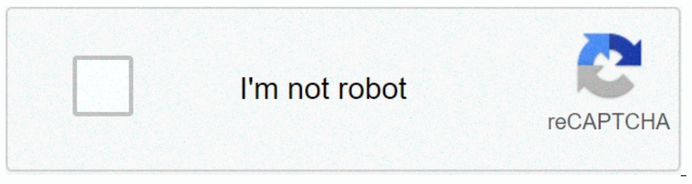

Continue

Set background color java android

How to change the background color of the Android View sample. MainActivity.java mainActivity of public class, mainactivity extends the activity { TextView hTextView; TableRow hTableRow; HButton button, hButtonStop; private handler mHandler = new Handler(); private int nCounter = 0; @Override open public noCreate (Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.main); hTextView = (TextView)findViewById(R.id.idTextView); hButton = (Button)findViewById(R.id.idButton); hButton.setOnClickListener(mButtonStartListener); hTableRow = (TableRow)findViewById(R.id.idTable1); } } end onCreate View.OnClickListener mButtonStartListener = new OnClickListener() { public void onClick(View v) { hTableRow.setBackgroundColor(Color.YELLOW); } } } main.xml <?xml version=1.0 encoding=utf-8?>android:accessibilityAnd this view is or is not a title for accessibility purposes. android:accessibilityLiveRegion Tells accessibility services whether the user should be notified when this view changes. android:AccessibilityPaneTitle The title of this view should be presented to accessibility as a panel title. android:accessibilityTraversalAfter Sets the id of a view after which it is visited in accessibility traversal. android:AccessibilityTraversalBefore Sets the id of a view before which it is visited in accessibility traversal. android:alpha view property, as a value between 0 (completely transparent) and 1 (completely opaque). android:autofillHints Describes the content so that an autocomplete service can fill in the appropriate data. android:autofilledHighlight Drawable to be drawn on the display to mark it as automatically populated It can be a reference to another feature <LinearLayout xmlns:android=//schemas.android.com/apk/res/android android:orientation=vertical android:layout_width=fill_parent android:layout_height=fill_parent> <TableLayout android:layout_height=wrap_content android:layout_width=fill_parent android:id=@+id/Table1> <TableRow android:id=@+id/idTableRow1 android:layout_width=wrap_content android:layout_height=wrap_content android:background=#5655AA> <TextView android:id=@+id/idTextView android:layout_width=fill_parent android:layout_height=wrap_content android:text=@string/></TextView> </TableRow> </TableLayout> </LinearLayout> , in form @[+][package:]type/name or a thematic attribute on the form? [package:]type/name. android:A drawable background to use as background. android:Tint backgroundt to apply to the background. android:backgroundIntMode Blending mode used to apply the background tint. android:clickable Sets whether this view reacts to click events. android:contentDescription Defines text that briefly describes the contents of the view. android:contextClickable Defines whether this view reacts to context clicks. android:defaultFocusHighlightEnabled If this View should use a default focus highlight when it gets focused but has no set in your background. android:drawingCacheQuality Defines the quality of translucent drawing caches. android:duplicateParentState When this attribute is set to true, the preview gets its draw state (focused, pressed, etc.) from its direct parent and not from itself. Android:Altitude base z depth of view. android:fadeScrollbars sets if you fade scroll bars when they are not in use. android:fadingEdgeLength defines the length of the faded edges. android:filterTouches When obscured specifies whether to filter touches when the view window is obscured by another visible window. android:fitsSystemWindows Boolean internal attribute to adjust the display layout based on system windows, such as the status bar. android:focusable Controls if a view can have focus. android:focusableInTouchMode Boolean that controls whether a display can have focus during touch mode. android:focusedByDefault If this view is a standard focus view. android:forceHasOverlappingRendering if this view has elements that can overlap when drawn. android:foreground sets the drawable to draw on the content. android:ForegroundGravity sets gravity to apply to drawable foreground. android:Tint foregroundTint to apply to the foreground. android:foregroundIntMode Blending mode used to apply foreground tint. android:hapticFeedbackEnabled Boolean that controls whether a display should have haptic feedback enabled for events such as long presses. android:id Provide an identifier name for this view, to retrieve it later with View.findViewById() or Activity.findViewById(). android:importantForAccessibility Describes whether or not this view is important for accessibility. android:importantForAutofill suggests to the Android System whether the display node associated with this View should be included in a display structure used for autofill purposes. android:importantForContentCapture suggests to the Android System whether the display node associated with this View should be used for content capture purposes. android:isScrollContainer set this if the preview serves as a scroll container, which means that it can be resized to shrink its general window so that there is room for an input method. android:keepScreenOn Controls if the display window should keep the screen on while it is visible. android:keyboardNavigationCluster If this view is a root of a keyboard navigation cluster. android:layerType specifies the type of layer that supports this view. android:Direction layout sets the direction of the layout drawing. android:longClickable Sets whether this view reacts to long-click events. android:minHeight Sets the minimum view height. android:minWidth sets the minimum view width. android:nextClusterForward defines the next keyboard navigation cluster. Sets the next view to focus when the next focus is View.FOCUS_DOWN If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a one will result when the reference is accessed. android:nextFocusForward Sets the next view to focus on when the next focus is View.FOCUS_FORWARD If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. android:nextFocusLeft Sets the next view to focus on when the next focus is View.FOCUS_LEFT. android:nextFocusRight Sets the next view to focus on when the next focus is View.FOCUS_RIGHT If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. android:nextFocusUp Sets the next view to focus on when the next focus is View.FOCUS_UP If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. android:onClick Method name in the context of this View to invoke when the view is clicked. android:outlineAmbientShadowColor sets the color of the ambient shadow that is drawn when the view has a Positive Z or Elevation value. android:outlineSpotShadowColor Sets the spot shadow color that is drawn when the preview has a Z value or positive elevation. android:Padding Sets the fill, in pixels, of the four borders. android:FillBottom sets the fill, in pixels, of the bottom edge; see R.attr.fill. android:fillEnd Sets the fill, in pixels, of the final edge; see R.attr.fill. android:fillHorizontal Sets the fill, in pixels, from the left and right edges; see R.attr.fill. android:PaddingLeft sets the fill, in pixels, of the left edge; see R.attr.fill. android:UpholsteryRight sets the fill, in pixels, of the right edge; see R.attr.fill. android:fillStart sets the fill, in pixels, of the initial border; see R.attr.fill. android:padding Sets the fill, in pixels, of the top edge; see R.attr.fill. android:Vertical padding sets the padding, in pixels, of the top and bottom edges; see R.attr.fill. android:requerFadingEdge Defines which edges should be faded in scrolling. android:rotation of view, in degrees. android:rotation rotaX of the view around the x-axis, in degrees. android: rotation of the view around the y-axis, in degrees. android:saveEnabled If false, no state will be saved for this view when being frozen. android:scaleX scale of view in the x. android direction:scaley scale of view in the direction y. android:screenReaderFocusable If this view should be treated as a focusable unit by screen reader accessibility tools. android:scrollIndicators Defines which scroll indicators should be displayed when the display can be scrolled. android:scrollX The horizontal offset in pixels. android:scrollY The initial vertical scroll scroll offset, in pixels. android:scrollbarAlwaysDrawHorizontalTrack sets if horizontal horizontal should always be drawn. android:scrollbarAlwaysDrawVerticalTrack defines whether the vertical scroll strip should always be drawn. android:scrollbarDefaultDelayBeforeFade sets the delay in milliseconds that a scroll bar waits before disappearing. android:scrollbarFadeDuration Sets the delay in milliseconds that a scroll bar takes to disappear. android:scrollbarSize Sets the width of vertical scroll bars and the height of horizontal scroll bars. android:scrollbarStyle controls the style and position of the scroll bar. android:scrollbarThumbHorizontal sets the horizontal thumb of the drawn scroll bar. android:scrollbarThumbVertical sets the thumb of the drawn vertical scroll bar. android:scrollbarTrackHorizontal Sets the drawn horizontal scroll strip. android:scrollbarTrackVertical sets the drawn vertical scroll strip. android:scrollbars defines which scroll bars should be displayed in the scroll or not. android:soundEffectsEnabled Boolean that controls whether a display should have sound effects enabled for events like click and tap. android:stateListAnimator Sets the state-based animator to the View. android:tag Provide a tag for this view containing a String, to be retrieved later with View.getTag() or searched with View.findViewWithTag(). android:textAlignment Sets the alignment of the text. android:textDirection Sets the direction of the text. android:theme specifies a thematic replacement for a display. android:tooltipText Sets text displayed in a small pop-up window on hover or long press. android:transformPivotX x location of the pivot point around which the view will rotate and scale. android:transformPivotY and location of the pivot point around which the view will rotate and scale. android:transitionName Names a View in such a way that it can be identified for transitions. android:translationX in view x. android:translationY translation in y of the view. android:translationZ z translation of view. android:Visibility Controls the initial visibility of the view. int ACCESSIBILITY_LIVE_REGION_ASSERTIVE live region mode specifying that accessibility services