposed by the parent
heightMeasureSpec	Vertical space requirements as imposed by the parent

See Also
onMeasure(int, int)

public void **offsetLeftAndRight** (int offset) Added in API level 1

Offset this view's horizontal location by the specified amount of pixels.

Parameters

offset	the number of pixels to offset the view by

public void **offsetTopAndBottom** (int offset) Added in API level 1

Offset this view's vertical location by the specified number of pixels.

Parameters

offset	the number of pixels to offset the view by

public void **onCancelPendingInputEvents** () Added in API level 19

Called as the result of a call to `cancelPendingInputEvents()` (/reference/android/view/View.html#cancelPendingInputEvents()) on this view or a parent view.

This method is responsible for removing any pending high-level input events that were posted to the event queue to run later. Custom view classes that post their own deferred high-level events via `post(Runnable)` (/reference/android/view/View.html#post(java.lang.Runnable)), `postDelayed(Runnable, long)` (/reference/android/view/View.html#postDelayed(java.lang.Runnable, long)) or `Handler` (/reference/android/os/Handler.html) should override this method, call `super.onCancelPendingInputEvents()` and remove those callbacks as appropriate.

public boolean **onCheckIsTextEditor** () Added in API level 3

Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. Subclasses should override this if they implement `onCreateInputConnection(EditorInfo)` (/reference/android/view/View.html#onCreateInputConnection(android.view.inputmethod.EditorInfo)) to return true if a call on that method would return a non-null InputConnection, and they are really a first-class editor that the user would normally start typing on when the go into a window containing your view.

The default implementation always returns false. This does *not* mean that its `onCreateInputConnection(EditorInfo)` (/reference/android/view/View.html#onCreateInputConnection(android.view.inputmethod.EditorInfo)) will not be called or the user can not otherwise perform edits on your view; it is just a hint to the system that this is not the primary purpose of this view.

Returns
Returns true if this view is a text editor, else false.

public InputConnection **onCreateInputConnection** (EditorInfo outAttrs) Added in API level 3

Create a new InputConnection for an InputMethod to interact with the view. The default implementation returns null, since it doesn't support input methods. You can override this to implement such support. This is only needed for views that take focus and text input.

When implementing this, you probably also want to implement `onCheckIsTextEditor()` (/reference/android

/view/View.html#onCheckIsTextEditor()) to indicate you will return a non-null InputConnection.

Parameters

outAttrs	Fill in with attribute information about the connection.

public boolean **onDragEvent** (DragEvent event)

Added in API level 11

Handles drag events sent by the system following a call to `startDrag()` (/reference/android/view/View.html#startDrag(android.content.ClipData, android.view.View.DragShadowBuilder, java.lang.Object, int)).

When the system calls this method, it passes a `DragEvent` (/reference/android/view/DragEvent.html) object. A call to `getAction()` (/reference/android/view/DragEvent.html#getAction()) returns one of the action type constants defined in DragEvent. The method uses these to determine what is happening in the drag and drop operation.

Parameters

event	The `DragEvent` sent by the system. The `getAction()` method returns an action type constant defined in DragEvent, indicating the type of drag event represented by this object.

Returns

`true` if the method was successful, otherwise `false`.

The method should return `true` in response to an action type of `ACTION_DRAG_STARTED` (/reference/android/view/DragEvent.html#ACTION_DRAG_STARTED) to receive drag events for the current operation.

The method should also return `true` in response to an action type of `ACTION_DROP` (/reference/android/view/DragEvent.html#ACTION_DROP) if it consumed the drop, or `false` if it didn't.

public boolean **onFilterTouchEventForSecurity** (MotionEvent event)

Added in API level 9

Filter the touch event to apply security policies.

Parameters

event	The motion event to be filtered.

Returns

True if the event should be dispatched, false if the event should be dropped.

See Also

`getFilterTouchesWhenObscured()`

public void **onFinishTemporaryDetach** ()

Added in API level 3

Called after `onStartTemporaryDetach()` (/reference/android/view/View.html#onStartTemporaryDetach()) when the container is done changing the view.

public boolean **onGenericMotionEvent** (MotionEvent event)

Added in API level 12

Implement this method to handle generic motion events.

Generic motion events describe joystick movements, mouse hovers, track pad touches, scroll wheel movements and other input events. The `source` (/reference/android/view/MotionEvent.html#getSource()) of the motion event specifies the class of input that was received. Implementations of this method must examine the bits in the source before processing the event. The following code example shows how this is done.

Generic motion events with source class `SOURCE_CLASS_POINTER` (/reference/android/view/InputDevice.html#SOURCE_CLASS_POINTER) are delivered to the view under the pointer. All other generic motion events are delivered to the focused view.

```java
public boolean onGenericMotionEvent(MotionEvent event) {
    if (event.isFromSource(InputDevice.SOURCE_CLASS_JOYSTICK)) {
        if (event.getAction() == MotionEvent.ACTION_MOVE) {
            // process the joystick movement...
            return true;
        }
    }
    if (event.isFromSource(InputDevice.SOURCE_CLASS_POINTER)) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_HOVER_MOVE:
                // process the mouse hover movement...
                return true;
            case MotionEvent.ACTION_SCROLL:
                // process the scroll wheel movement...
                return true;
        }
    }
    return super.onGenericMotionEvent(event);
}
```

Parameters

event	The generic motion event being processed.

Returns

True if the event was handled, false otherwise.

public void **onHoverChanged** (boolean hovered)

Added in API level 14

Implement this method to handle hover state changes.

This method is called whenever the hover state changes as a result of a call to `setHovered(boolean)` (/reference/android/view/View.html#setHovered(boolean)).

Parameters

hovered	The current hover state, as returned by `isHovered()`.

See Also

`isHovered()`
`setHovered(boolean)`

public boolean **onHoverEvent** (MotionEvent event)

Added in API level 14

Implement this method to handle hover events.

This method is called whenever a pointer is hovering into, over, or out of the bounds of a view and the view is not currently being touched. Hover events are represented as pointer events with action `ACTION_HOVER_ENTER` (/reference/android/view/MotionEvent.html#ACTION_HOVER_ENTER), `ACTION_HOVER_MOVE` (/reference/android/view/MotionEvent.html#ACTION_HOVER_MOVE), or `ACTION_HOVER_EXIT` (/reference/android/view/MotionEvent.html#ACTION_HOVER_EXIT).

- The view receives a hover event with action `ACTION_HOVER_ENTER` when the pointer enters the bounds of the view.
- The view receives a hover event with action `ACTION_HOVER_MOVE` when the pointer has already entered the bounds of the view and has moved.
- The view receives a hover event with action `ACTION_HOVER_EXIT` when the pointer has exited the bounds of the view or when the pointer is about to go down due to a button click, tap, or similar user action that causes the view to be touched.

The view should implement this method to return true to indicate that it is handling the hover event, such as by changing its drawable state.

The default implementation calls `setHovered(boolean)` (/reference/android/view/View.html#setHovered(boolean)) to update the hovered state of the view when a hover enter or hover exit event is received, if the view is enabled and is clickable. The default implementation also sends hover accessibility events.

Parameters

event	The motion event that describes the hover.

Returns

True if the view handled the hover event.

See Also

`isHovered()`
`setHovered(boolean)`
`onHoverChanged(boolean)`

public void **onInitializeAccessibilityEvent** (AccessibilityEvent event)

Added in API level 14

Initializes an `AccessibilityEvent` (/reference/android/view/accessibility/AccessibilityEvent.html) with information about this View which is the event source. In other words, the source of an accessibility event is the view whose state change triggered firing the event.

Example: Setting the password property of an event in addition to properties set by the super implementation:

```
public void onInitializeAccessibilityEvent(AccessibilityEvent event) {
    super.onInitializeAccessibilityEvent(event);
    event.setPassword(true);
}
```

If an `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate)` (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `onInitializeAccessibilityEvent(View, AccessibilityEvent)` (/reference/android/view/View.AccessibilityDelegate.html#onInitializeAccessibilityEvent(android.view.View, android.view.accessibility.AccessibilityEvent)) is responsible for handling this call.

> **Note:** Always call the super implementation before adding information to the event, in case the default implementation has basic information to add.

Parameters

event	The event to initialize.

See Also

`sendAccessibilityEvent(int)`
`dispatchPopulateAccessibilityEvent(AccessibilityEvent)`

public void **onInitializeAccessibilityNodeInfo** (AccessibilityNodeInfo info)

Added in API level 14

Initializes an `AccessibilityNodeInfo` (/reference/android/view/accessibility/AccessibilityNodeInfo.html) with information about this view. The base implementation sets:

- `setParent(View)`,
- `setBoundsInParent(Rect)`,
- `setBoundsInScreen(Rect)`,
- `setPackageName(CharSequence)`,
- `setClassName(CharSequence)`,
- `setContentDescription(CharSequence)`,
- `setEnabled(boolean)`,
- `setClickable(boolean)`,
- `setFocusable(boolean)`,
- `setFocused(boolean)`,
- `setLongClickable(boolean)`,
- `setSelected(boolean)`,

Subclasses should override this method, call the super implementation, and set additional attributes.

If an `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate)` (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `onInitializeAccessibilityNodeInfo(View, AccessibilityNodeInfo)` (/reference/android/view/View.AccessibilityDelegate.html#onInitializeAccessibilityNodeInfo(android.view.View, android.view.accessibility.AccessibilityNodeInfo)) is responsible for handling this call.

Parameters

info	The instance to initialize.

public boolean **onKeyDown** (int keyCode, KeyEvent event)

Added in API level 1

Default implementation of `KeyEvent.Callback.onKeyDown()` (/reference/android/view/KeyEvent.Callback.html#onKeyDown(int, android.view.KeyEvent)): perform press of the view when `KEYCODE_DPAD_CENTER` (/reference/android/view/KeyEvent.html#KEYCODE_DPAD_CENTER) or

`KEYCODE_ENTER` (/reference/android/view/KeyEvent.html#KEYCODE_ENTER) is released, if the view is enabled and clickable.

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

Parameters

keyCode	A key code that represents the button pressed, from `KeyEvent`.
event	The KeyEvent object that defines the button action.

Returns
If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyLongPress** (int keyCode, KeyEvent event)

Added in API level 5

Default implementation of `KeyEvent.Callback.onKeyLongPress()` (/reference/android/view/KeyEvent.Callback.html#onKeyLongPress(int, android.view.KeyEvent)): always returns false (doesn't handle the event).

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

Parameters

keyCode	The value in event.getKeyCode().
event	Description of the key event.

Returns
If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyMultiple** (int keyCode, int repeatCount, KeyEvent event)

Added in API level 1

Default implementation of `KeyEvent.Callback.onKeyMultiple()` (/reference/android/view/KeyEvent.Callback.html#onKeyMultiple(int, int, android.view.KeyEvent)): always returns false (doesn't handle the event).

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

Parameters

keyCode	A key code that represents the button pressed, from `KeyEvent`.
repeatCount	The number of times the action was made.
event	The KeyEvent object that defines the button action.

Returns
If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyPreIme** (int keyCode, KeyEvent event)

Added in API level 3

Handle a key event before it is processed by any input method associated with the view hierarchy. This can be used to intercept key events in special situations before the IME consumes them; a typical example would be handling the BACK key to update the application's UI instead of allowing the IME to see it and close itself.

Parameters

keyCode	The value in event.getKeyCode().
event	Description of the key event.

Returns
If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyShortcut** (int keyCode, KeyEvent event)

Added in API level 1

Called on the focused view when a key shortcut event is not handled. Override this method to implement local key shortcuts for the View. Key shortcuts can also be implemented by setting the `shortcut` (/reference/android/view/MenuItem.html#setShortcut(char, char)) property of menu items.

Parameters

keyCode	The value in event.getKeyCode().
event	Description of the key event.

Returns
If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyUp** (int keyCode, KeyEvent event)

Added in API level 1

Default implementation of `KeyEvent.Callback.onKeyUp()` (/reference/android/view/KeyEvent.Callback.html#onKeyUp(int, android.view.KeyEvent)): perform clicking of the view when `KEYCODE_DPAD_CENTER` (/reference/android/view/KeyEvent.html#KEYCODE_DPAD_CENTER) or `KEYCODE_ENTER` (/reference/android/view/KeyEvent.html#KEYCODE_ENTER) is released.

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

Parameters

keyCode	A key code that represents the button pressed, from `KeyEvent`.
event	The KeyEvent object that defines the button action.

Returns
If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public void **onPopulateAccessibilityEvent** (AccessibilityEvent event)

Added in API level 14

Called from `dispatchPopulateAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) giving a chance to this View to populate the accessibility event with its text content. While this method is free to modify event attributes other than text content, doing so should normally be performed in `onInitializeAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#onInitializeAccessibilityEvent(android.view.accessibility.AccessibilityEvent)).

Example: Adding formatted date string to an accessibility event in addition to the text added by the super implementation:

```
public void onPopulateAccessibilityEvent(AccessibilityEvent event) {
    super.onPopulateAccessibilityEvent(event);
    final int flags = DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY;
    String selectedDateUtterance = DateUtils.formatDateTime(mContext,
        mCurrentDate.getTimeInMillis(), flags);
    event.getText().add(selectedDateUtterance);
}
```

If an `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate)` (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `onPopulateAccessibilityEvent(View, AccessibilityEvent)` (/reference/android/view/View.AccessibilityDelegate.html#onPopulateAccessibilityEvent(android.view.View, android.view.accessibility.AccessibilityEvent)) is responsible for handling this call.

Note: Always call the super implementation before adding information to the event, in case the default implementation has basic information to add.

Parameters

event	The accessibility event which to populate.

See Also

`sendAccessibilityEvent(int)`
`dispatchPopulateAccessibilityEvent(AccessibilityEvent)`

public void **onRtlPropertiesChanged** (int layoutDirection)

Added in API level 17

Called when any RTL property (layout direction or text direction or text alignment) has been changed. Subclasses need to override this method to take care of cached information that depends on the resolved layout direction, or to inform child views that inherit their layout direction. The default implementation does nothing.

Parameters

layoutDirection	the direction of the layout

See Also

`LAYOUT_DIRECTION_LTR`
`LAYOUT_DIRECTION_RTL`

public void **onScreenStateChanged** (int screenState)

Added in API level 16

This method is called whenever the state of the screen this view is attached to changes. A state change will usually occurs when the screen turns on or off (whether it happens automatically or the user does it manually.)

Parameters

screenState	The new state of the screen. Can be either `SCREEN_STATE_ON` or `SCREEN_STATE_OFF`

public void **onStartTemporaryDetach** ()

Added in API level 3

This is called when a container is going to temporarily detach a child, with `ViewGroup.detachViewFromParent` (/reference/android/view/ViewGroup.html#detachViewFromParent(android.view.View)). It will either be followed by `onFinishTemporaryDetach()` (/reference/android/view/View.html#onFinishTemporaryDetach()) or `onDetachedFromWindow()` (/reference/android/view/View.html#onDetachedFromWindow()) when the container is done.

public boolean **onTouchEvent** (MotionEvent event)

Added in API level 1

Implement this method to handle touch screen motion events.

If this method is used to detect click actions, it is recommended that the actions be performed by implementing and calling `performClick()` (/reference/android/view/View.html#performClick()). This will ensure consistent system behavior, including:

- obeying click sound preferences
- dispatching OnClickListener calls
- handling `ACTION_CLICK` when accessibility features are enabled

Parameters

event	The motion event.