must stop the 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 in this view. int ACCESSIBILITY_LIVE_REGION_POLITE live region mode specifying that accessibility services should announce changes in this view. int AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS Flag by asking you to add views marked as not important for autocomplete (see setImportantForAutofill(int)) to a Display Setting. String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE Tip indicating that this view can be automatically populated with a credit card expiration date. String Tip indicating that this view can be automatically populated with a credit card expiration day. String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH tip indicating This view can be automatically filled with one month of credit card expiration. String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_YEAR Tip indicating that this view can be automatically populated with one year of credit card expiration. String AUTOFILL_HINT_CREDIT_CARD_NUMBER Tip indicating that this view can be automatically filled with a credit card number. String AUTOFILL_HINT_CREDIT_CARD_SECURITY_CODE Tip indicating that this view can be automatically populated with a credit card security code. String AUTOFILL_HINT_EMAIL_ADDRESS Tip indicating that this view can be automatically populated with an email address. String AUTOFILL_HINT_NAME Tip indicating that this view can be automatically populated with a user's real name. String AUTOFILL_HINT_PASSWORD Tip indicating that this view can be automatically populated with a password. String AUTOFILL_HINT_PHONE Tip indicating that this view can be automatically populated with a phone number. String AUTOFILL_HINT_POSTAL_ADDRESS Tip indicating that this view can be automatically filled with a postal address. String AUTOFILL_HINT_POSTAL_CODE Tip indicating that this visualization can be automatically populated with a zip code. String AUTOFILL_HINT_USERNAME Tip indicating that this view can be automatically populated with a user name. int AUTOFILL_TYPE_DATE autofill type for a field that contains a date, which is represented by AUTOFILL_TYPE_LIST a long one representing the number of milliseconds since the default base time known as time, i.e. January 1, 1970, 00:00 GMT (see Date.getTime(). which is filled by an int representing the element index within the list (from 0). int AUTOFILL_TYPE_NONE autofill type for visualizations that cannot be filled automatically AUTOFILL_TYPE_TEXT. which is filled by a CharSequence. int AUTOFILL_TYPE_TOGGLE autofill type for a togglable field, which is filled by a boolean. int DRAG_FLAG_GLOBAL Flag indicating that a drag can cross the DRAG_FLAG_GLOBAL_URI_WRITE DRAG_FLAG_GLOBAL_URI_READ DRAG_FLAG_GLOBAL_PERSISTABLE_URI_PERMISSION boundaries of the window. , the URI permission grant can be persisted through device restarts until it is explicitly revoked with contextePermission(Uri, int) Context.revokeUriPermission}. int DRAG_FLAG_GLOBAL_PREFIX_URI_PERMISSION When this flag is used with DRAG_FLAG_GLOBAL_URI_READ and/or DRAG_FLAG_GLOBAL_URI_WRITE, the URI permission grant applies to any URI that is a prefix compatible with the original GRANTED URI. int DRAG_FLAG_GLOBAL_URI_READ When this flag is used DRAG_FLAG_GLOBAL, the drag recipient may request read access to the URI(s) contained in the ClipData object. int DRAG_FLAG_GLOBAL_URI_WRITE When this flag is used with DRAG_FLAG_GLOBAL, the drag recipient can request access to the recording of the URI(s) content contained in the ClipData object. Int DRAG_FLAG_OPAQUE Flag indicating that the drag shadow will be opaque. int DRAWING_CACHE_QUALITY_AUTO This constant was deprethered in API level 28. The view drawing cache has largely become obsolete obsolete the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. int DRAWING_CACHE_QUALITY_HIGH this constant was deprethered in API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. int DRAWING_CACHE_QUALITY_LOW This constant was deprethered at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the display hierarchy or individual visualizations, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Int Int Find views that contain the description of the specified content. int FIND_VIEWS_WITH_TEXT Find views that make the specified text. int FOCUSABLE This view wants keys. int FOCUSABLES_ALL Display flag indicating whether addFocusables (java.util.ArrayList, int, int) should add all focusable views, regardless of whether they are focalable in touch mode. int FOCUSABLES_TOUCH_MODE Display flag indicating whether addFocusables (java.util.ArrayList, int, int) should only add focusable views in touch mode. int FOCUSABLE_AUTO This view automatically determines the focus capacity. use FOCUS_BACKWARD int with focoSearch(int). use FOCUS_DOWN int with focoSearch(int). use FOCUS_FORWARD int with focoSearch(int). use FOCUS_LEFT int with focoSearch(int). use FOCUS_RIGHT int with focoSearch(int). use FOCUS_UP int with focoSearch(int). int GONE This view is invisible, and takes no room for layout purposes. int HAPTIC_FEEDBACK_ENABLED display flag indicating whether this view should have plythic feedback enabled for events such as long presses. int IMPORTANT_FOR_ACCESSIBILITY_AUTO determine automatically whether a view is important for accessibility. int IMPORTANT_FOR_ACCESSIBILITY_NO Vision is not important for accessibility. int IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS Vision is not important for accessibility, nor any of its descending visions. int IMPORTANT_FOR_ACCESSIBILITY_YES Vision is important for accessibility. int IMPORTANT_FOR_AUTOFILL_AUTO determine automatically whether a view is important for autocomplete. int IMPORTANT_FOR_AUTOFILL_NO The view is not important for autocomplete, but your children (if any) will be traversed. int IMPORTANT_FOR_AUTOFILL_NO_EXCLUDE_DESCENDANTS The view is not important for autocomplete, and your children (if any) will not be traversed. int IMPORTANT_FOR_AUTOFILL_YES The view is important for autocomplete, and your children (if any) will be traversed. int IMPORTANT_FOR_AUTOFILL_YES_EXCLUDE_DESCENDANTS The view is important for autocomplete, but your children (if any) will not be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_AUTO automatically determine whether a view is important for capturing content. int IMPORTANT_FOR_CONTENT_CAPTURE_NO The view is not important for capturing content, but your children (if any) will be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_NO_EXCLUDE_DESCENDANTS View is not important for capturing content, and your children (if any) will not be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_YES The view is important for capturing content, and your children (if any) will be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_YES_EXCLUDE_DESCENDANTS The view is important for capturing content, but your children (if any) will not be traversed. Invisible int This view is invisible, but still space for layout purposes. int KEEP_SCREEN_ON Display flag indicating that the screen should remain while the window containing this view is visible to the user. int LAYER_TYPE_HARDWARE indicates that the preview has a hardware layer. Hardware. LAYER_TYPE_NONE indicates that the visualization 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 The horizontal direction of the layout of this visualization is deduced from the default language script for the location. int LAYOUT_DIRECTION_LTR horizontal layout direction of this view is left to right. int LAYOUT_DIRECTION_RTL horizontal layout direction of this view is right to left. int MEASURED_HEIGHT_STATE_SHIFT Bit shift of MEASURED_STATE_MASK to reach the height bits for functions that combine width and height in 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 measured actual size. int MEASURED_STATE_MASK bits of getMeasuredWidthAndState() and getMeasuredWidthAndState() that provide the additional bits of the state. int MEASURED_STATE_TOO_SMALL Bit of getMeasuredWidthAndState() and getMeasuredWidthAndState() that indicates that the measured size is smaller than the space that the view would like to have. int NOT_FOCUSABLE This view does not want keys. 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, as long as it is a view that can scroll. int OVER_SCROLL_IF_CONTENT_SCROLLS Allow a user to scroll too much of this view only if the content is large enough to scroll significantly, as long as 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 turned on. int SCROLLBARS_INSIDE_INSET The scrollbar style to display scroll bars within the padded area, increasing the view's padding. int SCROLLBARS_INSIDE_OVERLAY the scrollbar style to display scroll bars within the content area without increasing the fill. int SCROLLBARS_OUTSIDE_INSET The scrollbar style to display scroll bars at the edge of the view, increasing the view's padding. int SCROLLBARS_OUTSIDE_OVERLAY The scrollbar style to display scroll bars at the edge of the view without increasing the padding. int SCROLLBAR_POSITION_DEFAULT Position the scroll bar in 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 SCROLL_AXIS_HORIZONTAL Indica scroll along the horizontal axis. int SCROLL_AXIS_NONE indicates no viewing scroll axis. int SCROLL_AXIS_VERTICAL indicates rolling along the vertical axis. int SCROLL_INDICATOR_BOTTOM direction of scroll indicator to bottom edge View. int SCROLL_INDICATOR_END direction scrolling indicator to the final end edge the view. int SCROLL_INDICATOR_LEFT direction of the scroll indicator to the left edge of the view. int SCROLL_INDICATOR_RIGHT direction of the Scroll indicator to the right edge of the view. int SCROLL_INDICATOR_START direction of the Scroll indicator to the initial edge of the view. int SCROLL_INDICATOR_TOP direction of the scroll indicator to the top edge of the view. int SOUND_EFFECTS_ENABLED display flag indicating whether this view should have sound effects enabled for events such as click and tap. int STATUS_BAR_HIDDEN This constant was deprecated at API level 15. Use SYSTEM_UI_FLAG_LOW_PROFILE instead. int STATUS_BAR_VISIBLE this constant was deprecated at API level 15. Use SYSTEM_UI_FLAG_VISIBLE instead. int SYSTEM_UI_FLAG_FULLSCREEN this constant was deprecated at API level 30. Use windowinsetsController#hide(int) with Type#statusBars() instead. int SYSTEM_UI_FLAG_HIDE_NAVIGATION This constant was deprecated at API level 30. Use WindowInsetsController#hide(int) with