Returns

True if the event was handled, false otherwise.

public boolean **onTrackballEvent** (MotionEvent event)

Added in API level 1

Implement this method to handle trackball motion events. The *relative* movement of the trackball since the last event can be retrieve with `MotionEvent.getX()` (/reference/android/view/MotionEvent.html#getX()) and `MotionEvent.getY()` (/reference/android/view/MotionEvent.html#getY()). These are normalized so that a movement of 1 corresponds to the user pressing one DPAD key (so they will often be fractional values, representing the more fine-grained movement information available from a trackball).

Parameters

event	The motion event.

Returns

True if the event was handled, false otherwise.

public void **onWindowFocusChanged** (boolean hasWindowFocus)

Added in API level 1

Called when the window containing this view gains or loses focus. Note that this is separate from view focus: to receive key events, both your view and its window must have focus. If a window is displayed on top of yours that takes input focus, then your own window will lose focus but the view focus will remain unchanged.

Parameters

hasWindowFocus	True if the window containing this view now has focus, false otherwise.

public void **onWindowSystemUiVisibilityChanged** (int visible)

Added in API level 16

Override to find out when the window's requested system UI visibility has changed, that is the value returned by getWindowSystemUiVisibility() (/reference/android/view/View.html#getWindowSystemUiVisibility()). This is different from the callbacks received through setOnSystemUiVisibilityChangeListener(OnSystemUiVisibilityChangeListener) (/reference/android/view/View.html#setOnSystemUiVisibilityChangeListener(android.view.View.OnSystemUiVisibilityChangeListener)) in that this is only telling you about the local request of the window, not the actual values applied by the system.

public boolean **performAccessibilityAction** (int action, Bundle arguments) Added in API level 16

Performs the specified accessibility action on the view. For possible accessibility actions look at AccessibilityNodeInfo (/reference/android/view/accessibility/AccessibilityNodeInfo.html).

If an View.AccessibilityDelegate (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling setAccessibilityDelegate(AccessibilityDelegate) (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its performAccessibilityAction(View, int, Bundle) (/reference/android/view/View.AccessibilityDelegate.html#performAccessibilityAction(android.view.View, int, android.os.Bundle)) is responsible for handling this call.

Parameters

action	The action to perform.
arguments	Optional action arguments.

Returns
Whether the action was performed.

public boolean **performClick** () Added in API level 1

Call this view's OnClickListener, if it is defined. Performs all normal actions associated with clicking: reporting accessibility event, playing a sound, etc.

Returns
True there was an assigned OnClickListener that was called, false otherwise is returned.

public boolean **performHapticFeedback** (int feedbackConstant) Added in API level 3

BZZZTT!!1!

Provide haptic feedback to the user for this view.

The framework will provide haptic feedback for some built in actions, such as long presses, but you may wish to provide feedback for your own widget.

The feedback will only be performed if isHapticFeedbackEnabled() (/reference/android/view/View.html#isHapticFeedbackEnabled()) is true.

Parameters

feedbackConstant	One of the constants defined in HapticFeedbackConstants

public boolean **performHapticFeedback** (int feedbackConstant, int flags) Added in API level 3

BZZZTT!!1!

Like performHapticFeedback(int) (/reference/android/view/View.html#performHapticFeedback(int)), with additional options.

Parameters

feedbackConstant	One of the constants defined in HapticFeedbackConstants
flags	Additional flags as per HapticFeedbackConstants.

public boolean **performLongClick** () Added in API level 1

Call this view's OnLongClickListener, if it is defined. Invokes the context menu if the OnLongClickListener did not consume the event.

Returns
True if one of the above receivers consumed the event, false otherwise.

public void **playSoundEffect** (int soundConstant) Added in API level 1

Play a sound effect for this view.

The framework will play sound effects for some built in actions, such as clicking, but you may wish to play these effects in your widget, for instance, for internal navigation.

The sound effect will only be played if sound effects are enabled by the user, and isSoundEffectsEnabled() (/reference/android/view/View.html#isSoundEffectsEnabled()) is true.

Parameters

soundConstant	One of the constants defined in SoundEffectConstants

public boolean **post** (Runnable action) Added in API level 1

Causes the Runnable to be added to the message queue. The runnable will be run on the user interface thread.

Parameters

action	The Runnable that will be executed.

Returns
Returns true if the Runnable was successfully placed in to the message queue. Returns false on failure, usually because the looper processing the message queue is exiting.

See Also
postDelayed(Runnable, long)
removeCallbacks(Runnable)

public boolean **postDelayed** (Runnable action, long delayMillis) Added in API level 1

Causes the Runnable to be added to the message queue, to be run after the specified amount of time elapses. The runnable will be run on the user interface thread.

Parameters

action	The Runnable that will be executed.
delayMillis	The delay (in milliseconds) until the Runnable will be executed.

Returns

true if the Runnable was successfully placed in to the message queue. Returns false on failure, usually because the looper processing the message queue is exiting. Note that a result of true does not mean the Runnable will be processed -- if the looper is quit before the delivery time of the message occurs then the message will be dropped.

See Also

`post(Runnable)`
`removeCallbacks(Runnable)`

public void **postInvalidate** (int left, int top, int right, int bottom)

Added in API level 1

Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop. Use this to invalidate the View from a non-UI thread.

This method can be invoked from outside of the UI thread only when this View is attached to a window.

Parameters

left	The left coordinate of the rectangle to invalidate.
top	The top coordinate of the rectangle to invalidate.
right	The right coordinate of the rectangle to invalidate.
bottom	The bottom coordinate of the rectangle to invalidate.

See Also

`invalidate(int, int, int, int)`
`invalidate(Rect)`
`postInvalidateDelayed(long, int, int, int, int)`

public void **postInvalidate** ()

Added in API level 1

Cause an invalidate to happen on a subsequent cycle through the event loop. Use this to invalidate the View from a non-UI thread.

This method can be invoked from outside of the UI thread only when this View is attached to a window.

See Also

`invalidate()`
`postInvalidateDelayed(long)`

public void **postInvalidateDelayed** (long delayMilliseconds, int left, int top, int right, int bottom)

Added in API level 1

Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop. Waits for the specified amount of time.

This method can be invoked from outside of the UI thread only when this View is attached to a window.

Parameters

delayMilliseconds	the duration in milliseconds to delay the invalidation by
left	The left coordinate of the rectangle to invalidate.
top	The top coordinate of the rectangle to invalidate.
right	The right coordinate of the rectangle to invalidate.
bottom	The bottom coordinate of the rectangle to invalidate.

See Also

`invalidate(int, int, int, int)`
`invalidate(Rect)`
`postInvalidate(int, int, int, int)`

public void **postInvalidateDelayed** (long delayMilliseconds)

Added in API level 1

Cause an invalidate to happen on a subsequent cycle through the event loop. Waits for the specified amount of time.

This method can be invoked from outside of the UI thread only when this View is attached to a window.

Parameters

delayMilliseconds	the duration in milliseconds to delay the invalidation by

See Also

`invalidate()`
`postInvalidate()`

public void **postInvalidateOnAnimation** (int left, int top, int right, int bottom)

Added in API level 16

Cause an invalidate of the specified area to happen on the next animation time step, typically the next display frame.

This method can be invoked from outside of the UI thread only when this View is attached to a window.

Parameters

left	The left coordinate of the rectangle to invalidate.
top	The top coordinate of the rectangle to invalidate.
right	The right coordinate of the rectangle to invalidate.
bottom	The bottom coordinate of the rectangle to invalidate.

See Also

`invalidate(int, int, int, int)`
`invalidate(Rect)`

public void **postInvalidateOnAnimation** ()

Added in API level 16

Cause an invalidate to happen on the next animation time step, typically the next display frame.

This method can be invoked from outside of the UI thread only when this View is attached to a window.

See Also
`invalidate()`

public void **postOnAnimation** (Runnable action) Added in API level 16

Causes the Runnable to execute on the next animation time step. The runnable will be run on the user interface thread.

Parameters

action	The Runnable that will be executed.

See Also
`postOnAnimationDelayed(Runnable, long)`
`removeCallbacks(Runnable)`

public void **postOnAnimationDelayed** (Runnable action, long delayMillis) Added in API level 16

Causes the Runnable to execute on the next animation time step, after the specified amount of time elapses. The runnable will be run on the user interface thread.

Parameters

action	The Runnable that will be executed.
delayMillis	The delay (in milliseconds) until the Runnable will be executed.

See Also
`postOnAnimation(Runnable)`
`removeCallbacks(Runnable)`

public void **refreshDrawableState** () Added in API level 1

Call this to force a view to update its drawable state. This will cause drawableStateChanged to be called on this view. Views that are interested in the new state should call getDrawableState.

See Also
`drawableStateChanged()`
`getDrawableState()`

public boolean **removeCallbacks** (Runnable action) Added in API level 1

Removes the specified Runnable from the message queue.

Parameters

action	The Runnable to remove from the message handling queue

Returns
true if this view could ask the Handler to remove the Runnable, false otherwise. When the returned value is true, the Runnable may or may not have been actually removed from the message queue (for instance, if the Runnable was not in the queue already.)

See Also
`post(Runnable)`
`postDelayed(Runnable, long)`
`postOnAnimation(Runnable)`
`postOnAnimationDelayed(Runnable, long)`

public void **removeOnAttachStateChangeListener** (View.OnAttachStateChangeListener listener) Added in API level 12

Remove a listener for attach state changes. The listener will receive no further notification of window attach/detach events.

Parameters

listener	Listener to remove

See Also
`addOnAttachStateChangeListener(OnAttachStateChangeListener)`

public void **removeOnLayoutChangeListener** (View.OnLayoutChangeListener listener) Added in API level 11

Remove a listener for layout changes.

Parameters

listener	The listener for layout bounds change.

public void **requestFitSystemWindows** () Added in API level 16

Ask that a new dispatch of `fitSystemWindows(Rect)` (/reference/android/view/View.html#fitSystemWindows(android.graphics.Rect)) be performed.

public boolean **requestFocus** (int direction, Rect previouslyFocusedRect) Added in API level 1

Call this to try to give focus to a specific view or to one of its descendants and give it hints about the direction and a specific rectangle that the focus is coming from. The rectangle can help give larger views a finer grained hint about where focus is coming from, and therefore, where to show selection, or forward focus change internally. A view will not actually take focus if it is not focusable (`isFocusable()` (/reference/android/view/View.html#isFocusable()) returns false), or if it is focusable and it is not focusable in touch mode (`isFocusableInTouchMode()` (/reference/android/view/View.html#isFocusableInTouchMode())) while the device is in touch mode. A View will not take focus if it is not visible. A View will not take focus if one of its parents has `getDescendantFocusability()` (/reference/android/view/ViewGroup.html#getDescendantFocusability()) equal to `FOCUS_BLOCK_DESCENDANTS` (/reference/android/view/ViewGroup.html#FOCUS_BLOCK_DESCENDANTS). See also `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)), which is what you call to say that you have focus, and you want your parent to look for the next one. You may wish to override this method if your custom `View` (/reference/android/view/View.html) has an internal `View` (/reference/android/view/View.html) that it wishes to forward the request to.

Parameters

direction	One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT

previouslyFocusedRect	The rectangle (in this View's coordinate system) to give a finer grained hint about where focus is coming from. May be null if there is no hint.

Returns
Whether this view or one of its descendants actually took focus.

public final boolean **requestFocus** (int direction) Added in API level 1

Call this to try to give focus to a specific view or to one of its descendants and give it a hint about what direction focus is heading. A view will not actually take focus if it is not focusable (`isFocusable()` (/reference/android/view/View.html#isFocusable()) returns false), or if it is focusable and it is not focusable in touch mode (`isFocusableInTouchMode()` (/reference/android/view/View.html#isFocusableInTouchMode())) while the device is in touch mode. See also `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)), which is what you call to say that you have focus, and you want your parent to look for the next one. This is equivalent to calling `requestFocus(int, Rect)` (/reference/android/view/View.html#requestFocus(int, android.graphics.Rect)) with `null` set for the previously focused rectangle.

Parameters

direction	One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT

Returns
Whether this view or one of its descendants actually took focus.

public final boolean **requestFocus** () Added in API level 1

Call this to try to give focus to a specific view or to one of its descendants. A view will not actually take focus if it is not focusable (`isFocusable()` (/reference/android/view/View.html#isFocusable()) returns false), or if it is focusable and it is not focusable in touch mode (`isFocusableInTouchMode()` (/reference/android/view/View.html#isFocusableInTouchMode())) while the device is in touch mode. See also `focusSearch(int)` (/reference/android/view/View.html#focusSearch(int)), which is what you call to say that you have focus, and you want your parent to look for the next one. This is equivalent to calling `requestFocus(int, Rect)` (/reference/android/view/View.html#requestFocus(int, android.graphics.Rect)) with arguments `FOCUS_DOWN` (/reference/android/view/View.html#FOCUS_DOWN) and `null`.

Returns
Whether this view or one of its descendants actually took focus.

public final boolean **requestFocusFromTouch** () Added in API level 1

Call this to try to give focus to a specific view or to one of its descendants. This is a special variant of `requestFocus()` (/reference/android/view/View.html#requestFocus()) that will allow views that are not focuable in touch mode to request focus when they are touched.

Returns
Whether this view or one of its descendants actually took focus.

See Also
`isInTouchMode()`

public void **requestLayout** () Added in API level 1

Call this when something has changed which has invalidated the layout of this view. This will schedule a layout pass of the view tree. This should not be called while the view hierarchy is currently in a layout pass (`isInLayout()` (/reference/android/view/View.html#isInLayout()). If layout is happening, the request may be honored at the end of the current layout pass (and then layout will run again) or after the current frame is drawn and the next layout occurs.

Subclasses which override this method should call the superclass method to handle possible request-during-layout errors correctly.

public boolean **requestRectangleOnScreen** (Rect rectangle) Added in API level 1

Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough.

A View should call this if it maintains some notion of which part of its content is interesting. For example, a text editing view should call this when its cursor moves.

Parameters

rectangle	The rectangle.

Returns
Whether any parent scrolled.

public boolean **requestRectangleOnScreen** (Rect rectangle, boolean immediate) Added in API level 1

Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough.

A View should call this if it maintains some notion of which part of its content is interesting. For example, a text editing view should call this when its cursor moves.

When `immediate` is set to true, scrolling will not be animated.

Parameters

rectangle	The rectangle.
immediate	True to forbid animated scrolling, false otherwise

Returns
Whether any parent scrolled.

public static int **resolveSize** (int size, int measureSpec) Added in API level 1

Version of `resolveSizeAndState(int, int, int)` (/reference/android/view/View.html#resolveSizeAndState(int, int, int)) returning only the `MEASURED_SIZE_MASK` (/reference/android/view/View.html#MEASURED_SIZE_MASK) bits of the result.

public static int **resolveSizeAndState** (int size, int measureSpec, int childMeasuredState) Added in API level 11

Utility to reconcile a desired size and state, with constraints imposed by a MeasureSpec. Will take the desired size, unless a different size is imposed by the constraints. The returned value is a compound integer, with the resolved size in the `MEASURED_SIZE_MASK` (/reference/android/view/View.html#MEASURED_SIZE_MASK) bits and optionally the bit `MEASURED_STATE_TOO_SMALL` (/reference/android/view/View.html#MEASURED_STATE_TOO_SMALL) set if the resulting size is smaller than the size the view wants to be.

Parameters

size	How big the view wants to be
measureSpec	Constraints imposed by the parent

Returns

Size information bit mask as defined by `MEASURED_SIZE_MASK` and `MEASURED_STATE_TOO_SMALL`.

public void **restoreHierarchyState** (SparseArray<Parcelable> container) Added in API level 1

Restore this view hierarchy's frozen state from the given container.