Type#navigationBars() instead. int SYSTEM_UI_FLAG_IMMERSIVE this constant was deprethesised at API level 30. Use windowinsetscontroller#BEHAVIOR_SHOW_BARS_BY_SWIPE instead. int SYSTEM_UI_FLAG_IMMERSIVE_STICKY This constant was deprethesised at API level 30. Use windowinsetscontroller#BEHAVIOR_SHOW_TRANSIENT_BARS_BY_SWIPE instead. int SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN This constant was deprecated at API level 30. For floating windows, use LayoutParams#setFitInsetsTypes(int) with Type#statusBars() ()). For non-floating windows that fill the screen, call window#setDecorFitsSystemWindows (boolean) with fake. int SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION This constant was deprecated at API level 30. Use LayoutParams#setFitInsetsTypes(int) with Type#navigationBars(). For non-floating windows that fill the screen, window#setDecorFitsSystemWindows (boolean) with fake. int SYSTEM_UI_FLAG_LAYOUT_STABLE This constant was deprethesised at API level 30. Use WindowInsets#getInsetsIgnoringVisibility(int) instead to retrieve insets that do not change when system bars change visibility state. int SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR this constant was deprethesised at API level 30. Use windowinsetscontroller#APPEARANCE_LIGHT_NAVIGATION_BARS instead. int SYSTEM_UI_FLAG_LIGHT_STATUS_BAR This constant was deprethesised at API level 30. Use windowinsetscontroller#APPEARANCE_LIGHT_STATUS_BARS instead. int SYSTEM_UI_FLAG_LOW_PROFILE This constant was deprethesised at API level 30. Low profile mode is deprethesised. Hide system bars instead if the application needs to be in discrete mode. Use WindowInsetsController#hide(int) with Type#systemBars(). int SYSTEM_UI_FLAG_VISIBLE This constant was deprecated at API level 30. SystemUiVisibility flags are deprecated. Use windowinsetscontroller instead. int SYSTEM_UI_LAYOUT_FLAGS This constant was deprecated at level 30 Api. System interface layout flags are deprethesised. int TEXT_ALIGNMENT_CENTER Central paragraph, for example, ALIGN_CENTER. int TEXT_ALIGNMENT_GRAVITY Standard Standard the root vision. int TEXT_ALIGNMENT_INHERIT standard text alignment. int TEXT_ALIGNMENT_TEXT_END line to the end of the paragraph, for example, ALIGN_OPPOSITE. int TEXT_ALIGNMENT_TEXT_START to the beginning of the paragraph, for example, ALIGN_NORMAL. int TEXT_ALIGNMENT_VIEW_END align to the end of the view, which is ALIGN_RIGHT if the resolved layout of the Direction for LTR and ALIGN_LEFT otherwise. int TEXT_ALIGNMENT_VIEW_START align to the beginning of the view, which is ALIGN_LEFT if the resolved layout of the Direction for LTR and ALIGN_RIGHT otherwise. int TEXT_DIRECTION_ANY_RTL the text direction is using the any-RTL algorithm. int TEXT_DIRECTION_FIRST_STRONG the text direction is using strong algorithm first. int TEXT_DIRECTION_FIRST_STRONG_LTR the text direction is using strong algorithm first. int TEXT_DIRECTION_FIRST_STRONG_RTL the text direction is using strong algorithm first. int TEXT_DIRECTION_INHERIT the text direction is inherited through the ViewGroup int TEXT_DIRECTION_LOCALE the text direction is coming from the Locale system. int TEXT_DIRECTION_LTR The direction of the text is forced to LTR. int TEXT_DIRECTION_RTL the direction of the text is forced to RTL. String VIEW_LOG_TAG The registry tag used by this class with android.util.Log. int VISIBLE This view is visible. public static end wrapper<View, float=> Alpha A Property around alpha functionality handled by the View#setAlpha (float) and View#getAlpha() methods. EMPTY_STATE_SET protected static ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET indicates that the view has no defined states. Static end int protected[] ENABLED_FOCUSED_SELECTED_STATE_SET indicates that the view is enabled, focused, and selected. selected, and your window has the focus. Static end int protected[] ENABLED_FOCUSED_STATE_SET indicates that the view is enabled and has focus. Static end int protected[] ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET indicates that the view is enabled, focused and its window has focus. Static end int protected ENABLED_SELECTED_STATE_SET indicates that the view is on and selected ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET. , selected and its window has focus. Static end int protected ENABLED_STATE_SET indicates that the display is enabled. Static end int protected[] ENABLED_WINDOW_FOCUSED_STATE_SET indicates that the view is enabled and that your window has focus. Static end int protected int[] FOCUSED_SELECTED_STATE_SET indicates that the view is focused and selected. Static int protected int[] FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET indicates that the view is focused, selected, and your window has focus. static end int[] protected int[] FOCUSED_STATE_SET indicates that vision is focused. Static end int protected[] FOCUSED_WINDOW_FOCUSED_STATE_SET indicates that the view has focus and that its window has focus. static end int protected int[] PRESSED_ENABLED_FOCUSED_SELECTED_STATE_SET indicates that the display is </View,> </View,> focused and selected. Int static end protected[] PRESSED_ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET indicates that the preview is pressed, activated, focused, selected, and your window has focus. static end int[] protected PRESSED_ENABLED_FOCUSED_STATE_SET indicates that the preview is pressed, enabled, and focused. Int static end protected[] PRESSED_ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET indicates that the preview is pressed, activated, focused, and your window has focus. Static end int protected PRESSED_ENABLED_SELECTED_STATE_SET indicates that the view is pressed, activated, and selected. Static end int protected[] PRESSED_ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET indicates that the preview is pressed, activated, selected, and your window has focus. static end int[] protected int[] PRESSED_ENABLED_STATE_SET indicates that the preview is pressed and enabled. Static end int protected[] PRESSED_ENABLED_WINDOW_FOCUSED_STATE_SET indicates that the preview is pressed, activated, and your window has focus. Static end int protected int[] PRESSED_FOCUSED_SELECTED_STATE_SET indicates that the view is pressed, focused, and selected. Static end int protected[] PRESSED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET indicates that the preview is pressed, focused, selected, and your window has focus. Static end int[] protected PRESSED_FOCUSED_STATE_SET indicates that the view is pressed and focused. Static end int protected[] PRESSED_FOCUSED_WINDOW_FOCUSED_STATE_SET indicates that the view is pressed, focused and its window has focus. Static end int[] protected PRESSED_SELECTED_STATE_SET indicates that the view is pressed and selected. Int static end protected[] PRESSED_SELECTED_WINDOW_FOCUSED_STATE_SET indicates that the preview is pressed, selected, and your window has focus. static end int[] protected int[] PRESSED_STATE_SET indicates that the preview is pressed. Static end int protected[] PRESSED_WINDOW_FOCUSED_STATE_SET indicates that the preview is pressed and its window has focus. Ultimate static public Property<View, float=> ROTATION A property wrapper around rotation functionality handled by the View#setRotation(float) and View#getRotation() ROTATION_X .<View, float=> public static final Property ROTATION_Y A Property<View, float=> wrapper around the rotationY functionality handled by the View#setRotationY (float) and View#getRotationY() methods . Public static final Property<View, float=> SCALE_X A Property wrapper around scaleX functionality handled by the View#setScaleX(float) and View#getScaleX() methods. public static final Property<View, float=> SCALE_Y A Property wrapper around scaleY functionality handled by the View#setScaleY(float) and View#getScaleY() methods. static end int protected[] SELECTED_STATE_SET indicates that the view is selected. Static end int protected Indicates that the view is selected and that its final window window int[] indicates that its static window int[] indicates that its static window int[]</View,> </View,> </View,> </View,> </View,> the focus. public static final Property<View, float=> TRANSLATION_X A Property wrapper around the translationX functionality handled by the View#setTranslationX(float) and View#getTranslationX() methods. public static final Property<View, float=> TRANSLATION_Y A Property wrapper around the translationY functionality handled by the View#setTranslationY(float) and View#getTranslationY(). public static final Property<View, float=> TRANSLATION_Z A Property wrapper around the translationZ functionality handled by the View#setTranslationZ(float) and View#getTranslationZ() methods protected static end int[] WINDOW_FOCUSED_STATE_SET Indicates that the view window has focus. final publicly owned wrapper<View, float=> X A Property around the x functionality handled by the View#setX(float) and View#getX() methods. final property wrapper<View, float=> y a property around the y functionality handled by the View#setY(float) and View#getY() methods. public static final property wrapper<View, float=> Z A Property around the z functionality handled by the View#setZ(float) and View#getZ() methods. Visualization (context context) Simple-to-use constructor when creating a view from code. View (context context, attributeset attrs) constructor that is called when inflating an XML view. View (context context, attrs AttributeSet, int defStyleAttr) Run XML inflation and apply a class-specific base style from a thematic attribute. View (context context, attributeset attrs, int defStyleAttr, int defStyleRes) Run XML inflation and apply a class-specific base style from a themed attribute or style feature. void addChildrenForAccessibility(ArrayList<View> outChildren) Adds the children in this view relevant to accessibility to the given list as output. void addExtraDataToAccessibilityNodeInfo(Accessibility InformationNodeInfo, String extraDataKey, Bundle Arguments) Adds extra data to an AccessibilityNodeInfo based on an explicit request for additional data. void addFocusables<View> (ArrayList views, direction int) Add any focal views that are descending from that view (possibly including this view if it is focusable in itself) for visualizations.