Parameters

container	The SparseArray which holds previously frozen states.

See Also

`saveHierarchyState(android.util.SparseArray)`
`dispatchRestoreInstanceState(android.util.SparseArray)`
`onRestoreInstanceState(android.os.Parcelable)`

public void **saveHierarchyState** (SparseArray<Parcelable> container) Added in API level 1

Store this view hierarchy's frozen state into the given container.

Parameters

container	The SparseArray in which to save the view's state.

See Also

`restoreHierarchyState(android.util.SparseArray)`
`dispatchSaveInstanceState(android.util.SparseArray)`
`onSaveInstanceState()`

public void **scheduleDrawable** (Drawable who, Runnable what, long when) Added in API level 1

Schedules an action on a drawable to occur at a specified time.

Parameters

who	the recipient of the action
what	the action to run on the drawable
when	the time at which the action must occur. Uses the `uptimeMillis()` timebase.

public void **scrollBy** (int x, int y) Added in API level 1

Move the scrolled position of your view. This will cause a call to `onScrollChanged(int, int, int, int)` (/reference/android/view/View.html#onScrollChanged(int, int, int, int)) and the view will be invalidated.

Parameters

x	the amount of pixels to scroll by horizontally
y	the amount of pixels to scroll by vertically

public void **scrollTo** (int x, int y) Added in API level 1

Set the scrolled position of your view. This will cause a call to `onScrollChanged(int, int, int, int)` (/reference/android/view/View.html#onScrollChanged(int, int, int, int)) and the view will be invalidated.

Parameters

x	the x position to scroll to
y	the y position to scroll to

public void **sendAccessibilityEvent** (int eventType) Added in API level 4

Sends an accessibility event of the given type. If accessibility is not enabled this method has no effect. The default implementation calls `onInitializeAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#onInitializeAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) first to populate information about the event source (this View), then calls `dispatchPopulateAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) to populate the text content of the event source including its descendants, and last calls `requestSendAccessibilityEvent(View, AccessibilityEvent)` (/reference/android/view/ViewParent.html#requestSendAccessibilityEvent(android.view.View, android.view.accessibility.AccessibilityEvent)) on its parent to resuest sending of the event to interested parties.

If an `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate)` (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `sendAccessibilityEvent(View, int)` (/reference/android/view/View.AccessibilityDelegate.html#sendAccessibilityEvent(android.view.View, int)) is responsible for handling this call.

Parameters

eventType	The type of the event to send, as defined by several types from `AccessibilityEvent`, such as `TYPE_VIEW_CLICKED` or `TYPE_VIEW_HOVER_ENTER`.

See Also

`onInitializeAccessibilityEvent(AccessibilityEvent)`
`dispatchPopulateAccessibilityEvent(AccessibilityEvent)`
`requestSendAccessibilityEvent(View, AccessibilityEvent)`
`View.AccessibilityDelegate`

public void **sendAccessibilityEventUnchecked** (AccessibilityEvent event) Added in API level 4

This method behaves exactly as `sendAccessibilityEvent(int)` (/reference/android/view/View.html#sendAccessibilityEvent(int)) but takes as an argument an empty `AccessibilityEvent` (/reference/android/view/accessibility/AccessibilityEvent.html) and does not perform a check whether accessibility is enabled.

If an `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html) has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate)` (/reference/android

/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `sendAccessibilityEventUnchecked(View, AccessibilityEvent)` (/reference/android/view/View.AccessibilityDelegate.html#sendAccessibilityEventUnchecked(android.view.View, android.view.accessibility.AccessibilityEvent)) is responsible for handling this call.

Parameters

event	The event to send.

See Also

`sendAccessibilityEvent(int)`

public void **setAccessibilityDelegate** (View.AccessibilityDelegate delegate) Added in API level 14

Sets a delegate for implementing accessibility support via composition as opposed to inheritance. The delegate's primary use is for implementing backwards compatible widgets. For more details see `View.AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate.html).

Parameters

delegate	The delegate instance.

See Also

`View.AccessibilityDelegate`

public void **setAccessibilityLiveRegion** (int mode) Added in API level 19

Sets the live region mode for this view. This indicates to accessibility services whether they should automatically notify the user about changes to the view's content description or text, or to the content descriptions or text of the view's children (where applicable).

For example, in a login screen with a TextView that displays an "incorrect password" notification, that view should be marked as a live region with mode `ACCESSIBILITY_LIVE_REGION_POLITE` (/reference/android/view/View.html#ACCESSIBILITY_LIVE_REGION_POLITE).

To disable change notifications for this view, use `ACCESSIBILITY_LIVE_REGION_NONE` (/reference/android/view/View.html#ACCESSIBILITY_LIVE_REGION_NONE). This is the default live region mode for most views.

To indicate that the user should be notified of changes, use `ACCESSIBILITY_LIVE_REGION_POLITE` (/reference/android/view/View.html#ACCESSIBILITY_LIVE_REGION_POLITE).

If the view's changes should interrupt ongoing speech and notify the user immediately, use `ACCESSIBILITY_LIVE_REGION_ASSERTIVE` (/reference/android/view/View.html#ACCESSIBILITY_LIVE_REGION_ASSERTIVE).

Related XML Attributes

android:accessibilityLiveRegion

Parameters

mode The live region mode for this view, one of:

- `ACCESSIBILITY_LIVE_REGION_NONE`
- `ACCESSIBILITY_LIVE_REGION_POLITE`
- `ACCESSIBILITY_LIVE_REGION_ASSERTIVE`

public void **setActivated** (boolean activated) Added in API level 11

Changes the activated state of this view. A view can be activated or not. Note that activation is not the same as selection. Selection is a transient property, representing the view (hierarchy) the user is currently interacting with. Activation is a longer-term state that the user can move views in and out of. For example, in a list view with single or multiple selection enabled, the views in the current selection set are activated. (Um, yeah, we are deeply sorry about the terminology here.) The activated state is propagated down to children of the view it is set on.

Parameters

activated	true if the view must be activated, false otherwise

public void **setAlpha** (float alpha) Added in API level 11

Sets the opacity of the view. This is a value from 0 to 1, where 0 means the view is completely transparent and 1 means the view is completely opaque.

Note that setting alpha to a translucent value (0 < alpha < 1) can have significant performance implications, especially for large views. It is best to use the alpha property sparingly and transiently, as in the case of fading animations.

For a view with a frequently changing alpha, such as during a fading animation, it is strongly recommended for performance reasons to either override `hasOverlappingRendering()` (/reference/android/view/View.html#hasOverlappingRendering()) to return false if appropriate, or setting a `layer type` (/reference/android/view/View.html#setLayerType(int, android.graphics.Paint)) on the view.

If this view overrides `onSetAlpha(int)` (/reference/android/view/View.html#onSetAlpha(int)) to return true, then this view is responsible for applying the opacity itself.

Note that if the view is backed by a `layer` (/reference/android/view/View.html#setLayerType(int, android.graphics.Paint)) and is associated with a `layer paint` (/reference/android/view/View.html#setLayerPaint(android.graphics.Paint)), setting an alpha value less than 1.0 will supercede the alpha of the layer paint.

Related XML Attributes

android:alpha

Parameters

alpha	The opacity of the view.

See Also

`hasOverlappingRendering()`
`setLayerType(int, android.graphics.Paint)`

public void **setAnimation** (Animation animation) Added in API level 1

Sets the next animation to play for this view. If you want the animation to play immediately, use `startAnimation(android.view.animation.Animation)` (/reference/android/view/View.html#startAnimation(android.view.animation.Animation)) instead. This method provides allows fine-grained control over the start time and invalidation, but you must make sure that 1) the animation has a start time set, and 2) the view's parent (which controls animations on its children) will be invalidated when the animation is supposed to start.

Parameters

animation	The next animation, or null.

public void **setBackground** ([Drawable] background) Added in API level 16

Set the background to a given Drawable, or remove the background. If the background has padding, this View's padding is set to the background's padding. However, when a background is removed, this View's padding isn't touched. If setting the padding is desired, please use `setPadding(int, int, int, int)` (/reference/android/view/View.html#setPadding(int, int, int, int)).

Parameters

background	The Drawable to use as the background, or null to remove the background

public void **setBackgroundColor** (int color) Added in API level 1

Sets the background color for this view.

Parameters

color	the color of the background

public void **setBackgroundDrawable** (Drawable background) Added in API level 1

This method was deprecated in API level 16.
use `setBackground(Drawable)` (/reference/android/view/View.html#setBackground(android.graphics.drawable.Drawable)) instead

public void **setBackgroundResource** (int resid) Added in API level 1

Set the background to a given resource. The resource should refer to a Drawable object or 0 to remove the background.

Related XML Attributes

android:background

Parameters

resid	The identifier of the resource.

public final void **setBottom** (int bottom) Added in API level 11

Sets the bottom position of this view relative to its parent. This method is meant to be called by the layout system and should not generally be called otherwise, because the property may be changed at any time by the layout.

Parameters

bottom	The bottom of this view, in pixels.

public void **setCameraDistance** (float distance) Added in API level 12

Sets the distance along the Z axis (orthogonal to the X/Y plane on which views are drawn) from the camera to this view. The camera's distance affects 3D transformations, for instance rotations around the X and Y axis. If the rotationX or rotationY properties are changed and this view is large (more than half the size of the screen), it is recommended to always use a camera distance that's greater than the height (X axis rotation) or the width (Y axis rotation) of this view.

The distance of the camera from the view plane can have an affect on the perspective distortion of the view when it is rotated around the x or y axis. For example, a large distance will result in a large viewing angle, and there will not be much perspective distortion of the view as it rotates. A short distance may cause much more perspective distortion upon rotation, and can also result in some drawing artifacts if the rotated view ends up partially behind the camera (which is why the recommendation is to use a distance at least as far as the size of the view, if the view is to be rotated.)

The distance is expressed in "depth pixels." The default distance depends on the screen density. For instance, on a medium density display, the default distance is 1280. On a high density display, the default distance is 1920.

If you want to specify a distance that leads to visually consistent results across various densities, use the following formula:

```
float scale = context.getResources().getDisplayMetrics().density;
view.setCameraDistance(distance * scale);
```

The density scale factor of a high density display is 1.5, and 1920 = 1280 * 1.5.

Parameters

distance	The distance in "depth pixels", if negative the opposite value is used

See Also

`setRotationX(float)`
`setRotationY(float)`

public void **setClickable** (boolean clickable) Added in API level 1

Enables or disables click events for this view. When a view is clickable it will change its state to "pressed" on every click. Subclasses should set the view clickable to visually react to user's clicks.

Related XML Attributes

android:clickable

Parameters

clickable	true to make the view clickable, false otherwise

See Also

`isClickable()`

public void **setClipBounds** (Rect clipBounds) Added in API level 18

Sets a rectangular area on this view to which the view will be clipped when it is drawn. Setting the value to null will remove the clip bounds and the view will draw normally, using its full bounds.

Parameters

clipBounds	The rectangular area, in the local coordinates of this view, to which future drawing operations will be clipped.

public void **setContentDescription** (CharSequence contentDescription) Added in API level 4

Sets the `View` (/reference/android/view/View.html) description. It briefly describes the view and is primarily used for accessibility support. Set this property to enable better accessibility support for your application. This is especially true for views that do not have textual representation (For example, ImageButton).

Related XML Attributes
android:contentDescription

Parameters

contentDescription	The content description.

public void **setDrawingCacheBackgroundColor** (int color) Added in API level 1

Setting a solid background color for the drawing cache's bitmaps will improve performance and memory usage. Note, though that this should only be used if this view will always be drawn on top of a solid color.

Parameters

color	The background color to use for the drawing cache's bitmap

See Also
`setDrawingCacheEnabled(boolean)`
`buildDrawingCache()`
`getDrawingCache()`

public void **setDrawingCacheEnabled** (boolean enabled) Added in API level 1

Enables or disables the drawing cache. When the drawing cache is enabled, the next call to `getDrawingCache()` (/reference/android/view/View.html#getDrawingCache()) or `buildDrawingCache()` (/reference/android/view/View.html#buildDrawingCache()) will draw the view in a bitmap. Calling `draw(android.graphics.Canvas)` (/reference/android/view/View.html#draw(android.graphics.Canvas)) will not draw from the cache when the cache is enabled. To benefit from the cache, you must request the drawing cache by calling `getDrawingCache()` (/reference/android/view/View.html#getDrawingCache()) and draw it on screen if the returned bitmap is not null.

Enabling the drawing cache is similar to `setting a layer` (/reference/android/view/View.html#setLayerType(int, android.graphics.Paint)) when hardware acceleration is turned off. When hardware acceleration is turned on, enabling the drawing cache has no effect on rendering because the system uses a different mechanism for acceleration which ignores the flag. If you want to use a Bitmap for the view, even when hardware acceleration is enabled, see `setLayerType(int, android.graphics.Paint)` (/reference/android/view/View.html#setLayerType(int, android.graphics.Paint)) for information on how to enable software and hardware layers.

This API can be used to manually generate a bitmap copy of this view, by setting the flag to `true` and calling `getDrawingCache()` (/reference/android/view/View.html#getDrawingCache()).

Parameters

enabled	true to enable the drawing cache, false otherwise

See Also
`isDrawingCacheEnabled()`
`getDrawingCache()`
`buildDrawingCache()`
`setLayerType(int, android.graphics.Paint)`

public void **setDrawingCacheQuality** (int quality) Added in API level 1

Set the drawing cache quality of this view. This value is used only when the drawing cache is enabled

Related XML Attributes
android:drawingCacheQuality

Parameters

quality	One of `DRAWING_CACHE_QUALITY_AUTO`, `DRAWING_CACHE_QUALITY_LOW`, or `DRAWING_CACHE_QUALITY_HIGH`

See Also
`getDrawingCacheQuality()`
`setDrawingCacheEnabled(boolean)`
`isDrawingCacheEnabled()`

public void **setDuplicateParentStateEnabled** (boolean enabled) Added in API level 1

Enables or disables the duplication of the parent's state into this view. When duplication is enabled, this view gets its drawable state from its parent rather than from its own internal properties.

Note: in the current implementation, setting this property to true after the view was added to a ViewGroup might have no effect at all. This property should always be used from XML or set to true before adding this view to a ViewGroup.

Note: if this view's parent addStateFromChildren property is enabled and this property is enabled, an exception will be thrown.

Note: if the child view uses and updates additionnal states which are unknown to the parent, these states should not be affected by this method.

Parameters

enabled	True to enable duplication of the parent's drawable state, false to disable it.

See Also
`getDrawableState()`
`isDuplicateParentStateEnabled()`

public void **setEnabled** (boolean enabled) Added in API level 1

Set the enabled state of this view. The interpretation of the enabled state varies by subclass.

Parameters

enabled	True if this view is enabled, false otherwise.

public void **setFadingEdgeLength** (int length) Added in API level 1

Set the size of the faded edge used to indicate that more content in this view is available. Will not change whether the fading edge is enabled;

use `setVerticalFadingEdgeEnabled(boolean)` (/reference/android/view/View.html#setVerticalFadingEdgeEnabled(boolean)) or `setHorizontalFadingEdgeEnabled(boolean)` (/reference/android/view/View.html#setHorizontalFadingEdgeEnabled(boolean)) to enable the fading edge for the vertical or horizontal fading edges.

Parameters

length	The size in pixels of the faded edge used to indicate that more content in this view is visible.

public void **setFilterTouchesWhenObscured** (boolean enabled)

Added in API level 9

Sets whether the framework should discard touches when the view's window is obscured by another visible window. Refer to the `View` (/reference/android/view/View.html) security documentation for more details.

Related XML Attributes
android:filterTouchesWhenObscured

Parameters

enabled	True if touch filtering should be enabled.

See Also
`getFilterTouchesWhenObscured()`

public void **setFitsSystemWindows** (boolean fitSystemWindows)

Added in API level 14

Sets whether or not this view should account for system screen decorations such as the status bar and inset its content; that is, controlling whether the default implementation of `fitSystemWindows(Rect)` (/reference/android/view/View.html#fitSystemWindows(android.graphics.Rect)) will be executed. See that method for more details.

Note that if you are providing your own implementation of `fitSystemWindows(Rect)` (/reference/android /view/View.html#fitSystemWindows(android.graphics.Rect)), then there is no need to set this flag to true -- your implementation will be overriding the default implementation that checks this flag.

Related XML Attributes
android:fitsSystemWindows

Parameters

fitSystemWindows	If true, then the default implementation of `fitSystemWindows(Rect)` will be executed.

See Also
`getFitsSystemWindows()`
`fitSystemWindows(Rect)`
`setSystemUiVisibility(int)`

public void **setFocusable** (boolean focusable)

Added in API level 1

Set whether this view can receive the focus. Setting this to false will also ensure that this view is not focusable in touch mode.

Related XML Attributes
android:focusable

Parameters

focusable	If true, this view can receive the focus.

See Also
`setFocusableInTouchMode(boolean)`

public void **setFocusableInTouchMode** (boolean focusableInTouchMode)

Added in API level 1

Set whether this view can receive focus while in touch mode. Setting this to true will also ensure that this view is focusable.

Related XML Attributes
android:focusableInTouchMode

Parameters

focusableInTouchMode	If true, this view can receive the focus while in touch mode.

See Also
`setFocusable(boolean)`

public void **setHapticFeedbackEnabled** (boolean hapticFeedbackEnabled)

Added in API level 3

Set whether this view should have haptic feedback for events such as long presses.

You may wish to disable haptic feedback if your view already controls its own haptic feedback.

Related XML Attributes
android:hapticFeedbackEnabled

Parameters

hapticFeedbackEnabled	whether haptic feedback enabled for this view.

See Also
`isHapticFeedbackEnabled()`
`performHapticFeedback(int)`

public void **setHasTransientState** (boolean hasTransientState)

Added in API level 16

Set whether this view is currently tracking transient state that the framework should attempt to preserve when possible. This flag is reference counted, so every call to setHasTransientState(true) should be paired with a later call to setHasTransientState(false).

A view with transient state cannot be trivially rebound from an external data source, such as an adapter binding item views in a list. This may be because the view is performing an animation, tracking user selection of content, or similar.

Parameters

hasTransientState	true if this view has transient state

public void **setHorizontalFadingEdgeEnabled** (boolean horizontalFadingEdgeEnabled)

Added in API level 1

Define whether the horizontal edges should be faded when this view is scrolled horizontally.

Related XML Attributes
android:requiresFadingEdge

Parameters

horizontalFadingEdgeEnabled	true if the horizontal edges should be faded when the view is scrolled horizontally

See Also
`isHorizontalFadingEdgeEnabled()`

public void **setHorizontalScrollBarEnabled** (boolean horizontalScrollBarEnabled) Added in API level 1

Define whether the horizontal scrollbar should be drawn or not. The scrollbar is not drawn by default.

Parameters

horizontalScrollBarEnabled	true if the horizontal scrollbar should be painted

See Also
`isHorizontalScrollBarEnabled()`

public void **setHovered** (boolean hovered) Added in API level 14

Sets whether the view is currently hovered.

Calling this method also changes the drawable state of the view. This enables the view to react to hover by using different drawable resources to change its appearance.

The `onHoverChanged(boolean)` (/reference/android/view/View.html#onHoverChanged(boolean)) method is called when the hovered state changes.

Parameters

hovered	True if the view is hovered.

See Also
`isHovered()`
`onHoverChanged(boolean)`

public void **setId** (int id) Added in API level 1

Sets the identifier for this view. The identifier does not have to be unique in this view's hierarchy. The identifier should be a positive number.

Related XML Attributes
android:id

Parameters

id	a number used to identify the view

See Also
`NO_ID`
`getId()`
`findViewById(int)`

public void **setImportantForAccessibility** (int mode) Added in API level 16

Sets how to determine whether this view is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen.

Related XML Attributes
android:importantForAccessibility

Parameters

mode	How to determine whether this view is important for accessibility.

See Also
`IMPORTANT_FOR_ACCESSIBILITY_YES`
`IMPORTANT_FOR_ACCESSIBILITY_NO`
`IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS`
`IMPORTANT_FOR_ACCESSIBILITY_AUTO`

public void **setKeepScreenOn** (boolean keepScreenOn) Added in API level 1

Controls whether the screen should remain on, modifying the value of `KEEP_SCREEN_ON` (/reference/android/view/View.html#KEEP_SCREEN_ON).

Related XML Attributes
android:keepScreenOn

Parameters

keepScreenOn	Supply true to set `KEEP_SCREEN_ON`.

See Also
`getKeepScreenOn()`

public void **setLabelFor** (int id) Added in API level 17

Sets the id of a view for which this view serves as a label for accessibility purposes.

Parameters

id	The labeled view id.

public void **setLayerPaint** (Paint paint) Added in API level 17

Updates the `Paint` (/reference/android/graphics/Paint.html) object used with the current layer (used only if the current layer type is not set to `LAYER_TYPE_NONE` (/reference/android/view/View.html#LAYER_TYPE_NONE)). Changed properties of the Paint provided to `setLayerType(int, android.graphics.Paint)` (/reference/android/view/View.html#setLayerType(int, android.graphics.Paint)) will be used the next time the View is redrawn, but `setLayerPaint(android.graphics.Paint)` (/reference/android/view/View.html#setLayerPaint(android.graphics.Paint)) must be called to ensure that the view gets redrawn immediately.

A layer is associated with an optional Paint (/reference/android/graphics/Paint.html) instance that controls how the layer is composed on screen. The following properties of the paint are taken into account when composing the layer:

- Translucency (alpha)
- Blending mode
- Color filter

If this view has an alpha value set to < 1.0 by calling setAlpha(float) (/reference/android/view/View.html#setAlpha(float)), the alpha value of the layer's paint is superceded by this view's alpha value.

Parameters

paint	The paint used to compose the layer. This argument is optional and can be null. It is ignored when the layer type is LAYER_TYPE_NONE

See Also

setLayerType(int, android.graphics.Paint)

public void **setLayerType** (int layerType, Paint paint)

Added in API level 11

Specifies the type of layer backing this view. The layer can be LAYER_TYPE_NONE (/reference/android/view/View.html#LAYER_TYPE_NONE), LAYER_TYPE_SOFTWARE (/reference/android/view/View.html#LAYER_TYPE_SOFTWARE) or LAYER_TYPE_HARDWARE (/reference/android /view/View.html#LAYER_TYPE_HARDWARE).

A layer is associated with an optional Paint (/reference/android/graphics/Paint.html) instance that controls how the layer is composed on screen. The following properties of the paint are taken into account when composing the layer:

- Translucency (alpha)
- Blending mode
- Color filter

If this view has an alpha value set to < 1.0 by calling setAlpha(float) (/reference/android/view/View.html#setAlpha(float)), the alpha value of the layer's paint is superceded by this view's alpha value.

Refer to the documentation of LAYER_TYPE_NONE (/reference/android/view/View.html#LAYER_TYPE_NONE), LAYER_TYPE_SOFTWARE (/reference /android/view/View.html#LAYER_TYPE_SOFTWARE) and LAYER_TYPE_HARDWARE (/reference/android/view/View.html#LAYER_TYPE_HARDWARE) for more information on when and how to use layers.

Related XML Attributes

android:layerType

Parameters

layerType	The type of layer to use with this view, must be one of LAYER_TYPE_NONE, LAYER_TYPE_SOFTWARE or LAYER_TYPE_HARDWARE
paint	The paint used to compose the layer. This argument is optional and can be null. It is ignored when the layer type is LAYER_TYPE_NONE

See Also

getLayerType()
LAYER_TYPE_NONE
LAYER_TYPE_SOFTWARE
LAYER_TYPE_HARDWARE
setAlpha(float)

public void **setLayoutDirection** (int layoutDirection)

Added in API level 17

Set the layout direction for this view. This will propagate a reset of layout direction resolution to the view's children and resolve layout direction for this view.

Related XML Attributes

android:layoutDirection

Parameters

layoutDirection	the layout direction to set. Should be one of: LAYOUT_DIRECTION_LTR, LAYOUT_DIRECTION_RTL, LAYOUT_DIRECTION_INHERIT, LAYOUT_DIRECTION_LOCALE. Resolution will be done if the value is set to LAYOUT_DIRECTION_INHERIT. The resolution proceeds up the parent chain of the view to get the value. If there is no parent, then it will return the default LAYOUT_DIRECTION_LTR.

public void **setLayoutParams** (ViewGroup.LayoutParams params)

Added in API level 1

Set the layout parameters associated with this view. These supply parameters to the *parent* of this view specifying how it should be arranged. There are many subclasses of ViewGroup.LayoutParams, and these correspond to the different subclasses of ViewGroup that are responsible for arranging their children.

Parameters

params	The layout parameters for this view, cannot be null

public final void **setLeft** (int left)

Added in API level 11

Sets the left position of this view relative to its parent. This method is meant to be called by the layout system and should not generally be called otherwise, because the property may be changed at any time by the layout.

Parameters

left	The bottom of this view, in pixels.

public void **setLongClickable** (boolean longClickable)

Added in API level 1

Enables or disables long click events for this view. When a view is long clickable it reacts to the user holding down the button for a longer duration than a tap. This event can either launch the listener or a context menu.

Related XML Attributes

android:longClickable

Parameters

longClickable	true to make the view long clickable, false otherwise

See Also

isLongClickable()

public void **setMinimumHeight** (int minHeight)

Added in API level 1

Sets the minimum height of the view. It is not guaranteed the view will be able to achieve this minimum height (for example, if its parent layout constrains it with less available height).

Related XML Attributes
android:minHeight

Parameters

minHeight	The minimum height the view will try to be.

See Also
getMinimumHeight()

public void **setMinimumWidth** (int minWidth)

Added in API level 1

Sets the minimum width of the view. It is not guaranteed the view will be able to achieve this minimum width (for example, if its parent layout constrains it with less available width).

Related XML Attributes
android:minWidth

Parameters

minWidth	The minimum width the view will try to be.

See Also
getMinimumWidth()

public void **setNextFocusDownId** (int nextFocusDownId)

Added in API level 1

Sets the id of the view to use when the next focus is `FOCUS_DOWN` (/reference/android/view/View.html#FOCUS_DOWN).

Related XML Attributes
android:nextFocusDown

Parameters

nextFocusDownId	The next focus ID, or `NO_ID` if the framework should decide automatically.

public void **setNextFocusForwardId** (int nextFocusForwardId)

Added in API level 11

Sets the id of the view to use when the next focus is `FOCUS_FORWARD` (/reference/android/view/View.html#FOCUS_FORWARD).

Related XML Attributes
android:nextFocusForward

Parameters

nextFocusForwardId	The next focus ID, or `NO_ID` if the framework should decide automatically.

public void **setNextFocusLeftId** (int nextFocusLeftId)

Added in API level 1

Sets the id of the view to use when the next focus is `FOCUS_LEFT` (/reference/android/view/View.html#FOCUS_LEFT).

Related XML Attributes
android:nextFocusLeft

Parameters

nextFocusLeftId	The next focus ID, or `NO_ID` if the framework should decide automatically.

public void **setNextFocusRightId** (int nextFocusRightId)

Added in API level 1

Sets the id of the view to use when the next focus is `FOCUS_RIGHT` (/reference/android/view/View.html#FOCUS_RIGHT).

Related XML Attributes
android:nextFocusRight

Parameters

nextFocusRightId	The next focus ID, or `NO_ID` if the framework should decide automatically.

public void **setNextFocusUpId** (int nextFocusUpId)

Added in API level 1

Sets the id of the view to use when the next focus is `FOCUS_UP` (/reference/android/view/View.html#FOCUS_UP).

Related XML Attributes
android:nextFocusUp

Parameters

nextFocusUpId	The next focus ID, or `NO_ID` if the framework should decide automatically.

public void **setOnClickListener** (View.OnClickListener l)

Added in API level 1

Register a callback to be invoked when this view is clicked. If this view is not clickable, it becomes clickable.

Parameters

l	The callback that will run

See Also
setClickable(boolean)

public void **setOnCreateContextMenuListener** (View.OnCreateContextMenuListener l)

Added in API level 1

Register a callback to be invoked when the context menu for this view is being built. If this view is not long clickable, it becomes long clickable.

Parameters

l	The callback that will run

public void **setOnDragListener** (View.OnDragListener l) Added in API level 11

Register a drag event listener callback object for this View. The parameter is an implementation of `View.OnDragListener` (/reference/android/view/View.OnDragListener.html). To send a drag event to a View, the system calls the `onDrag(View, DragEvent)` (/reference/android/view/View.OnDragListener.html#onDrag(android.view.View, android.view.DragEvent)) method.

Parameters

l	An implementation of `View.OnDragListener`.

public void **setOnFocusChangeListener** (View.OnFocusChangeListener l) Added in API level 1

Register a callback to be invoked when focus of this view changed.

Parameters

l	The callback that will run.

public void **setOnGenericMotionListener** (View.OnGenericMotionListener l) Added in API level 12

Register a callback to be invoked when a generic motion event is sent to this view.

Parameters

l	the generic motion listener to attach to this view

public void **setOnHoverListener** (View.OnHoverListener l) Added in API level 14

Register a callback to be invoked when a hover event is sent to this view.

Parameters

l	the hover listener to attach to this view

public void **setOnKeyListener** (View.OnKeyListener l) Added in API level 1

Register a callback to be invoked when a hardware key is pressed in this view. Key presses in software input methods will generally not trigger the methods of this listener.

Parameters

l	the key listener to attach to this view

public void **setOnLongClickListener** (View.OnLongClickListener l) Added in API level 1

Register a callback to be invoked when this view is clicked and held. If this view is not long clickable, it becomes long clickable.

Parameters

l	The callback that will run

See Also
`setLongClickable(boolean)`

public void **setOnSystemUiVisibilityChangeListener** (View.OnSystemUiVisibilityChangeListener l) Added in API level 11

Set a listener to receive callbacks when the visibility of the system bar changes.

Parameters

l	The `View.OnSystemUiVisibilityChangeListener` to receive callbacks.

public void **setOnTouchListener** (View.OnTouchListener l) Added in API level 1

Register a callback to be invoked when a touch event is sent to this view.

Parameters

l	the touch listener to attach to this view

public void **setOverScrollMode** (int overScrollMode) Added in API level 9

Set the over-scroll mode for this view. Valid over-scroll modes are `OVER_SCROLL_ALWAYS` (/reference/android/view/View.html#OVER_SCROLL_ALWAYS) (default), `OVER_SCROLL_IF_CONTENT_SCROLLS` (/reference/android/view/View.html#OVER_SCROLL_IF_CONTENT_SCROLLS) (allow over-scrolling only if the view content is larger than the container), or `OVER_SCROLL_NEVER` (/reference/android/view/View.html#OVER_SCROLL_NEVER). Setting the over-scroll mode of a view will have an effect only if the view is capable of scrolling.