<View> int direction, int focusableMode) Adds any focalviews that are descending from that view (possibly including this view if it is focusable in itself) views. void addKeyboardNavigationClusters<View> (Collection views, direction int) Adds any keyboard navigation cluster roots that are descendants of this view (possibly including this view if it is a cluster root 1 itself) to views. void addOnAttachStateChangeListener (View.OnAttachStateChangeListener listener) Add a listener to attach changes from void addOnLayoutChangeListener (View.OnLayoutChangeListener listener) Add a listener that will be called when the view change limits due to the display change due to the display change due to the preview change due to the preview change due to the preview change due to the view change due to the display change due to the display change due to the display change due to the display change due to the preview change preview due to the preview change due to the preview change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the display change due to the change of display layout processing. addOnunhandledKeyEventListener (View.OnUnhandledKeyEventListener listener) Adds a listener that will receive </View> </View> </View> </View> </View,> </View,> </View,> </View,> </View,> </View,> </View,> </View,> Keyevents. void addTouchables<View> (ArrayList views) Add any touchable views that are descending from this view (possibly including this view if it is touchable in itself) to the visualizations. ViewPropertyAnimator animates() This method returns a ViewPropertyAnimator object, which can be used to animate specific properties in this view. void announceForAccessibility (CharSequence text) Convenience method for sending an Accessibility EventEvent#TYPE_ANNOUNCEMENT AccessibilityEvent to suggest that an accessibility service advertise the specified text to its users. empty autofill (autofillVato value) automatically fills the contents of this visualization with the value. void autofill<AutofillValue> (SparseArray values) Automatically fills the contents of virtual children within this view. void bringToFront() Change the z order of the view in the tree, so it is on top of other sibling views. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. constructing the emptyDrawingCache() This method has been deprecated at api level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Empty Layer() Construction Forces the layer of this display to be created</AutofillValue> </View> </View> this view to be rendered on your layer. call booleanaOnClick() Call directly to any attached OnClickListener. boolean canResolveLayoutDirection() Make sure that the layout direction resolution can be made. boolean canResolveTextAlignment() Make sure that the text alignment resolution can be made. boolean canResolveTextDirection() Make sure that the resolution of the text direction can be made. boolean canScrollHorizontally (direction int) Make sure this view can be scrolled horizontally in a certain direction. boolean canScrollVerically (direction int) Make sure this view can be scrolled vertically in a certain direction. Final cancellation of the emptyDragAndDrop() Cancels a continuous drag-and-drop operation. empty cancellationSOprim () Cancels a long pending press. final cancellation of the emptyPendingInputEvents() Cancel any deferred high-level input events that were previously posted to the event queue. Boolean verificationInputConnectionProxy(Preview) called by InputMethodManager when a view that is not the current target of incoming connection is trying to make a call in the manager. clearAnimation() Cancels any animations for this view. empty clearFocus() Called when this view wants to give up focus. static int combinesMeasuredStates (int curState, int newState) merges two states as returned by getMeasuredState(). void computeScroll() Called by a parent to request that a child update their values to mScrollX and mScrollY if necessary. WindowInsets computeSystemWindowInsets (WindowInsets in, Rect outLocalInsets) Computational sets that must be consumed by this visualization and those that must propagate to those under it. AccessibilityNodeInfo createsAccessibilityNodeInfo() Returns an AccessibilityNodeInfo representing this view from the point of view of an Accessibility Service. empty createContextMenu (ContextMenu menu) Show the context menu for this view. empty destroyDrawingCache() This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have problems compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. WindowInsets dispatchApplyWindowInsets (Insets WindowInsets) Request to apply the window insets given to this view or other view in your subarnão. Boolean Boolean event) Pass a captured pointer event to the focused view. empty dispatchConfigurationSchanged (New Config configuration) Send a notification about a resource configuration changing the view hierarchy. empty dispatchDisplayHint (hint int) Send a hint about whether this view is displayed. boolean dispatchDragEvent (DragEvent event) detects whether this view is enabled and has a drag event listener. empty dispatchDwableHotspotChanged (float x, float y) Dispatches drawableHotspotChanged for all children of this View. empty dispatchFinishTemporaryDetach() Dispatch onFinishTemporaryDetach() to this View and its direct children, if this is a container view point. 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. Dispatch booleanNestedFling (floating speedX, floating speedY, boolean consumed) Dispatches a pitch to a nested scrolling parent. Dispatch booleanNestedPreFling (floating speedX, floating speedY) Dispatch a pitch to a nested scrolling parent before being processed by this view. boolean dispatchNestedPrePerformAccessibilityAction (int action, Bundle arguments) Reports an accessibility action to parents of this view for delegated processing. boolean dispatchNestedPreScroll (int dx, int dy, int[] consumed, int[] offsetInWindow) Dispatch a step of a nested parchment in progress before this view consumes any part of it. boolean dispatchNestedScroll (int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed, int[] offsetInWindow) Dispatch a step of a nested parchment in progress. empty dispatchPointerCaptureChanged (boolean hasCapture) boolean dispatchPopulateAccessibilityEvent (AccessibilityEvent event) Dispatches an Accessibility Event to the View first, and then to its children to add its text content to the event. Empty dispatchProvideAutofillStructure (Display structure, flags int) Dispatches the creation of a ViewStructures for autofill purposes in the hierarchy, when an assist structure is being created as part of an autocomplete request. empty dispatchProvideStructure (Display Structure) Dispatch creation of the ViewStructure in the hierarchy. empty dispatchStartTemporaryDetach() Dispatch onStartTemporaryDetach() to this View and its direct children if this is a container view point. empty dispatchSSO emptySystemUiVisibilityChanged (visibility int) This method has been deprecated at API level 30. Use WindowInsets#isVisible(int) to find out about the visibility of the setting an OnApplyWindowInsetsListener in this view. boolean dispatchTouchEvent (motionEvent event) Pass the touch screen motion event to the target view, or this view if it is the target. dispatch booleanoTrackballEvent (MotionEvent event) MotionEvent) a trackball motion event up to the focused view. boolean dispatchSDhandledMove (View focused, in the direction) This method is the last chance for focused vision and its ancestors to respond to an arrow key. Empty dispatchWindowFocusChanged (boolean hasFocus) Called when the window containing this view gains or loses the focus of the window. void dispatchWindowInsetsAnimationEnd (WindowInsetsAnimation animation) Dispatchs windowInsetsAnimation.Callback#onEnd (WindowInsetsAnimation) when the Window Insets animation ends. void dispatchWindowInsetsAnimationPrepare (WindowInsetsAnimation animation) DispatchwindowInsetsAnimation.Callback#onPrepare(WindowInsetsAnimation) when the Window Insets animation is being prepared. WindowInsets dispatchWindowInsetsAnimationProgress (Insets windowInsets, List<WindowInsetsAnimation> runningAnimations) Dispatches WindowInsetsAnimation.Callback#onProgress(WindowInsets, List) when the Window Insets animation makes progress. WindowInsetsAnimation.Bounds dispatchWindowInsetsAnimationStart (windowInsetsAnimation animation, WindowInsetsAnimation.Boundsbounds) Dispatches WindowInsetsAnimation.Callback#onStart(WindowInsetsAnimation, Bounds) when the Window Insets animation is started. Empty dispatchWindowSystemUiVisiblityChanged(int visible) This method was deprethered at API level 30. SystemUiVisibility flags are depreced. Use windowinsetscontroller instead. Empty dispatchWindowVisibilityChanged (int visibility) Dispatches a window visibility change down from the view hierarchy. Empty drawing (canvas screen) Manually renderize this view (and all your children) to the given screen. void drawableChangedHotspot (float x, float y) This function is called whenever the display hotspot changes and needs to be propagated to drawables or views of children managed by the view. View findFocus() Find the view in the hierarchy rooted in this view that currently has focus. <T extends= view=>T findViewById(int id) Finds the first descending view with the given ID, the view itself if the ID matches getId(), or null if the ID is invalid < 0)= or= there= is= no= matching= in= the= hierarchy.= final=> <T extends= view=> (T findViewWithTag(object tag) Look for a child view with the particular tag. void findVersWithText(ArrayList<View> outViews, CharSequence searched, int flags) Find views that contain given text. View focusSearch (direction int) Find the nearest view in the specified direction that may have focus. Empty forceHasOverlappingRendering (boolean hasOverlappingRendering) sets the behavior for overlapping rendering for this visualization (see hasOverlappingRendering() for more details about this behavior). void forceLayout() Forces this view to be set during the next layout pass. int static generateViewId() Generate an appropriate value for setId(int). CharSequence getAccessibilityClassName() Return the class name of this object to be used for accessibility purposes. View.AccessibilityDelegate getAccessibilityDelegate() Returns delegate to implement accessibility support via </View> </T> </T> </WindowInsetsAnimation> </WindowInsetsAnimation> int getAccessibilityLiveRegion() Receives live region mode for this View. AccessibilityNodeProvider getAccessibilityNodeProvider() Receives the provider to manage a virtual view hierarchy rooted in this view and reported to Accessibility Services that exploit window content. CharSequence getAccessibilityPaneTitle() Get the panel title for accessibility purposes. int getAccessibilityTraversalAfter() gets the id of a view after which it is visited on accessibility traversal. int getAccessibilityTraversalBefore() gets the id of a view before which it is visited on accessibility traversal. float getAlpha() The opacity of the view. Get Animation animation() Get the animation currently associated with this view. Matrix getAnimationMatrix() Return the current view transformation matrix. IBinder getApplicationWindowToken() Retrieve a unique token by identifying the actual top-level window of the window to which this view is attached. int[] getAttributeResolutionStack (attribute int) returns the ordered list of resource ID that are considered when resolving attribute values for this view. Map<Integer, integer=> getAttributeSourceResourceMap() Returns the mapping of the attribute resource ID to source resource id where the attribute value was set. String[] getAutofillHints() Receives hints that help an AutofillService determine how to automatically filter the visualization with user data. Final AutofillId getAutofillId() gets the unique and logical identifier of this view in the activity, for autofill purposes. int getAutofillType() Describes the type of autofill in this view, so that an AutofillService can create the appropriate AutofillValue by automatically filtering the view. AutofillValue getAutofillValue() gets the current autofill value of the View. GetBackground() Drawable Get the background drawable BlendMode getBackgroundTintBlendMode() Return the blending mode used to apply the tint to the bedding, if specified. ColorStateList getBackgroundTintList() Return the tint applied to the foundation, if specified. PorterDuff.Mode getBackgroundTintMode() Return the blending mode used to apply the tint to the drawn bedding, if specified. int getBaseline() Return the widget baseline compensation from the upper limit of the widget. int end getBottom() Lower position of this view relative to its parent. float getCameraDistance() gets the distance along the Camera's Z axis for this view. boolean getClipBounds(Rect outRect) fills an output rectangle with the clip boundaries of the view, returning true if successful or false if the clip limits of the view are null. Rect getClipBounds() Returns a copy of the current clips. final boolean getClipToOutline() Returns whether the Outline should be used to crop the contents of the View. finalCaptureSession getContentCaptureSession() Receives the session used to notify content capture events. CharSequence getContentDescription() Returns the description of the contents of the view. final context getContext() </Integer,> </Integer,> the context in which the preview is running, through which you can access the current theme, resources, etc. final boolean getDefaultFocusHighlightEnabled() /** Returns whether this View should use a default focus highlight when it focuses, but does not have R.attr.state_focused set in its background. static int getDefaultSize (size int, int medeSpec) Utility to return a default size. Display getDisplay() Receives the logical display to which the display window was attached. int final[] getDrawableState() Return a series of resource IDs from drawable states representing the current state of the view. Bitmap getDrawingCache() This method has been deprecated at api level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Bitmap getDrawingCache (boolean autoScale) This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. int getDrawingCacheBackgroundColor() method has been deprecated at api level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a network network performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. int getDrawingCacheQuality() This method was deprethered at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. empty getDrawingRect (Rect outRect) Return the visible drawing boundaries of your view. getDrawingTime() Return the time at which the drawing of the visualization hierarchy began. the base elevation of this view relative to its parent, in pixels. int getExplicitStyle() Returns the resource ID for the specified style using style=... in attributeset support XML element or resources#ID_NULL otherwise, if not specified or not applicable. boolean getFilterTouchesWhenObscured() Understands whether the structure should drop touches when the view window is obscured by another visible window. boolean getFitsSystemWindows() Check for setFitsSystemWindows(boolean) state. int getFocusable() Returns the focalable setting for this view. ArrayList<View> getFocusables (direction int) Find and return all focusable views that are descending from this view, possibly including this view if it is focusable on itself. void getFocusedRect(Rect r) When a has focus and the user navigates away from it, the next view is searched from the rectangle filled by this method. Drawable getForeground() Returns the drawable used as the foreground of this view. int getForegroundGravity() Describes as </View> </View> in the foreground is positioned. BlendMode getForegroundTintBlendMode() Return the blendmode used to apply the tint to the drawn foreground, if specified. ColorStateList getForegroundTintList() Return the tint applied to the foreground drawn, if specified. PorterDuff.Mode getForegroundTintMode() Return the blending mode used to apply the tint to the drawn foreground, if specified. final boolean getGlobalVisibleRect (Rect r) boolean getGlobalVisibleRect (Rect r, Point globalOffset) If any part of this view is not cut off by any of your parents, then return that area in r in global coordinates (root). Handler getHandler() final boolean getHasOverlapping() Returns the value for overlapping rendering that is used internally. int final getHeight() Return the height of your vision. empty getHitRect (Rect outRect) Hit rectangle at parent coordinates int getHorizontalFadingEdgeLength() Returns the size of the faded horizontal edges used to indicate that more content in this view is visible. Drawable getHorizontalScrollbarThumbDrawable() Returns the Drawable currently configured to the thumb of the horizontal scrollbar if it exists, null otherwise. Drawable getHorizontalScrollbarTrackDrawable() Returns the Drawable currently configured to the horizontal scrollbar range if it exists, null otherwise. int getId() returns the identifier of this visualization. int getImportantForAccessibility() gets the mode to determine whether this view is important for accessibility. int getImportantForAutofill() gets the mode to determine whether this view is important for autocomplete. int getImportantForContentCapture() gets the mode to determine whether this view is important for content capture. boolean getKeepScreenOn() returns whether the screen should remain in effect, corresponding to the current value of KEEP_SCREEN_ON. KeyEvent.DispatcherState getKeyDispatcherState() Return the global KeyEvent.DispatcherState to the window of this view. 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 visualization. int getLayoutDirection() Returns the resolved layout direction for this view. ViewGroup.LayoutParams getLayoutParams() Get the LayoutParams associated with this view. int end getLeft () Left position of this view relative to its parent. final boolean getLocalVisibleRect (Rect r) void getLocationInSurface (int[] location) Computation of the view coordinate within the surface. empty getLocationInWindow(int[] outLocation) calculates the coordinates of this view in your window. empty getLocationOnScreen (int[] outLocation) calculates the coordinates of this view on the screen. Matrix getMatrix() The transformation matrix of this view, which is calculated based on the rotation properties, and current pivot. int end getMeasuredHeight() as getMeasuredHeightAndState(), but only returns the raw height component (that is, the result is masked by by final int getMeasuredHeightAndState() Return the total height measurement information for this view computed by the most recent call to measure (int, int). int final getMeasuredState() Return only the state bits of getMeasuredWidthAndState() and get MeasuredHeightAndState(), combined into an integer. int end getMeasuredWidth() as getMeasuredWidthAndState(), but only returns the raw width component (that is, the result is masqueraded by MEASURED_SIZE_MASK). final int getMeasuredWidthAndState() Return the total width measurement information to this visualization, as calculated by the most recent call to measure (int, int). int getMinimumHeight() Returns the minimum view height. int getMinimumWidth() returns the minimum view width. int getNextClusterForwardId() gets the root id of the next keyboard navigation cluster. int getNextFocusDownId() gets the view id to use when the next focus is FOCUS_DOWN. int getNextFocusForwardId() gets the view id to use when the next focus is FOCUS_FORWARD. int getNextFocusLeftId() gets the view id to use when the next focus is FOCUS_LEFT. int getNextFocusRightId() gets the view id to use when the next focus is FOCUS_RIGHT. int getNextFocusUpId() gets the view id to use when the next focus is FOCUS_UP. View.OnFocusChangeListener getOnFocusChangeListener() Returns the focus change callback registered to this view. int getOutlineAmbientShadowColor() ViewOutlineProvider getOutlineProvider() returns the current viewOutlineProvider of the view, which generates the Outline that defines the shape of the shadow that casts and allows for contour clipping. int getOutlineSpotShadowColor() int getOverScrollMode() returns over-scroll mode for this view. ViewOverlay getOverlay() Returns the overlay for this view, creating it if it does not already exist. int getPaddingBottom() Returns the bottom padding of this view. int getPaddingEnd() Returns the final fill of this view, depending on the resolved layout direction. int getPaddingLeft() Returns the left fill of this view. int getPaddingRight() Returns the right fill of this view. int getPaddingStart() Returns the initial fill of this view, depending on the resolved layout direction. int getPaddingTop() Returns the top padding of this view. end ViewParent getParent() Receives the parent of this view. ViewParent getParentForAccessibility() Receives the parent for accessibility purposes. 