Parameters

overScrollMode	The new over-scroll mode for this view.

public void **setPadding** (int left, int top, int right, int bottom) Added in API level 1

Sets the padding. The view may add on the space required to display the scrollbars, depending on the style and visibility of the scrollbars. So the values returned from `getPaddingLeft()` (/reference/android/view/View.html#getPaddingLeft()), `getPaddingTop()` (/reference/android/view/View.html#getPaddingTop()), `getPaddingRight()` (/reference/android/view/View.html#getPaddingRight()) and `getPaddingBottom()` (/reference/android/view/View.html#getPaddingBottom()) may be different from the values set in this call.

Related XML Attributes
android:padding
android:paddingBottom
android:paddingLeft
android:paddingRight
android:paddingTop

Parameters

left	the left padding in pixels
top	the top padding in pixels
right	the right padding in pixels
bottom	the bottom padding in pixels

public void **setPaddingRelative** (int start, int top, int end, int bottom)

Added in API level 17

Sets the relative padding. The view may add on the space required to display the scrollbars, depending on the style and visibility of the scrollbars. So the values returned from `getPaddingStart()` (/reference/android/view/View.html#getPaddingStart()), `getPaddingTop()` (/reference/android/view/View.html#getPaddingTop()), `getPaddingEnd()` (/reference/android/view/View.html#getPaddingEnd()) and `getPaddingBottom()` (/reference/android/view/View.html#getPaddingBottom()) may be different from the values set in this call.

Related XML Attributes

android:padding
android:paddingBottom
android:paddingStart
android:paddingEnd
android:paddingTop

Parameters

start	the start padding in pixels
top	the top padding in pixels
end	the end padding in pixels
bottom	the bottom padding in pixels

public void **setPivotX** (float pivotX)

Added in API level 11

Sets the x location of the point around which the view is `rotated` (/reference/android/view/View.html#setRotation(float)) and `scaled` (/reference/android/view/View.html#setScaleX(float)). By default, the pivot point is centered on the object. Setting this property disables this behavior and causes the view to use only the explicitly set pivotX and pivotY values.

Related XML Attributes

android:transformPivotX

Parameters

pivotX	The x location of the pivot point.

See Also

`getRotation()`
`getScaleX()`
`getScaleY()`
`getPivotY()`

public void **setPivotY** (float pivotY)

Added in API level 11

Sets the y location of the point around which the view is `rotated` (/reference/android/view/View.html#setRotation(float)) and `scaled` (/reference/android/view/View.html#setScaleY(float)). By default, the pivot point is centered on the object. Setting this property disables this behavior and causes the view to use only the explicitly set pivotX and pivotY values.

Related XML Attributes

android:transformPivotY

Parameters

pivotY	The y location of the pivot point.

See Also

`getRotation()`
`getScaleX()`
`getScaleY()`
`getPivotY()`

public void **setPressed** (boolean pressed)

Added in API level 1

Sets the pressed state for this view.

Parameters

pressed	Pass true to set the View's internal state to "pressed", or false to reverts the View's internal state from a previously set "pressed" state.

See Also

`isClickable()`
`setClickable(boolean)`

public final void **setRight** (int right)

Added in API level 11

Sets the right position of this view relative to its parent. This method is meant to be called by the layout system and should not generally be called otherwise, because the property may be changed at any time by the layout.

Parameters

right	The bottom of this view, in pixels.

public void **setRotation** (float rotation)

Added in API level 11

Sets the degrees that the view is rotated around the pivot point. Increasing values result in clockwise rotation.

Related XML Attributes

android:rotation

Parameters

rotation	The degrees of rotation.

See Also

`getRotation()`
`getPivotX()`
`getPivotY()`
`setRotationX(float)`
`setRotationY(float)`

public void **setRotationX** (float rotationX)

Added in API level 11

Sets the degrees that the view is rotated around the horizontal axis through the pivot point. Increasing values result in clockwise rotation from the viewpoint of looking down the x axis. When rotating large views, it is recommended to adjust the camera distance accordingly. Refer to `setCameraDistance(float)` (/reference/android/view/View.html#setCameraDistance(float)) for more information.

Related XML Attributes

android:rotationX

Parameters

rotationX	The degrees of X rotation.

See Also

`getRotationX()`
`getPivotX()`
`getPivotY()`
`setRotation(float)`
`setRotationY(float)`
`setCameraDistance(float)`

public void **setRotationY** (float rotationY)

Added in API level 11

Sets the degrees that the view is rotated around the vertical axis through the pivot point. Increasing values result in counter-clockwise rotation from the viewpoint of looking down the y axis. When rotating large views, it is recommended to adjust the camera distance accordingly. Refer to `setCameraDistance(float)` (/reference/android/view/View.html#setCameraDistance(float)) for more information.

Related XML Attributes

android:rotationY

Parameters

rotationY	The degrees of Y rotation.

See Also

`getRotationY()`
`getPivotX()`
`getPivotY()`
`setRotation(float)`
`setRotationX(float)`
`setCameraDistance(float)`

public void **setSaveEnabled** (boolean enabled)

Added in API level 1

Controls whether the saving of this view's state is enabled (that is, whether its `onSaveInstanceState()` (/reference/android/view/View.html#onSaveInstanceState()) method will be called). Note that even if freezing is enabled, the view still must have an id assigned to it (via `setId(int)` (/reference/android/view/View.html#setId(int))) for its state to be saved. This flag can only disable the saving of this view; any child views may still have their state saved.

Related XML Attributes

android:saveEnabled

Parameters

enabled	Set to false to *disable* state saving, or true (the default) to allow it.

See Also

`isSaveEnabled()`
`setId(int)`
`onSaveInstanceState()`

public void **setSaveFromParentEnabled** (boolean enabled)

Added in API level 11

Controls whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent. The default is true; if false, these views will not be saved unless `saveHierarchyState(SparseArray)` (/reference/android/view/View.html#saveHierarchyState(android.util.SparseArray<android.os.Parcelable>)) is called directly on this view.

Parameters

enabled	Set to false to *disable* state saving, or true (the default) to allow it.

See Also

`isSaveFromParentEnabled()`
`setId(int)`
`onSaveInstanceState()`

public void **setScaleX** (float scaleX)

Added in API level 11

Sets the amount that the view is scaled in x around the pivot point, as a proportion of the view's unscaled width. A value of 1 means that no scaling is applied.

Related XML Attributes

android:scaleX

Parameters

scaleX	The scaling factor.

See Also

`getPivotX()`
`getPivotY()`

public void **setScaleY** (float scaleY)

Added in API level 11

Sets the amount that the view is scaled in Y around the pivot point, as a proportion of the view's unscaled width. A value of 1 means that no scaling is applied.

Related XML Attributes

android:scaleY

Parameters

scaleY	The scaling factor.

See Also

getPivotX()
getPivotY()

public void **setScrollBarDefaultDelayBeforeFade** (int scrollBarDefaultDelayBeforeFade)

Added in API level 16

Define the delay before scrollbars fade.

Related XML Attributes

android:scrollbarDefaultDelayBeforeFade

Parameters

scrollBarDefaultDelayBeforeFade	- the delay before scrollbars fade

public void **setScrollBarFadeDuration** (int scrollBarFadeDuration)

Added in API level 16

Define the scrollbar fade duration.

Related XML Attributes

android:scrollbarFadeDuration

Parameters

scrollBarFadeDuration	- the scrollbar fade duration

public void **setScrollBarSize** (int scrollBarSize)

Added in API level 16

Define the scrollbar size.

Related XML Attributes

android:scrollbarSize

Parameters

scrollBarSize	- the scrollbar size

public void **setScrollBarStyle** (int style)

Added in API level 1

Specify the style of the scrollbars. The scrollbars can be overlaid or inset. When inset, they add to the padding of the view. And the scrollbars can be drawn inside the padding area or on the edge of the view. For example, if a view has a background drawable and you want to draw the scrollbars inside the padding specified by the drawable, you can use SCROLLBARS_INSIDE_OVERLAY or SCROLLBARS_INSIDE_INSET. If you want them to appear at the edge of the view, ignoring the padding, then you can use SCROLLBARS_OUTSIDE_OVERLAY or SCROLLBARS_OUTSIDE_INSET.

Related XML Attributes

android:scrollbarStyle

Parameters

style	the style of the scrollbars. Should be one of SCROLLBARS_INSIDE_OVERLAY, SCROLLBARS_INSIDE_INSET, SCROLLBARS_OUTSIDE_OVERLAY or SCROLLBARS_OUTSIDE_INSET.

See Also

SCROLLBARS_INSIDE_OVERLAY
SCROLLBARS_INSIDE_INSET
SCROLLBARS_OUTSIDE_OVERLAY
SCROLLBARS_OUTSIDE_INSET

public void **setScrollContainer** (boolean isScrollContainer)

Added in API level 3

Change whether this view is one of the set of scrollable containers in its window. This will be used to determine whether the window can resize or must pan when a soft input area is open -- scrollable containers allow the window to use resize mode since the container will appropriately shrink.

Related XML Attributes

android:isScrollContainer

public void **setScrollX** (int value)

Added in API level 14

Set the horizontal scrolled position of your view. This will cause a call to `onScrollChanged(int, int, int, int)` (/reference/android/view/View.html#onScrollChanged(int, int, int, int)) and the view will be invalidated.

Parameters

value	the x position to scroll to

public void **setScrollY** (int value)

Added in API level 14

Set the vertical scrolled position of your view. This will cause a call to `onScrollChanged(int, int, int, int)` (/reference/android/view/View.html#onScrollChanged(int, int, int, int)) and the view will be invalidated.

Parameters

value	the y position to scroll to

public void **setScrollbarFadingEnabled** (boolean fadeScrollbars)

Added in API level 5

Define whether scrollbars will fade when the view is not scrolling.

Related XML Attributes

android:fadeScrollbars

Parameters

fadeScrollbars	wheter to enable fading

public void **setSelected** (boolean selected)

Added in API level 1

Changes the selection state of this view. A view can be selected or not. Note that selection is not the same as focus. Views are typically selected in the context of an AdapterView like ListView or GridView; the selected view is the view that is highlighted.

Parameters

selected true if the view must be selected, false otherwise

public void **setSoundEffectsEnabled** (boolean soundEffectsEnabled)

Added in API level 1

Set whether this view should have sound effects enabled for events such as clicking and touching.

You may wish to disable sound effects for a view if you already play sounds, for instance, a dial key that plays dtmf tones.

Related XML Attributes

android:soundEffectsEnabled

Parameters

soundEffectsEnabled	whether sound effects are enabled for this view.

See Also

`isSoundEffectsEnabled()`
`playSoundEffect(int)`

public void **setSystemUiVisibility** (int visibility)

Added in API level 11

Request that the visibility of the status bar or other screen/window decorations be changed.

This method is used to put the over device UI into temporary modes where the user's attention is focused more on the application content, by dimming or hiding surrounding system affordances. This is typically used in conjunction with `Window.FEATURE_ACTION_BAR_OVERLAY` (/reference/android/view/Window.html#FEATURE_ACTION_BAR_OVERLAY), allowing the applications content to be placed behind the action bar (and with these flags other system affordances) so that smooth transitions between hiding and showing them can be done.

Two representative examples of the use of system UI visibility is implementing a content browsing application (like a magazine reader) and a video playing application.

The first code shows a typical implementation of a View in a content browsing application. In this implementation, the application goes into a content-oriented mode by hiding the status bar and action bar, and putting the navigation elements into lights out mode. The user can then interact with content while in this mode. Such an application should provide an easy way for the user to toggle out of the mode (such as to check information in the status bar or access notifications). In the implementation here, this is done simply by tapping on the content.

```java
public static class Content extends ScrollView
        implements View.OnSystemUiVisibilityChangeListener, View.OnClickListener {
    TextView mText;
    TextView mTitleView;
    SeekBar mSeekView;
    boolean mNavVisible;
    int mBaseSystemUiVisibility = SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
            | SYSTEM_UI_FLAG_LAYOUT_STABLE;
    int mLastSystemUiVis;

    Runnable mNavHider = new Runnable() {
        @Override public void run() {
            setNavVisibility(false);
        }
    };

    public Content(Context context, AttributeSet attrs) {
        super(context, attrs);

        mText = new TextView(context);
        mText.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 16);
        mText.setText(context.getString(R.string.alert_dialog_two_buttons2ultra_msg));
        mText.setClickable(false);
        mText.setOnClickListener(this);
        mText.setTextIsSelectable(true);
        addView(mText, new ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT));

        setOnSystemUiVisibilityChangeListener(this);
    }

    public void init(TextView title, SeekBar seek) {
        // This called by the containing activity to supply the surrounding
        // state of the content browser that it will interact with.
        mTitleView = title;
        mSeekView = seek;
        setNavVisibility(true);
    }

    @Override public void onSystemUiVisibilityChange(int visibility) {
        // Detect when we go out of low-profile mode, to also go out
        // of full screen.  We only do this when the low profile mode
        // is changing from its last state, and turning off.
        int diff = mLastSystemUiVis ^ visibility;
        mLastSystemUiVis = visibility;
        if ((diff&SYSTEM_UI_FLAG_LOW_PROFILE) != 0
                && (visibility&SYSTEM_UI_FLAG_LOW_PROFILE) == 0) {
            setNavVisibility(true);
        }
    }

    @Override protected void onWindowVisibilityChanged(int visibility) {
        super.onWindowVisibilityChanged(visibility);

        // When we become visible, we show our navigation elements briefly
        // before hiding them.
        setNavVisibility(true);
        getHandler().postDelayed(mNavHider, 2000);
    }

    @Override protected void onScrollChanged(int l, int t, int oldl, int oldt) {
        super.onScrollChanged(l, t, oldl, oldt);

        // When the user scrolls, we hide navigation elements.
        setNavVisibility(false);
    }

    @Override public void onClick(View v) {
        // When the user clicks, we toggle the visibility of navigation elements.
        int curVis = getSystemUiVisibility();
        setNavVisibility((curVis&SYSTEM_UI_FLAG_LOW_PROFILE) != 0);
    }

    void setBaseSystemUiVisibility(int visibility) {
        mBaseSystemUiVisibility = visibility;
    }

    void setNavVisibility(boolean visible) {
        int newVis = mBaseSystemUiVisibility;
        if (!visible) {
            newVis |= SYSTEM_UI_FLAG_LOW_PROFILE | SYSTEM_UI_FLAG_FULLSCREEN;
        }
        final boolean changed = newVis == getSystemUiVisibility();

        // Unschedule any pending event to hide navigation if we are
        // changing the visibility, or making the UI visible.
        if (changed || visible) {
            Handler h = getHandler();
            if (h != null) {
                h.removeCallbacks(mNavHider);
            }
        }

        // Set the new desired visibility.
        setSystemUiVisibility(newVis);
        mTitleView.setVisibility(visible ? VISIBLE : INVISIBLE);
        mSeekView.setVisibility(visible ? VISIBLE : INVISIBLE);
    }
}
```