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. PointerIcon getPointerIcon() Receives the pointer icon for the current view. Resources receive Resources() Returns the resources associated with this view. final boolean getRevealOnFocusHint() Returns the preference of this to reveal behavior when it gains focus. int end getRight() Correct position of this view relative to its parent. View getRootView() Finds the highest view in the current Hierarchy. Window sets receive RootWindowInsets() Provide original window sets that are sent to the display 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 non-escalating width of the view. float getScaleY() The amount that the view is sized in y around the pivot point, as a proportion of the non-escalating height of the view. int getScrollBarDefaultDelayBeforeFade() Returns the delay before the scroll bars disappear. int getScrollBarFadeDuration() Returns the duration of the scrollbar fade. int getScrollBarSize() Returns the size of the scrollbar. int getScrollBarStyle() Returns the current style of the scroll bar. int getScrollIndicators() Returns a bit mask representing the enabled scroll indicators. int end getScrollX() Return the left scrolled position of this view. int end getScrollY() Return the top scrolled position of this view. int getSolidColor() Undo this if your vision is known to always be drawn on top of a solid color background, and need to draw faded edges. int getSourceLayoutResId() A View can be inflated from an XML layout. final charsequence getStateDescription() Returns the view state description. StateListAnimator getStateListAnimator() Returns the current StateListAnimator if it exists. List<Rect> getSystemGestureExclusionRects() Retrieve the list of areas within the post-layout coordinate space of this view where the system should not intercept touch or other pointing device gestures. int getSystemUiVisibility() This method was deprethered at API level 30. SystemUiVisibility flags are depreced. Use windowinsetscontroller instead. GetTag object() Returns the tag of this view. GetTag object returns the tag associated with this view and the specified key. int getTextAlignment() Return the resolved text alignment. int getTextDirection() Return the direction of resolved text. CharSequence getTooltipText() Returns the text of the display tooltip. int end getTop() Top position of this view relative to its parent. TouchDelegate getTouchDelegate() Receives touchdelegate for this view. ArrayList<View> getTouchables() Find and return all touchable views that are descending from that view, possibly including this view if it is touchable in itself. float getTransitionAlpha() This property is intended only for use by the Fade transition, which animates it to produce a visual translucency that does not effect (or is affected by) the actual alpha property. String Returns the name of the view to use to identify visualizations in transitions. 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. </View></Rect> </View></Rect> getTranslationZ() The depth location of this view in relation to its elevation. get uUniqueDrawingId() Get the handle used for this visualization by the drawing system. int getVerticalFadingEdgeLength() Returns the size of the faded vertical edges used to indicate that more content in this view is visible. int getVerticalScrollbarPosition() Drawable getVerticalScrollbarThumbDrawable() Returns the currently configured Drawable to the thumb of the vertical scrollbar if it exists, null otherwise. Drawable getVerticalScrollbarTrackDrawable() Returns the Drawable currently configured to the vertical scrollbar range if it exists, null otherwise. int getVerticalScrollbarWidth() Returns the width of the vertical scroll bar. ViewTreeObserver getViewTreeObserver() Returns the ViewTreeObserver to the hierarchy of this view. int getVisibility() Returns the visibility status for this view. int final getWidth() Return the width of your view. WindowId getWindowId() Retrieve the WindowId for the window to which this view is currently attached. WindowInsetsController getWindowInsetsController() Retrieves the only WindowInsetsController from the window to which this view is attached. int getWindowSystemUiVisibility() This method was deprethered at API level 30. SystemUiVisibility flags are depreced. Use windowinsetscontroller instead. IBinder getWindowToken() Retrieve a unique token by identifying the window to which this view is attached. int getWindowVisibility() Returns the current visibility of the window to which this view is attached (gone, invisible, or visible). empty getWindowVisibleDisplayFrame (Rect outRect) Retrieve the overall size of the visible display in which the window to which this view is attached has been positioned. float getX() The visual position x of this view, in pixels. float getY() The visual position y of this view, in pixels. float getZ() The visual position z of this view, in pixels. boolean hasExplicitFodable() Returns true if this view is focusable or if it contains an reachable view for which it has explicitfodible returns() is true. boolean hasFocus() Returns true if this view focuses on itself, or is the ancestor of the vision that has focus. boolean hasFocusable() returns true if this view is focusable or if it contains an reachable view for which the hasFocusable() returns true. boolean hasNestedScrollingParent() returns true if this view has a nested scrolling parent. boolean hasOnClickListeners() Return if this view has an OnClickListener attached. boolean hasOnLongClickListeners() Return if this view has an OnLongClickListener attached. boolean hasOverlappingRendering() Returns if this view has overlapping content. boolean has PointerCapture() checks the capture status of the pointer. boolean hasTransientState() Indicates whether the view is currently tracking the state that the application should not worry about saving and restoring, but that the framework should take special note to preserve when possible. boolean hasWindowFocus() Returns Returns if this view is in a window that currently has window focus. static View inflate (context context, int resource, viewgroup root) inflates a view of an XML resource. invalidate() Invalidate the entire vision. invalidated empty (Dirty Rect) This method has been deprecated at API level 28. Switching to accelerated hardware rendering in API 14 has reduced the importance of dirty rectangle. In API 21, the given rectangle is ignored entirely in favor of an internally calculated area. Because of this, customers are encouraged to just call it invalidate(). invalidated void (int l, int t, int r, int b) This method was deprecated at API level 28. Switching to accelerated hardware rendering in API 14 has reduced the importance of dirty rectangle. In API 21, the given rectangle is ignored entirely in favor of an internally calculated area. Because of this, customers are encouraged to just call it invalidate(). Invalidated VoidSays (Drawable drawable) Invalidates the specified Drawable. empty invalidateOutline() Called to rebuild the outline of this view from your Boolean ViewOutlineProvider isAccessibilityFocused() returns whether this view is focused on accessibility. boolean isAccessibilityHeading() Understands whether this view is a title for accessibility purposes. 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 isContextClickable() Indicates whether this view reacts to context clicks or not. boolean isDirty() True if this view has changed since the last time it was drawn. boolean isDrawingCacheEnabled() This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. boolean isDuplicateParentStateEnabled() of his father. boolean isEnabled() Returns the status enabled for this view. boolean

isFocusable() Returns if this View is currently able to have focus. final boolean final When a view is focusable, it may not want to focus when in touch mode. boolean isFocused() Returns true if this view has final boolean focus isFocusedByDefault() Returns whether this view should receive focus when the focus is restored to the view hierarchy that contains this view. boolean isForceDarkAllowed() See setForceDarkAllowed(boolean) boolean isHapticFeedbackEnabled() boolean isHardwareAccelerated() Indicates whether this view is connected to an accelerated hardware window or not. boolean isHorizontalScrollBarEnabled() Indicate whether the horizontal scroll bar should be drawn or not. boolean isHovered() Returns true if the view is currently hovered. boolean isImportantForAccessibility() Calculates whether this view should be exposed for accessibility. boolean isImportantForAutofill() Suggests to the Android System whether the AssistStructure.ViewNode associated with this view is considered important for autocomplete purposes. final boolean isImportantForContentCapture() Suggests to the Android System whether this view is considered important for content capture, based on the value explicitly defined by setImportantForContentCapture(int) and heuristic when it is IMPORTANT_FOR_CONTENT_CAPTURE_AUTO. boolean isInEditMode() Indicates whether this View is currently in edit mode. boolean isInLayout() returns whether the view hierarchy is currently passing a layout pass. boolean isInTouchMode() Returns whether the device is currently in touch mode. final boolean isKeyboardNavigationCluster() returns whether this view is a root of a keyboard navigation cluster. boolean isLaidOut() returns true if this view has passed at least one layout since it was last attached or separated from a window. boolean isLayoutDirectionResolved() boolean isLayoutRequested() Indicates whether or not the layout of this view will be requested during the next hierarchy layout pass. boolean isLongClickable() Indicates whether this view reacts to long-click events or not. boolean isNestedScrollingEnabled() returns true if nested scrolling is enabled for this view. boolean isOpaque() Indicates whether this display is opaque. boolean isPaddingRelative() Return if the fill has been set through relative values defined PaddingRelative (int, int, int) or through boolean isPivotSet() Returns whether or not a pivot was defined by a call to define PivotX or setPivotY (float). boolean isPressed() Indicates whether the view is currently in a pressed state. boolean isSaveEnabled() Indicates whether this view will save its state (that is, whether your onSaveInstanceState() method is