This second code sample shows a typical implementation of a View in a video playing application. In this situation, while the video is playing the application would like to go into a complete full-screen mode, to use as much of the display as possible for the video. When in this state the user can not interact with the application; the system intercepts touching on the screen to pop the UI out of full screen mode. See `fitSystemWindows(Rect)` (/reference/android/view/View.html#fitSystemWindows(android.graphics.Rect)) for a sample layout that goes with this code.

```
public static class Content extends ImageView implements
        View.OnSystemUiVisibilityChangeListener, View.OnClickListener,
        ActionBar.OnMenuVisibilityListener {
    Activity mActivity;
    TextView mTitleView;
    Button mPlayButton;
    SeekBar mSeekView;
    boolean mAddedMenuListener;
    boolean mMenusOpen;
    boolean mPaused;
    boolean mNavVisible;
    int mLastSystemUiVis;

    Runnable mNavHider = new Runnable() {
        @Override public void run() {
            setNavVisibility(false);
        }
    };

    public Content(Context context, AttributeSet attrs) {
        super(context, attrs);
        setOnSystemUiVisibilityChangeListener(this);
        setOnClickListener(this);
    }

    public void init(Activity activity, TextView title, Button playButton,
            SeekBar seek) {
        // This called by the containing activity to supply the surrounding
        // state of the video player that it will interact with.
        mActivity = activity;
        mTitleView = title;
        mPlayButton = playButton;
        mSeekView = seek;
        mPlayButton.setOnClickListener(this);
        setPlayPaused(true);
    }

    @Override protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (mActivity != null) {
            mAddedMenuListener = true;
            mActivity.getActionBar().addOnMenuVisibilityListener(this);
        }
    }

    @Override protected void onDetachedFromWindow() {
        super.onDetachedFromWindow();
        if (mAddedMenuListener) {
            mActivity.getActionBar().removeOnMenuVisibilityListener(this);
        }
    }

    @Override public void onSystemUiVisibilityChange(int visibility) {
        // Detect when we go out of nav-hidden mode, to clear our state
        // back to having the full UI chrome up.  Only do this when
        // the state is changing and nav is no longer hidden.
        int diff = mLastSystemUiVis ^ visibility;
        mLastSystemUiVis = visibility;
        if ((diff&SYSTEM_UI_FLAG_HIDE_NAVIGATION) != 0
                && (visibility&SYSTEM_UI_FLAG_HIDE_NAVIGATION) == 0) {
            setNavVisibility(true);
        }
    }

    @Override protected void onWindowVisibilityChanged(int visibility) {
        super.onWindowVisibilityChanged(visibility);

        // When we become visible or invisible, play is paused.
        setPlayPaused(true);
    }

    @Override public void onClick(View v) {
        if (v == mPlayButton) {
            // Clicking on the play/pause button toggles its state.
            setPlayPaused(!mPaused);
        } else {
            // Clicking elsewhere makes the navigation visible.
            setNavVisibility(true);
        }
    }

    @Override public void onMenuVisibilityChanged(boolean isVisible) {
        mMenusOpen = isVisible;
        setNavVisibility(true);
    }

    void setPlayPaused(boolean paused) {
        mPaused = paused;
        mPlayButton.setText(paused ? R.string.play : R.string.pause);
        setKeepScreenOn(!paused);
        setNavVisibility(true);
    }

    void setNavVisibility(boolean visible) {
        int newVis = SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                | SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                | SYSTEM_UI_FLAG_LAYOUT_STABLE;
        if (!visible) {
            newVis |= SYSTEM_UI_FLAG_LOW_PROFILE | SYSTEM_UI_FLAG_FULLSCREEN
                    | SYSTEM_UI_FLAG_HIDE_NAVIGATION;
        }

        // If we are now visible, schedule a timer for us to go invisible.
        if (visible) {
            Handler h = getHandler();
            if (h != null) {
                h.removeCallbacks(mNavHider);
                if (!mMenusOpen && !mPaused) {
                    // If the menus are open or play is paused, we will not auto-hide.
                    h.postDelayed(mNavHider, 3000);
                }
            }
        }

        // Set the new desired visibility.
        setSystemUiVisibility(newVis);
        mTitleView.setVisibility(visible ? VISIBLE : INVISIBLE);
        mPlayButton.setVisibility(visible ? VISIBLE : INVISIBLE);
        mSeekView.setVisibility(visible ? VISIBLE : INVISIBLE);
    }
}
```

Parameters

visibility	Bitwise-or of flags `SYSTEM_UI_FLAG_LOW_PROFILE`, `SYSTEM_UI_FLAG_HIDE_NAVIGATION`, `SYSTEM_UI_FLAG_FULLSCREEN`, `SYSTEM_UI_FLAG_LAYOUT_STABLE`, `SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION`, `SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN`, `SYSTEM_UI_FLAG_IMMERSIVE`, and `SYSTEM_UI_FLAG_IMMERSIVE_STICKY`.

public void **setTag** (int key, Object tag) Added in API level 4

Sets a tag associated with this view and a key. A tag can be used to mark a view in its hierarchy and does not have to be unique within the hierarchy. Tags can also be used to store data within a view without resorting to another data structure. The specified key should be an id declared in the resources of the application to ensure it is unique (see the ID resource type (/guide/topics/resources/more-resources.html#Id")). Keys identified as belonging to the Android framework or not associated with any package will cause an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException.html) to be thrown.

Parameters

key	The key identifying the tag
tag	An Object to tag the view with

Throws

IllegalArgumentException	If they specified key is not valid

See Also

`setTag(Object)`
`getTag(int)`

public void **setTag** (Object tag) Added in API level 1

Sets the tag associated with this view. A tag can be used to mark a view in its hierarchy and does not have to be unique within the hierarchy. Tags can also be used to store data within a view without resorting to another data structure.

Parameters

tag	an Object to tag the view with

See Also

`getTag()`
`setTag(int, Object)`

public void **setTextAlignment** (int textAlignment) Added in API level 17

Set the text alignment.

Related XML Attributes

android:textAlignment

Parameters

textAlignment	The text alignment to set. Should be one of `TEXT_ALIGNMENT_INHERIT`, `TEXT_ALIGNMENT_GRAVITY`, `TEXT_ALIGNMENT_CENTER`, `TEXT_ALIGNMENT_TEXT_START`, `TEXT_ALIGNMENT_TEXT_END`, `TEXT_ALIGNMENT_VIEW_START`, `TEXT_ALIGNMENT_VIEW_END` Resolution will be done if the value is set to TEXT_ALIGNMENT_INHERIT. The resolution proceeds up the parent chain of the view to get the value. If there is no parent, then it will return the default `TEXT_ALIGNMENT_GRAVITY`.

public void **setTextDirection** (int textDirection) Added in API level 17

Set the text direction.

Related XML Attributes

android:textDirection

Parameters

textDirection	the direction to set. Should be one of: `TEXT_DIRECTION_INHERIT`, `TEXT_DIRECTION_FIRST_STRONG` `TEXT_DIRECTION_ANY_RTL`, `TEXT_DIRECTION_LTR`, `TEXT_DIRECTION_RTL`, `TEXT_DIRECTION_LOCALE` Resolution will be done if the value is set to TEXT_DIRECTION_INHERIT. The resolution proceeds up the parent chain of the view to get the value. If there is no parent, then it will return the default `TEXT_DIRECTION_FIRST_STRONG`.

public final void **setTop** (int top) Added in API level 11

Sets the top position of this view relative to its parent. This method is meant to be called by the layout system and should not generally be called otherwise, because the property may be changed at any time by the layout.

Parameters

top	The top of this view, in pixels.

public void **setTouchDelegate** (TouchDelegate delegate) Added in API level 1

Sets the TouchDelegate for this View.

public void **setTranslationX** (float translationX) Added in API level 11

Sets the horizontal location of this view relative to its `left` (/reference/android/view/View.html#getLeft()) position. This effectively positions the object post-layout, in addition to wherever the object's layout placed it.

Related XML Attributes

android:translationX

Parameters

translationX	The horizontal position of this view relative to its left position, in pixels.

public void **setTranslationY** (float translationY) Added in API level 11

Sets the vertical location of this view relative to its `top` (/reference/android/view/View.html#getTop()) position. This effectively positions the object post-layout, in addition to wherever the object's layout placed it.

Related XML Attributes

android:translationY

Parameters

translationY	The vertical position of this view relative to its top position, in pixels.

public void **setVerticalFadingEdgeEnabled** (boolean verticalFadingEdgeEnabled) Added in API level 1

Define whether the vertical edges should be faded when this view is scrolled vertically.

Related XML Attributes

android:requiresFadingEdge

Parameters

verticalFadingEdgeEnabled	true if the vertical edges should be faded when the view is scrolled vertically

See Also

`isVerticalFadingEdgeEnabled()`

public void **setVerticalScrollBarEnabled** (boolean verticalScrollBarEnabled) Added in API level 1

Define whether the vertical scrollbar should be drawn or not. The scrollbar is not drawn by default.

Parameters

verticalScrollBarEnabled	true if the vertical scrollbar should be painted

See Also

`isVerticalScrollBarEnabled()`

public void **setVerticalScrollbarPosition** (int position) Added in API level 11

Set the position of the vertical scroll bar. Should be one of `SCROLLBAR_POSITION_DEFAULT` (/reference/android /view/View.html#SCROLLBAR_POSITION_DEFAULT), `SCROLLBAR_POSITION_LEFT` (/reference/android/view/View.html#SCROLLBAR_POSITION_LEFT) or `SCROLLBAR_POSITION_RIGHT` (/reference/android/view/View.html#SCROLLBAR_POSITION_RIGHT).

Parameters

position	Where the vertical scroll bar should be positioned.

public void **setVisibility** (int visibility) Added in API level 1

Set the enabled state of this view.

Related XML Attributes
android:visibility

Parameters

visibility	One of `VISIBLE`, `INVISIBLE`, or `GONE`.

public void **setWillNotCacheDrawing** (boolean willNotCacheDrawing) Added in API level 1

When a View's drawing cache is enabled, drawing is redirected to an offscreen bitmap. Some views, like an ImageView, must be able to bypass this mechanism if they already draw a single bitmap, to avoid unnecessary usage of the memory.

Parameters

willNotCacheDrawing	true if this view does not cache its drawing, false otherwise

public void **setWillNotDraw** (boolean willNotDraw) Added in API level 1

If this view doesn't do any drawing on its own, set this flag to allow further optimizations. By default, this flag is not set on View, but could be set on some View subclasses such as ViewGroup. Typically, if you override `onDraw(android.graphics.Canvas)` (/reference/android /view/View.html#onDraw(android.graphics.Canvas)) you should clear this flag.

Parameters

willNotDraw	whether or not this View draw on its own

public void **setX** (float x) Added in API level 11

Sets the visual x position of this view, in pixels. This is equivalent to setting the `translationX` (/reference/android /view/View.html#setTranslationX(float)) property to be the difference between the x value passed in and the current `left` (/reference/android /view/View.html#getLeft()) property.

Parameters

x	The visual x position of this view, in pixels.

public void **setY** (float y) Added in API level 11

Sets the visual y position of this view, in pixels. This is equivalent to setting the `translationY` (/reference/android /view/View.html#setTranslationY(float)) property to be the difference between the y value passed in and the current `top` (/reference/android /view/View.html#getTop()) property.

Parameters

y	The visual y position of this view, in pixels.

public boolean **showContextMenu** () Added in API level 1

Bring up the context menu for this view.

Returns
Whether a context menu was displayed.

public ActionMode **startActionMode** (ActionMode.Callback callback) Added in API level 11

Start an action mode.

Parameters

callback	Callback that will control the lifecycle of the action mode

Returns
The new action mode if it is started, null otherwise

See Also
`ActionMode`

public void **startAnimation** (Animation animation) Added in API level 1

Start the specified animation now.

Parameters

animation	the animation to start now

public final boolean **startDrag** (ClipData data, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) Added in API level 11

Starts a drag and drop operation. When your application calls this method, it passes a `View.DragShadowBuilder` (/reference/android /view/View.DragShadowBuilder.html) object to the system. The system calls this object's `onProvideShadowMetrics(Point, Point)` (/reference/android/view/View.DragShadowBuilder.html#onProvideShadowMetrics(android.graphics.Point, android.graphics.Point)) to get metrics for the drag shadow, and then calls the object's `onDrawShadow(Canvas)` (/reference/android /view/View.DragShadowBuilder.html#onDrawShadow(android.graphics.Canvas)) to draw the drag shadow itself.

Once the system has the drag shadow, it begins the drag and drop operation by sending drag events to all the View objects in your application that are currently visible. It does this either by calling the View object's drag listener (an implementation of `onDrag()` (/reference/android /view/View.OnDragListener.html#onDrag(android.view.View, android.view.DragEvent)) or by calling the View object's `onDragEvent()` (/reference

/android/view/View.html#onDragEvent(android.view.DragEvent)) method. Both are passed a `DragEvent` (/reference/android/view/DragEvent.html) object that has a `getAction()` (/reference/android/view/DragEvent.html#getAction()) value of `ACTION_DRAG_STARTED` (/reference/android/view/DragEvent.html#ACTION_DRAG_STARTED).

Your application can invoke startDrag() on any attached View object. The View object does not need to be the one used in `View.DragShadowBuilder` (/reference/android/view/View.DragShadowBuilder.html), nor does it need to be related to the View the user selected for dragging.

Parameters

data	A `ClipData` object pointing to the data to be transferred by the drag and drop operation.
shadowBuilder	A `View.DragShadowBuilder` object for building the drag shadow.
myLocalState	An `Object` containing local data about the drag and drop operation. This Object is put into every DragEvent object sent by the system during the current drag. myLocalState is a lightweight mechanism for the sending information from the dragged View to the target Views. For example, it can contain flags that differentiate between a a copy operation and a move operation.
flags	Flags that control the drag and drop operation. No flags are currently defined, so the parameter should be set to 0.

Returns
`true` if the method completes successfully, or `false` if it fails anywhere. Returning `false` means the system was unable to do a drag, and so no drag operation is in progress.

public String **toString** ()
Added in API level 1

Returns a string containing a concise, human-readable description of this object. Subclasses are encouraged to override this method and provide an implementation that takes into account the object's type and data. The default implementation is equivalent to the following expression:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

See Writing a useful `toString` method (/reference/java/lang/Object.html#writing_toString) if you intend implementing your own `toString` method.

Returns
a printable representation of this object.

public void **unscheduleDrawable** (Drawable who)
Added in API level 1

Unschedule any events associated with the given Drawable. This can be used when selecting a new Drawable into a view, so that the previous one is completely unscheduled.

Parameters

who	The Drawable to unschedule.

See Also
`drawableStateChanged()`

public void **unscheduleDrawable** (Drawable who, Runnable what)
Added in API level 1

Cancels a scheduled action on a drawable.

Parameters

who	the recipient of the action
what	the action to cancel

public boolean **willNotCacheDrawing** ()
Added in API level 1

Returns whether or not this View can cache its drawing or not.

Returns
true if this view does not cache its drawing, false otherwise

public boolean **willNotDraw** ()
Added in API level 1

Returns whether or not this View draws on its own.

Returns
true if this view has nothing to draw, false otherwise

Protected Methods

protected boolean **awakenScrollBars** (int startDelay)
Added in API level 5

Trigger the scrollbars to draw. When invoked this method starts an animation to fade the scrollbars out after a fixed delay. If a subclass provides animated scrolling, the start delay should equal the duration of the scrolling animation.

The animation starts only if at least one of the scrollbars is enabled, as specified by `isHorizontalScrollBarEnabled()` (/reference/android/view/View.html#isHorizontalScrollBarEnabled()) and `isVerticalScrollBarEnabled()` (/reference/android/view/View.html#isVerticalScrollBarEnabled()). When the animation is started, this method returns true, and false otherwise. If the animation is started, this method calls `invalidate()` (/reference/android/view/View.html#invalidate()); in that case the caller should not call `invalidate()` (/reference/android/view/View.html#invalidate()).

This method should be invoked everytime a subclass directly updates the scroll parameters.

Parameters

startDelay	the delay, in milliseconds, after which the animation should start; when the delay is 0, the animation starts immediately

Returns
true if the animation is played, false otherwise

See Also
`scrollBy(int, int)`
`scrollTo(int, int)`

`isHorizontalScrollBarEnabled()`
`isVerticalScrollBarEnabled()`
`setHorizontalScrollBarEnabled(boolean)`
`setVerticalScrollBarEnabled(boolean)`

protected boolean **awakenScrollBars** (int startDelay, boolean invalidate)

Added in API level 5

Trigger the scrollbars to draw. When invoked this method starts an animation to fade the scrollbars out after a fixed delay. If a subclass provides animated scrolling, the start delay should equal the duration of the scrolling animation.

The animation starts only if at least one of the scrollbars is enabled, as specified by `isHorizontalScrollBarEnabled()` (/reference/android/view/View.html#isHorizontalScrollBarEnabled()) and `isVerticalScrollBarEnabled()` (/reference/android/view/View.html#isVerticalScrollBarEnabled()). When the animation is started, this method returns true, and false otherwise. If the animation is started, this method calls `invalidate()` (/reference/android/view/View.html#invalidate()) if the invalidate parameter is set to true; in that case the caller should not call `invalidate()` (/reference/android/view/View.html#invalidate()).

This method should be invoked everytime a subclass directly updates the scroll parameters.

Parameters

startDelay	the delay, in milliseconds, after which the animation should start; when the delay is 0, the animation starts immediately
invalidate	Wheter this method should call invalidate

Returns
true if the animation is played, false otherwise

See Also
`scrollBy(int, int)`
`scrollTo(int, int)`
`isHorizontalScrollBarEnabled()`
`isVerticalScrollBarEnabled()`
`setHorizontalScrollBarEnabled(boolean)`
`setVerticalScrollBarEnabled(boolean)`

protected boolean **awakenScrollBars** ()

Added in API level 5

Trigger the scrollbars to draw. When invoked this method starts an animation to fade the scrollbars out after a default delay. If a subclass provides animated scrolling, the start delay should equal the duration of the scrolling animation.

The animation starts only if at least one of the scrollbars is enabled, as specified by `isHorizontalScrollBarEnabled()` (/reference/android/view/View.html#isHorizontalScrollBarEnabled()) and `isVerticalScrollBarEnabled()` (/reference/android/view/View.html#isVerticalScrollBarEnabled()). When the animation is started, this method returns true, and false otherwise. If the animation is started, this method calls `invalidate()` (/reference/android/view/View.html#invalidate()); in that case the caller should not call `invalidate()` (/reference/android/view/View.html#invalidate()).

This method should be invoked every time a subclass directly updates the scroll parameters.

This method is automatically invoked by `scrollBy(int, int)` (/reference/android/view/View.html#scrollBy(int, int)) and `scrollTo(int, int)` (/reference/android/view/View.html#scrollTo(int, int)).

Returns
true if the animation is played, false otherwise

See Also
`awakenScrollBars(int)`
`scrollBy(int, int)`
`scrollTo(int, int)`
`isHorizontalScrollBarEnabled()`
`isVerticalScrollBarEnabled()`
`setHorizontalScrollBarEnabled(boolean)`
`setVerticalScrollBarEnabled(boolean)`

protected int **computeHorizontalScrollExtent** ()

Added in API level 1

Compute the horizontal extent of the horizontal scrollbar's thumb within the horizontal range. This value is used to compute the length of the thumb within the scrollbar's track.

The range is expressed in arbitrary units that must be the same as the units used by `computeHorizontalScrollRange()` (/reference/android/view/View.html#computeHorizontalScrollRange()) and `computeHorizontalScrollOffset()` (/reference/android/view/View.html#computeHorizontalScrollOffset()).

The default extent is the drawing width of this view.

Returns
the horizontal extent of the scrollbar's thumb

See Also
`computeHorizontalScrollRange()`
`computeHorizontalScrollOffset()`
ScrollBarDrawable

protected int **computeHorizontalScrollOffset** ()

Added in API level 1

Compute the horizontal offset of the horizontal scrollbar's thumb within the horizontal range. This value is used to compute the position of the thumb within the scrollbar's track.

The range is expressed in arbitrary units that must be the same as the units used by `computeHorizontalScrollRange()` (/reference/android/view/View.html#computeHorizontalScrollRange()) and `computeHorizontalScrollExtent()` (/reference/android/view/View.html#computeHorizontalScrollExtent()).

The default offset is the scroll offset of this view.

Returns
the horizontal offset of the scrollbar's thumb

See Also
`computeHorizontalScrollRange()`

computeHorizontalScrollExtent()
ScrollBarDrawable

protected int **computeHorizontalScrollRange** ()

Added in API level 1

Compute the horizontal range that the horizontal scrollbar represents.

The range is expressed in arbitrary units that must be the same as the units used by `computeHorizontalScrollExtent()` (/reference/android/view/View.html#computeHorizontalScrollExtent()) and `computeHorizontalScrollOffset()` (/reference/android/view/View.html#computeHorizontalScrollOffset()).

The default range is the drawing width of this view.

Returns
the total horizontal range represented by the horizontal scrollbar

See Also
`computeHorizontalScrollExtent()`
`computeHorizontalScrollOffset()`
ScrollBarDrawable

protected int **computeVerticalScrollExtent** ()

Added in API level 1

Compute the vertical extent of the horizontal scrollbar's thumb within the vertical range. This value is used to compute the length of the thumb within the scrollbar's track.

The range is expressed in arbitrary units that must be the same as the units used by `computeVerticalScrollRange()` (/reference/android/view/View.html#computeVerticalScrollRange()) and `computeVerticalScrollOffset()` (/reference/android/view/View.html#computeVerticalScrollOffset()).

The default extent is the drawing height of this view.

Returns
the vertical extent of the scrollbar's thumb

See Also
`computeVerticalScrollRange()`
`computeVerticalScrollOffset()`
ScrollBarDrawable

protected int **computeVerticalScrollOffset** ()

Added in API level 1

Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range. This value is used to compute the position of the thumb within the scrollbar's track.

The range is expressed in arbitrary units that must be the same as the units used by `computeVerticalScrollRange()` (/reference/android/view/View.html#computeVerticalScrollRange()) and `computeVerticalScrollExtent()` (/reference/android/view/View.html#computeVerticalScrollExtent()).

The default offset is the scroll offset of this view.

Returns
the vertical offset of the scrollbar's thumb

See Also
`computeVerticalScrollRange()`
`computeVerticalScrollExtent()`
ScrollBarDrawable

protected int **computeVerticalScrollRange** ()

Added in API level 1

Compute the vertical range that the vertical scrollbar represents.

The range is expressed in arbitrary units that must be the same as the units used by `computeVerticalScrollExtent()` (/reference/android/view/View.html#computeVerticalScrollExtent()) and `computeVerticalScrollOffset()` (/reference/android/view/View.html#computeVerticalScrollOffset()).

Returns
the total vertical range represented by the vertical scrollbar
The default range is the drawing height of this view.

See Also
`computeVerticalScrollExtent()`
`computeVerticalScrollOffset()`
ScrollBarDrawable

protected void **dispatchDraw** (Canvas canvas)

Added in API level 1

Called by draw to draw the child views. This may be overridden by derived classes to gain control just before its children are drawn (but after its own view has been drawn).

Parameters

canvas	the canvas on which to draw the view

protected boolean **dispatchGenericFocusedEvent** (MotionEvent event)

Added in API level 14

Dispatch a generic motion event to the currently focused view.

Do not call this method directly. Call `dispatchGenericMotionEvent(MotionEvent)` (/reference/android/view/View.html#dispatchGenericMotionEvent(android.view.MotionEvent)) instead.

Parameters

event	The motion event to be dispatched.

Returns
True if the event was handled by the view, false otherwise.

protected boolean **dispatchGenericPointerEvent** (MotionEvent event) Added in API level 14

Dispatch a generic motion event to the view under the first pointer.

Do not call this method directly. Call `dispatchGenericMotionEvent(MotionEvent)` (/reference/android/view/View.html#dispatchGenericMotionEvent(android.view.MotionEvent)) instead.

Parameters

event	The motion event to be dispatched.

Returns
True if the event was handled by the view, false otherwise.

protected boolean **dispatchHoverEvent** (MotionEvent event) Added in API level 14

Dispatch a hover event.

Do not call this method directly. Call `dispatchGenericMotionEvent(MotionEvent)` (/reference/android/view/View.html#dispatchGenericMotionEvent(android.view.MotionEvent)) instead.

Parameters

event	The motion event to be dispatched.

Returns
True if the event was handled by the view, false otherwise.

protected void **dispatchRestoreInstanceState** (SparseArray<Parcelable> container) Added in API level 1

Called by `restoreHierarchyState(android.util.SparseArray)` (/reference/android/view/View.html#restoreHierarchyState(android.util.SparseArray<android.os.Parcelable>)) to retrieve the state for this view and its children. May be overridden to modify how restoring happens to a view's children; for example, some views may want to not store state for their children.

Parameters

container	The SparseArray which holds previously saved state.

See Also
`dispatchSaveInstanceState(android.util.SparseArray)`
`restoreHierarchyState(android.util.SparseArray)`
`onRestoreInstanceState(android.os.Parcelable)`

protected void **dispatchSaveInstanceState** (SparseArray<Parcelable> container) Added in API level 1

Called by `saveHierarchyState(android.util.SparseArray)` (/reference/android/view/View.html#saveHierarchyState(android.util.SparseArray<android.os.Parcelable>)) to store the state for this view and its children. May be overridden to modify how freezing happens to a view's children; for example, some views may want to not store state for their children.

Parameters

container	The SparseArray in which to save the view's state.

See Also
`dispatchRestoreInstanceState(android.util.SparseArray)`
`saveHierarchyState(android.util.SparseArray)`
`onSaveInstanceState()`

protected void **dispatchSetActivated** (boolean activated) Added in API level 11

Dispatch setActivated to all of this View's children.

Parameters

activated	The new activated state

See Also
`setActivated(boolean)`

protected void **dispatchSetPressed** (boolean pressed) Added in API level 1

Dispatch setPressed to all of this View's children.

Parameters

pressed	The new pressed state

See Also
`setPressed(boolean)`

protected void **dispatchSetSelected** (boolean selected) Added in API level 1

Dispatch setSelected to all of this View's children.

Parameters

selected	The new selected state

See Also
`setSelected(boolean)`

protected void **dispatchVisibilityChanged** (View changedView, int visibility) Added in API level 8

Dispatch a view visibility change down the view hierarchy. ViewGroups should override to route to their children.

Parameters

changedView	The view whose visibility changed. Could be 'this' or an ancestor view.
visibility	The new visibility of changedView: `VISIBLE`, `INVISIBLE` or `GONE`.

protected void **drawableStateChanged** () Added in API level 1

This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown.

Be sure to call through to the superclass when overriding this function.

See Also

setState(int[])

protected boolean **fitSystemWindows** (Rect insets) Added in API level 1

Called by the view hierarchy when the content insets for a window have changed, to allow it to adjust its content to fit within those windows. The content insets tell you the space that the status bar, input method, and other system windows infringe on the application's window.

You do not normally need to deal with this function, since the default window decoration given to applications takes care of applying it to the content of the window. If you use `SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN) or `SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION` (/reference/android/view/View.html#SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION) this will not be the case, and your content can be placed under those system elements. You can then use this method within your view hierarchy if you have parts of your UI which you would like to ensure are not being covered.

The default implementation of this method simply applies the content insets to the view's padding, consuming that content (modifying the insets to be 0), and returning true. This behavior is off by default, but can be enabled through `setFitsSystemWindows(boolean)` (/reference/android/view/View.html#setFitsSystemWindows(boolean)).

This function's traversal down the hierarchy is depth-first. The same content insets object is propagated down the hierarchy, so any changes made to it will be seen by all following views (including potentially ones above in the hierarchy since this is a depth-first traversal). The first view that returns true will abort the entire traversal.

The default implementation works well for a situation where it is used with a container that covers the entire window, allowing it to apply the appropriate insets to its content on all edges. If you need a more complicated layout (such as two different views fitting system windows, one on the top of the window, and one on the bottom), you can override the method and handle the insets however you would like. Note that the insets provided by the framework are always relative to the far edges of the window, not accounting for the location of the called view within that window. (In fact when this method is called you do not yet know where the layout will place the view, as it is done before layout happens.)

Note: unlike many View methods, there is no dispatch phase to this call. If you are overriding it in a ViewGroup and want to allow the call to continue to your children, you must be sure to call the super implementation.

Here is a sample layout that makes use of fitting system windows to have controls for a video view placed inside of the window decorations that it hides and shows. This can be used with code like the second sample (video player) shown in `setSystemUiVisibility(int)` (/reference/android/view/View.html#setSystemUiVisibility(int)).