called). boolean isSaveFromParentEnabled() Indicates whether the entire hierarchy under this view will save its state when a state save traverse occurs from its parent. boolean isScreenReaderFocusable() Returns whether the display should be treated as a Focalable by screen reader Tools. boolean isScrollContainer() Indicates whether this view is one of the scrollable containers in your window. boolean isScrollbarFadingEnabled() Returns true if scroll bars disappear when this display is not scrolling boolean isSelected() Indicates the selection state of this view. Final boolean isShowingLayoutBounds() Returns true when the view is attached and the system developer setting to show layout boundaries is enabled or false otherwise. boolean isShown() Returns the visibility of this view and all its boolean ancestors isSoundEffectsEnabled() final boolean isTemporarilyDetached() Says whether the view is in the state between onStartTemporaryDetach() and onFinishTemportemporaryDetach(). boolean isTextAlignmentResolved() boolean isTextDirection boolean isVerticalScrollBarEnabled() Indicate whether the vertical scroll bar should be drawn or not. boolean isVisibleToUserForAutofill(int virtualId) Calculates whether this virtual autocomplete view is visible to the user. empty hopSdrawablesToCurrentState() Call Drawable#jumpToCurrentState() on all Drawable objects associated with this visualization. View keyboardNavigationClusterSearch (View currentCluster, direction int) Find the nearest keyboard navigation cluster in the specified direction. empty layout (int l, int t, int r, int b) Assign a size and position to a view and to all its descendants This is the second phase of the layout engine. final measure of void (intMeasureSpec width, int heightMeasureSpec) This is called to find out how large a view should be. Empty offsetReftAndRight (int offset) Compensate for the horizontal location of this view by the specified amount of pixels. Empty offsetTopAndBottom (int offset) Compensate for the vertical location of this view by the specified number of pixels. WindowInsets inSets onApplyWindowInsets (Insets windowInsets) called when the view should apply WindowInsets according to its internal policy. empty onCancelPendingPendingEvents() called as a result of a call to cancelPendingPendingEvents() in this view or a parent view. boolean onCapturedPointerEvent (motionEvent event) Implement this method to handle pointer events captured boolean onCheckIsTextEditor() Make sure that 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 visualization. boolean onDragEvent (DragEvent event) Handles drag events sent by the system after a call to startDragAndDrop(). empty on drawforeground (screen screen) Draw any content first for this view. boolean onFilterTouchEventForSecurity (MotionEvent event) filters the touch event to enforce security policies. empty in finishtemporarydetach() called after the starttemporarydetach() when the container finishes changing the view. Boolean Boolean event) Implement this method to handle generic motion events. empty onHoverChanged (hovered boolean) implement this method to handle hover state changes. boolean onHoverEvent (motionEvent event) implement this method to handle hovering events. empty onInitializeAccessibilityEvent (AccessibilityEvent event) Initializes an Accessibility Event with information about this view that is the source of the event. empty onInitializeAccessibilityNodeInfo (accessibility informationNodeInfo) Initializes an AccessibilityNodeInfo with information about this visualization. boolean onKeyDown (int keyCode, keyEvent event) Default keyevent implementation.Callback#onKeyDown (int, KeyEvent): Run the display press when KeyEvent#KEYCODE_DPAD_CENTER or KeyEvent#KEYCODE_ENTER is released, if the view is enabled and clickable. boolean onKeyLongPress (int keyCode, keyEvent event) Default implementation of KeyEvent.Callback#onKeyLongPress (int, KeyEvent): always returns false (does not handle the event). boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) Default implementation of KeyEvent.Callback#onKeyMultiple (int, int, KeyEvent): always returns false (does not handle the event). boolean onKeyPreIme (int keyCode, keyEvent event) handles a key event before being processed by any input method associated with the view hierarchy. boolean onKeyShortcut (int keyCode, keyEvent event) called in the focused view when a key shortcut event is not handled. boolean onKeyUp (int keyCode, keyEvent event) KeyEvent default implementation.Callback#onKeyUp(int, KeyEvent): Click the view when KeyEvent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER, or KeyEvent#KEYCODE_SPACE is released. empty in PointerCaptureChange (boolean hasCapture) called when the window has just acquired or lost the pointer capture. empty onPopulateAccessibilityEvent (AccessibilityEvent event) called the dispatchPopulateAccessibilityEvent (android.view.accessibility.AccessibilityEvent) giving this View a chance to fill the accessibility event with its text content. empty in the ProvideAutofillStructure (Display structure, flags int) Populates a Display Structure to completely fill an autofill request. empty in the ProvideAutofillVirtualStructure (Display structure, flags int) Populates a Display Structure containing virtual children to fully fill out an autocomplete request. empty in the ProvideContentCaptureStructure (Display structure, flags int) Populates a Display Structure for content capture. empty in the View Structure (view structure structure) called when the assistance structure is being retrieved from a view as part of Activity.onProvideAssistData. empty onProvideVirtualStructure (Visualization Structure) Called when the assistance structure is being retrieved a view as part of Activity.onProvideAssistData to generate additional virtual structure under this view. PointerIcon onResolvePointerPointerIcon (motionEvent event, int pointerIndex) returns the pointer icon for the motion event or null if it does not specify the icon. empty inRtlPropertiesChanged (intDirection layout) intDirection) when any RTL (layout direction or text direction or text alignment) property has changed. empty on ScreenStateChanged (intState screen) This method is called whenever the screen state in this view is attached to changes. empty in StartTemporaryDetach() This is called when a container will temporarily detach a child, with ViewGroup#detachViewFromParent(View). 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. empty onVisibilityAggregated (boolean isVisible) called when user visibility of this view is potentially affected by a change in this view itself, an ancestral view, or the window to which that view is attached. empty in WindowFocusChanged (boolean hasWindowFocus) called when the window that contains this view gains or loses focus. empty in WindowSystemUiVisibilityChanged(int visible) This method was deprethered at API level 30. SystemUiVisibility flags are depreced. Use windowinsetscontroller instead. boolean performAaction (int action, Bundle arguments) performs the accessibility action specified in the view. boolean performClick() Call the OnClickListener of this view if set. boolean performContextClick (float x, float y) Call the OnContextClickListener from this view, if set. boolean performContextClick() Call the OnContextClickListener of this view, if set. boolean performHapticFeedback(int feedbackConstant) BZZZTT!! 1! Provide the user with a plyth feedback for this view. boolean performHapticFeedback (int feedbackConstant, int flags) BZZZTT!! 1! How to run HapticFeedback(int), with additional options. boolean performLongClick (float x, float y) Calls the OnLongClickListener from this view, if set. boolean performLongClick() Calls the OnLongClickListener from this view, if set. void playSoundEffect (int soundConstant) Play a sound effect for this view. boolean posts (runnable action) causes Runnable to be added to the message queue. posteilesadoSa (runnable action, long delayMillis) causes Runnable to be added to the message queue, to run after the specified amount of time eats. empty postInvalidate() Cause an invalidated to happen in a subsequent loop through the event loop. Empty postInvalidate (int left, int top, int right, int bottom) Cause an invalidated of the specified area to happen in a subsequent loop through the event loop. Invalidated post emptyDelayed (long delayMilliseconds, int left, int top, int right, int bottom) will cause an invalidated specified area to happen in a subsequent loop through the event loop. Empty postInvalidatedDelayed (long delayMilliseconds) Cause an invalidated in a subsequent loop through the event loop. Empty postInvalidateOnAnimation (int left, int top, int right, bottom int) Cause an invalidated of the specified area to happen in the next animation time step, typically the next display frame. Empty Empty Make an invalidated one happen in the next animation time step, typically the next display frame. empty postOnAnimation (runnable action) causes Runnable to run in the next animation time step. empty postOnAnimationDelayed causes Runnable to run in the next step of animation time after the specified amount of time eats. empty update (Dadsedistado) Call this to force a view to update its drawable state. release the empty PointerCapture() releases the pointer capture. boolean removes Callbacks (runnable action) Removes the specified Runnable from the message queue. remove emptyOnAttachStateChangeListener (View.OnAttachStateChangeListener listener) Remove a listener to attach state changes. remove the emptyOnLayoutChangeListener (View.OnLayoutChangeListener listener) Remove a listener for layout changes. Empty removeOnUnhandledKeyEventListener (View.OnUnhandledKeyEventListener listener) Removes a listener that will receive unhandled KeyEvents. request for emptyApplyInsets() Request that a new dispatch of onApplyWindowInsets (android.view.WindowInsets) be performed. This method has been deprecated at API level 20. Use requestApplyInsets() for newer versions of the platform. the final Focus boolean request (int direction) Call this to try to focus on a specific view or one of its descendants and give you a hint about which direction the focus is going. boolean request finalFocus() Call this to try to focus on a specific view or one of its descendants. booleanaFocus request (direction int, rect previouslyFocusedRect) Call this to try to focus on a specific view