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    >
    <view class="com.example.android.apis.view.VideoPlayerActivity$Content"
        android:id="@+id/content"
        android:src="@drawable/frantic"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="center"
        />
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        android:animateLayoutChanges="true"
        >
        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="top|center_horizontal"
            android:textColor="#ffffffff"
            android:background="#a0000000"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:gravity="left"
            android:padding="16dp"
            android:text="A title goes here"
            />
        <Button
            android:id="@+id/play"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:textSize="28dp"
            />
        <SeekBar
            android:id="@+id/seekbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom|center_horizontal"
            android:layout_marginBottom="16dp"
            />
    </FrameLayout>
</FrameLayout>
```

Parameters

insets	Current content insets of the window. Prior to `JELLY_BEAN` you must not modify the insets or else you and Android will be unhappy.

Returns

`true` if this view applied the insets and it should not continue propagating further down the hierarchy, `false` otherwise.

See Also

getFitsSystemWindows()
setFitsSystemWindows(boolean)
setSystemUiVisibility(int)

protected float **getBottomFadingEdgeStrength** ()

Returns the strength, or intensity, of the bottom faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

Returns
the intensity of the bottom fade as a float between 0.0f and 1.0f

protected int **getBottomPaddingOffset** () Added in API level 2

Amount by which to extend the bottom fading region. Called only when `isPaddingOffsetRequired()` (/reference/android/view/View.html#isPaddingOffsetRequired()) returns true.

Returns
The bottom padding offset in pixels.

See Also
`isPaddingOffsetRequired()`

protected ContextMenu.ContextMenuInfo **getContextMenuInfo** () Added in API level 1

Views should implement this if they have extra information to associate with the context menu. The return result is supplied as a parameter to the `onCreateContextMenu(ContextMenu, View, ContextMenuInfo)` (/reference/android/view/View.OnCreateContextMenuListener.html#onCreateContextMenu(android.view.ContextMenu, android.view.View, android.view.ContextMenu.ContextMenuInfo)) callback.

Returns
Extra information about the item for which the context menu should be shown. This information will vary across different subclasses of View.

protected int **getHorizontalScrollbarHeight** () Added in API level 1

Returns the height of the horizontal scrollbar.

Returns
The height in pixels of the horizontal scrollbar or 0 if there is no horizontal scrollbar.

protected float **getLeftFadingEdgeStrength** () Added in API level 1

Returns the strength, or intensity, of the left faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

Returns
the intensity of the left fade as a float between 0.0f and 1.0f

protected int **getLeftPaddingOffset** () Added in API level 2

Amount by which to extend the left fading region. Called only when `isPaddingOffsetRequired()` (/reference/android/view/View.html#isPaddingOffsetRequired()) returns true.

Returns
The left padding offset in pixels.

See Also
`isPaddingOffsetRequired()`

protected float **getRightFadingEdgeStrength** () Added in API level 1

Returns the strength, or intensity, of the right faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

Returns
the intensity of the right fade as a float between 0.0f and 1.0f

protected int **getRightPaddingOffset** () Added in API level 2

Amount by which to extend the right fading region. Called only when `isPaddingOffsetRequired()` (/reference/android/view/View.html#isPaddingOffsetRequired()) returns true.

Returns
The right padding offset in pixels.

See Also
`isPaddingOffsetRequired()`

protected int **getSuggestedMinimumHeight** () Added in API level 1

Returns the suggested minimum height that the view should use. This returns the maximum of the view's minimum height and the background's minimum height (`getMinimumHeight()` (/reference/android/graphics/drawable/Drawable.html#getMinimumHeight())).

When being used in `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)), the caller should still ensure the returned height is within the requirements of the parent.

Returns
The suggested minimum height of the view.

protected int **getSuggestedMinimumWidth** () Added in API level 1

Returns the suggested minimum width that the view should use. This returns the maximum of the view's minimum width) and the background's minimum width (`getMinimumWidth()` (/reference/android/graphics/drawable/Drawable.html#getMinimumWidth())).

When being used in `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)), the caller should still ensure the returned width is within the requirements of the parent.

Returns
The suggested minimum width of the view.

protected float **getTopFadingEdgeStrength** () Added in API level 1

Returns the strength, or intensity, of the top faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

Returns
the intensity of the top fade as a float between 0.0f and 1.0f

protected int **getTopPaddingOffset** () Added in API level 2

Amount by which to extend the top fading region. Called only when `isPaddingOffsetRequired()` (/reference/android/view/View.html#isPaddingOffsetRequired()) returns true.

Returns
The top padding offset in pixels.

See Also
`isPaddingOffsetRequired()`

protected int **getWindowAttachCount** () Added in API level 1

Returns
The number of times this view has been attached to a window

protected void **initializeFadingEdge** (TypedArray a) Added in API level 1

Initializes the fading edges from a given set of styled attributes. This method should be called by subclasses that need fading edges and when an instance of these subclasses is created programmatically rather than being inflated from XML. This method is automatically called when the XML is inflated.

Parameters

a	the styled attributes set to initialize the fading edges from

protected void **initializeScrollbars** (TypedArray a) Added in API level 1

Initializes the scrollbars from a given set of styled attributes. This method should be called by subclasses that need scrollbars and when an instance of these subclasses is created programmatically rather than being inflated from XML. This method is automatically called when the XML is inflated.

Parameters

a	the styled attributes set to initialize the scrollbars from

protected boolean **isPaddingOffsetRequired** () Added in API level 2

If the View draws content inside its padding and enables fading edges, it needs to support padding offsets. Padding offsets are added to the fading edges to extend the length of the fade so that it covers pixels drawn inside the padding. Subclasses of this class should override this method if they need to draw content inside the padding.

Returns
True if padding offset must be applied, false otherwise.

See Also
`getLeftPaddingOffset()`
`getRightPaddingOffset()`
`getTopPaddingOffset()`
`getBottomPaddingOffset()`

protected static int[] **mergeDrawableStates** (int[] baseState, int[] additionalState) Added in API level 1

Merge your own state values in *additionalState* into the base state values *baseState* that were returned by `onCreateDrawableState(int)` (/reference/android/view/View.html#onCreateDrawableState(int)).

Parameters

baseState	The base state values returned by `onCreateDrawableState(int)`, which will be modified to also hold your own additional state values.
additionalState	The additional state values you would like added to *baseState*; this array is not modified.

Returns
As a convenience, the *baseState* array you originally passed into the function is returned.

See Also
`onCreateDrawableState(int)`

protected void **onAnimationEnd** () Added in API level 1

Invoked by a parent ViewGroup to notify the end of the animation currently associated with this view. If you override this method, always call super.onAnimationEnd();

See Also
`setAnimation(android.view.animation.Animation)`
`getAnimation()`

protected void **onAnimationStart** () Added in API level 1

Invoked by a parent ViewGroup to notify the start of the animation currently associated with this view. If you override this method, always call super.onAnimationStart();

See Also
`setAnimation(android.view.animation.Animation)`
`getAnimation()`

protected void **onAttachedToWindow** () Added in API level 1

This is called when the view is attached to a window. At this point it has a Surface and will start drawing. Note that this function is guaranteed to be called before `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)), however it may be called any time before the first onDraw -- including before or after `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)).

See Also
`onDetachedFromWindow()`

protected void **onConfigurationChanged** (Configuration newConfig) Added in API level 8

Called when the current configuration of the resources being used by the application have changed. You can use this to decide when to reload resources that can changed based on orientation and other configuration characterstics. You only need to use this if you are not relying on the normal `Activity` (/reference/android/app/Activity.html) mechanism of recreating the activity instance upon a configuration change.

Parameters

newConfig	The new resource configuration.

protected void **onCreateContextMenu** (ContextMenu menu) Added in API level 1

Views should implement this if the view itself is going to add items to the context menu.

Parameters

menu	the context menu to populate

protected int[] **onCreateDrawableState** (int extraSpace) Added in API level 1

Generate the new `Drawable` (/reference/android/graphics/drawable/Drawable.html) state for this view. This is called by the view system when the cached Drawable state is determined to be invalid. To retrieve the current state, you should use `getDrawableState()` (/reference/android/view/View.html#getDrawableState()).

Parameters

extraSpace	if non-zero, this is the number of extra entries you would like in the returned array in which you can place your own states.

Returns
Returns an array holding the current `Drawable` state of the view.

See Also
`mergeDrawableStates(int[], int[])`

protected void **onDetachedFromWindow** () Added in API level 1

This is called when the view is detached from a window. At this point it no longer has a surface for drawing.

See Also
`onAttachedToWindow()`

protected void **onDisplayHint** (int hint) Added in API level 8

Gives this view a hint about whether is displayed or not. For instance, when a View moves out of the screen, it might receives a display hint indicating the view is not displayed. Applications should not *rely* on this hint as there is no guarantee that they will receive one.

Parameters

hint	A hint about whether or not this view is displayed: `VISIBLE` or `INVISIBLE`.

protected void **onDraw** (Canvas canvas) Added in API level 1

Implement this to do your drawing.

Parameters

canvas	the canvas on which the background will be drawn

protected final void **onDrawScrollBars** (Canvas canvas) Added in API level 7

Request the drawing of the horizontal and the vertical scrollbar. The scrollbars are painted only if they have been awakened first.

Parameters

canvas	the canvas on which to draw the scrollbars

See Also
`awakenScrollBars(int)`

protected void **onFinishInflate** () Added in API level 1

Finalize inflating a view from XML. This is called as the last phase of inflation, after all child views have been added.

Even if the subclass overrides onFinishInflate, they should always be sure to call the super method, so that we get called.

protected void **onFocusChanged** (boolean gainFocus, int direction, Rect previouslyFocusedRect) Added in API level 1

Called by the view system when the focus state of this view changes. When the focus change event is caused by directional navigation, direction and previouslyFocusedRect provide insight into where the focus is coming from. When overriding, be sure to call up through to the super class so that the standard focus handling will occur.

Parameters

gainFocus	True if the View has focus; false otherwise.
direction	The direction focus has moved when requestFocus() is called to give this view focus. Values are `FOCUS_UP`, `FOCUS_DOWN`, `FOCUS_LEFT`, `FOCUS_RIGHT`, `FOCUS_FORWARD`, or `FOCUS_BACKWARD`. It may not always apply, in which case use the default.
previouslyFocusedRect	The rectangle, in this view's coordinate system, of the previously focused view. If applicable, this will be passed in as finer grained information about where the focus is coming from (in addition to direction). Will be `null` otherwise.

protected void **onLayout** (boolean changed, int left, int top, int right, int bottom) Added in API level 1

Called from layout when this view should assign a size and position to each of its children. Derived classes with children should override this method and call layout on each of their children.

Parameters

changed	This is a new size or position for this view
left	Left position, relative to parent
top	Top position, relative to parent
right	Right position, relative to parent
bottom	Bottom position, relative to parent

protected void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec) Added in API level 1

Measure the view and its content to determine the measured width and the measured height. This method is invoked by `measure(int, int)` (/reference/android/view/View.html#measure(int, int)) and should be overriden by subclasses to provide accurate and efficient measurement of their contents.

CONTRACT: When overriding this method, you *must* call `setMeasuredDimension(int, int)` (/reference/android/view/View.html#setMeasuredDimension(int, int)) to store the measured width and height of this view. Failure to do so will trigger an `IllegalStateException`, thrown by `measure(int, int)` (/reference/android/view/View.html#measure(int, int)). Calling the superclass' `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)) is a valid use.

The base class implementation of measure defaults to the background size, unless a larger size is allowed by the MeasureSpec. Subclasses should override `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)) to provide better measurements of their content.

If this method is overridden, it is the subclass's responsibility to make sure the measured height and width are at least the view's minimum height and width (`getSuggestedMinimumHeight()` (/reference/android/view/View.html#getSuggestedMinimumHeight()) and `getSuggestedMinimumWidth()` (/reference/android/view/View.html#getSuggestedMinimumWidth())).

Parameters

widthMeasureSpec	horizontal space requirements as imposed by the parent. The requirements are encoded with `View.MeasureSpec`.
heightMeasureSpec	vertical space requirements as imposed by the parent. The requirements are encoded with `View.MeasureSpec`.

See Also
`getMeasuredWidth()`
`getMeasuredHeight()`
`setMeasuredDimension(int, int)`
`getSuggestedMinimumHeight()`
`getSuggestedMinimumWidth()`
`getMode(int)`
`getSize(int)`

protected void **onOverScrolled** (int scrollX, int scrollY, boolean clampedX, boolean clampedY) Added in API level 9

Called by `overScrollBy(int, int, int, int, int, int, int, int, boolean)` (/reference/android/view/View.html#overScrollBy(int, int, int, int, int, int, int, int, boolean)) to respond to the results of an over-scroll operation.

Parameters

scrollX	New X scroll value in pixels
scrollY	New Y scroll value in pixels
clampedX	True if scrollX was clamped to an over-scroll boundary
clampedY	True if scrollY was clamped to an over-scroll boundary

protected void **onRestoreInstanceState** (Parcelable state) Added in API level 1

Hook allowing a view to re-apply a representation of its internal state that had previously been generated by `onSaveInstanceState()` (/reference/android/view/View.html#onSaveInstanceState()). This function will never be called with a null state.

Parameters

state	The frozen state that had previously been returned by `onSaveInstanceState()`.

See Also
`onSaveInstanceState()`
`restoreHierarchyState(android.util.SparseArray)`
`dispatchRestoreInstanceState(android.util.SparseArray)`

protected Parcelable **onSaveInstanceState** () Added in API level 1

Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state. This state should only contain information that is not persistent or can not be reconstructed later. For example, you will never store your current position on screen because that will be computed again when a new instance of the view is placed in its view hierarchy.

Some examples of things you may store here: the current cursor position in a text view (but usually not the text itself since that is stored in a content provider or other persistent storage), the currently selected item in a list view.

Returns
Returns a Parcelable object containing the view's current dynamic state, or null if there is nothing interesting to save. The default implementation returns null.

See Also
`onRestoreInstanceState(android.os.Parcelable)`
`saveHierarchyState(android.util.SparseArray)`
`dispatchSaveInstanceState(android.util.SparseArray)`
`setSaveEnabled(boolean)`

protected void **onScrollChanged** (int l, int t, int oldl, int oldt) Added in API level 1

This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). This is typically as a result of `scrollBy(int, int)` (/reference/android/view/View.html#scrollBy(int, int)) or `scrollTo(int, int)` (/reference/android/view/View.html#scrollTo(int, int)) having been called.

Parameters

l	Current horizontal scroll origin.
t	Current vertical scroll origin.
oldl	Previous horizontal scroll origin.
oldt	Previous vertical scroll origin.

protected boolean **onSetAlpha** (int alpha)

Added in API level 1

Invoked if there is a Transform that involves alpha. Subclass that can draw themselves with the specified alpha should return true, and then respect that alpha when their onDraw() is called. If this returns false then the view may be redirected to draw into an offscreen buffer to fulfill the request, which will look fine, but may be slower than if the subclass handles it internally. The default implementation returns false.

Parameters

alpha	The alpha (0..255) to apply to the view's drawing

Returns
true if the view can draw with the specified alpha.

protected void **onSizeChanged** (int w, int h, int oldw, int oldh)

Added in API level 1

This is called during layout when the size of this view has changed. If you were just added to the view hierarchy, you're called with the old values of 0.

Parameters

w	Current width of this view.
h	Current height of this view.
oldw	Old width of this view.
oldh	Old height of this view.

protected void **onVisibilityChanged** (View changedView, int visibility)

Added in API level 8

Called when the visibility of the view or an ancestor of the view is changed.

Parameters

changedView	The view whose visibility changed. Could be 'this' or an ancestor view.
visibility	The new visibility of changedView: `VISIBLE`, `INVISIBLE` or `GONE`.

protected void **onWindowVisibilityChanged** (int visibility)

Added in API level 1

Called when the window containing has change its visibility (between `GONE` (/reference/android/view/View.html#GONE), `INVISIBLE` (/reference/android/view/View.html#INVISIBLE), and `VISIBLE` (/reference/android/view/View.html#VISIBLE)). Note that this tells you whether or not your window is being made visible to the window manager; this does *not* tell you whether or not your window is obscured by other windows on the screen, even if it is itself visible.

Parameters

visibility	The new visibility of the window.

protected boolean **overScrollBy** (int deltaX, int deltaY, int scrollX, int scrollY, int scrollRangeX, int scrollRangeY, int maxOverScrollX, int maxOverScrollY, boolean isTouchEvent)

Added in API level 9

Scroll the view with standard behavior for scrolling beyond the normal content boundaries. Views that call this method should override `onOverScrolled(int, int, boolean, boolean)` (/reference/android/view/View.html#onOverScrolled(int, int, boolean, boolean)) to respond to the results of an over-scroll operation. Views can use this method to handle any touch or fling-based scrolling.

Parameters

deltaX	Change in X in pixels
deltaY	Change in Y in pixels
scrollX	Current X scroll value in pixels before applying deltaX
scrollY	Current Y scroll value in pixels before applying deltaY
scrollRangeX	Maximum content scroll range along the X axis
scrollRangeY	Maximum content scroll range along the Y axis
maxOverScrollX	Number of pixels to overscroll by in either direction along the X axis.
maxOverScrollY	Number of pixels to overscroll by in either direction along the Y axis.
isTouchEvent	true if this scroll operation is the result of a touch event.

Returns
true if scrolling was clamped to an over-scroll boundary along either axis, false otherwise.

protected final void **setMeasuredDimension** (int measuredWidth, int measuredHeight)

Added in API level 1

This method must be called by `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)) to store the measured width and measured height. Failing to do so will trigger an exception at measurement time.

Parameters

measuredWidth	The measured width of this view. May be a complex bit mask as defined by `MEASURED_SIZE_MASK` and `MEASURED_STATE_TOO_SMALL`.
measuredHeight	The measured height of this view. May be a complex bit mask as defined by `MEASURED_SIZE_MASK` and `MEASURED_STATE_TOO_SMALL`.

protected boolean **verifyDrawable** (Drawable who)

Added in API level 1

If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying. This allows animations for those drawables to be scheduled.

Be sure to call through to the super class when overriding this function.

Parameters

who	The Drawable to verify. Return true if it is one you are displaying, else return the result of calling through to the super class.

Returns

boolean If true than the Drawable is being displayed in the view; else false and it is not allowed to animate.

See Also

`unscheduleDrawable(android.graphics.drawable.Drawable)`
`drawableStateChanged()`