or one of its descendants and give you tips on the direction and a specific rectangle from where the focus comes from. the final Boolean RequestFocusFromTouch() Call this to try to focus on a specific view or one of its descendants. Empty requestLayout() Call this when something has changed, which has invalidated the layout of this view. Empty requestProsela() Requests the capture mode of the pointer. booleanaRectangleOnScreen request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. booleanaRectangleOnScreen request (rect rectangle, immediate boolean) Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. Final empty requestInbufferedDispatch (int source) Request unoffered submission of the event source class given to this view. Final empty requestInbufferedDispatch (MotionEvent event) Request the unoffered submission of the given flow of Motion Events to this View. Final T <T extends= view=> requerViewById(int id) Finds the first descending view with the given ID, the view itself if the ID the getId(), or post an IllegalArgumentException if the ID is invalid or there is no corresponding view in the hierarchy. </T> </T> Version of resolveSizeAndState (int, int, int) returning only the MEASURED_SIZE_MASK bits of the result. statization int resolveSizeAndState (size int, int measureSpec, int childMeasuredState) Utility to reconcile a desired size and state, with restrictions imposed by a MeasureSpec. Boolean restoreDefaultFocus() Focuses on the default focus view in the view hierarchy that has this view as a root. void restoreHierarchyState<Parcelable> (sparseArray container) Restore the frozen state of this visualization hierarchy from the given container. final empty savedAttributeDataForStyleable (Context, int[] int[] attrs AttributeSet, TypedArray t, int defStyleAttr, int defStyleRes) Stores debug information about attributes. void saveHierarchyState<Parcelable> (sparseArray container) Store the frozen state of this hierarchy in the given container. 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. empty sendAccessibilityEventUnchecked (AccessibilityEvent event) This method behaves exactly like sendAccessibilityEvent (int), but takes as argument an empty Accessibility Event and does not perform a check if accessibility is enabled. delegate to implement accessibility support via composition (as opposed to inheritance). void setAccessibilityHeading (boolean isHeading) Set if the view is a title for a content section for accessibility purposes. empty setAccessibilityLiveRegion (int mode) sets the live region mode for this view. Empty setThe empty configurationSibilityPaneTitle (CharSequence accessibilityPaneTitle) Visually distinct part of a window with window-like semantics are considered panepanels for accessibility purposes. empty setAccessibilityTraversalAfter (int afterId) Sets the id of a visualization after which it is visited in accessibility traversal. empty definedAccessibilityTraversalBefore (int beforeId) Defines the id of a view before which it is visited in accessibility traversal. Empty setActivated (boolean enabled) Changes the activated state of this view. empty definedAlpha (float alpha) Sets the opacity of the view to a value from 0 to 1, where 0 means that the view is completely transparent and 1 means that the view is completely opaque. Empty setAnimation (animation animation) defines the next animation to play for this view. defined emptyAnimationMatrix changes the transformation matrix in the view. empty setAutofillHints (String... autofillHints) Defines tips that help an AutofillService determine how to automatically filter the view with user data. void setAutofillId (AutofillId id) sets the unique and logical identifier of this visualization in the activity, for autofill purposes. Empty setBackground (drawing background) Define</Parcelable> </Parcelable> </Parcelable> background for a particular Drawable, or remove the bottom. Empty setSa background color (color int) Sets the background color for this display. Empty setBackgroundDrawable (drawable background) This method has been deprecated at API level 16. use setBackground (android.graphics.drawable.Drawable) instead of void setBackgroundResource (int resid) Set the background for a given resource. void SetBackgroundTintBlendMode(BlendMode blendMode) Specifies the blending mode used to apply the shade specified by the setBackgroundTintList (android.content.res.ColorStateList)} to the drawn background. void setBackgroundTintList (ColorStateList tint) Applies a tint to the drawable bedding. void SetBackgroundTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the hue specified by the SetBackgroundTintList (android.content.res.ColorStateList)} to the drawn background. Final set of emptySestratto (bottom int) Sets the lower position of this view relative to its parent. Empty setCameraDistance sets the distance along the Z axis (orthogonal to the X/Y plane on which visualizations are drawn) from the camera to this view. empty setClickable (clickable boolean) enables or disables click events for this view. Empty setClipBounds (Rect clipBounds) Defines a rectangular area in this view for which the display will be cropped when it is drawn. Empty setClipToOutline (Boolean clipToOutline) Defines whether the Display Outline should be used to crop the contents of the View. empty definedContentCaptureSession (ContentCaptureSession contentCaptureSession) defines the (optional) ContentCaptureSession associated with this view. void definedDescription (Charsequence description content) defines the description of the contents of the view. empty definedContextClickable (boolean contextClickable) Allows or disables the context by clicking for this view. Empty setDefaultFocusHighlightEnabled (default booleanfocushighlightEnabled) Defines whether this view should use a default focus highlight when it focuses, but does not have R.attr.state_focused set in its background. Empty setDrawingCacheBackgroundColor (color int) This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual visualizations, it is recommended to create a Screen from a Bitmap or and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For screenshots of the for feedback reports or unit testing is recommended. Empty setDrawingCacheEnabled (boolean enabled) This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. empty definedDrawingCacheQuality (quality int) This method was deprethered at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. empty configuredDuplicateParentStateEnabled (boolean enabled) Allows or disables duplication of the parent state in this view. Empty setElevation (float elevation) Sets the base elevation of this view, in pixels. Empty setSeabled (boolean enabled) Set the enabled state of this view. Empty setFadingEdgeLength (length int) Set the size of the faded border used to indicate that more content in this view is available. Set of emptyFilterTouchesWhen obscured (boolean enabled) Defines whether the structure should touches when the display window is obscured by another visible window. void setFitsSystemWindows (boolean fitSystemWindows) Defines whether or not this view should explain system screen decorations, such as the status bar and inset its contents; that is, to control whether the default implementation of will be executed. Empty setFocusable (boolean focusable) Set whether this view can receive focus. empty defined Semsetosessos (int focaly) Defines whether this visualization can receive focus. Empty setFocusableInTouchMode (boolean focusableInTouchMode) Set whether this view can receive focus while in touch mode. Empty setFocusedByDefault (boolean isFocusedByDefault) defines whether this view should receive focus when the focus is restored to the view hierarchy that contains this view. void setForceDarkAllowed (boolean allow) Defines whether or not the dark force is allowed to apply to this view. Empty setForeground (drawable foreground) Provide a Drawable that must be rendered on top of all the contents of the view. Empty setTo the gravity of the ground (gravity int) describes how the foreground is positioned. Empty setForegroundTintBlendMode(BlendMode blendMode) Specifies the blending mode used to apply the shade specified by setForegroundTintList (android.content.res.ColorStateList)} to the drawn background. empty setForegroundTintList (colorStateList tint) Applies a tint to the drawable foreground. Empty setForegroundTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the tine specified by setForegroundTintList (android.content.res.ColorStateList)} to the drawn background. void setHapticFeedbackEnabled (boolean hapticFeedbackEnabled) Set whether this view should have haptic feedback for events such as long presses. empty defined TransientState Has (boolean hasTransientState) set whether this view is currently tracking the transient state that the frame should try to preserve when possible. Empty setHorizontalFadingEdgeEnabled (horizontal booleanaFadingEdgeEnabled) Set whether horizontal edges should be faded when this view is scrolled horizontally. empty setHorizontalScrollBarEnabled (horizontal booleanAScrollBarEnabled) Set whether the horizontal scroll bar should be drawn or not. Empty setHorizontalScrollbarThumbDrawable (Drawable drawable) Sets the set of unalike horizontal voidsHorizontalScrollbarTrackDrawable (Drawable drawable) Sets the set of drawable hovered horizontal strip voids (boolean hovered) Defines whether the display is currently hovered. void setId (int id) sets the handle for this view. Empty definedSportantForAccessibility(int mode) Defines how to determine whether this view is important for accessibility, which is whether it triggers accessibility events and whether it is reported to accessibility services that query the screen. Empty setPortanteParaAutofill (int mode) Sets the mode to determine whether this view is considered important for autocomplete. Empty setImportantForContentCapture (int mode) Sets the mode to determine whether this view is considered important to the content. Empty setKeepScreenOn (boolean keepScreenOn) Controls whether the screen should remain by modifying the value of KEEP_SCREEN_ON. void setKeyboardNavigationCluster(boolean isCluster) Set whether this view is a root of a keyboard navigation cluster. empty setLabelFor (int id) Sets the id of a for which this vision serves as a label for accessibility purposes. Empty setLayerPaint updates the Paint object used with the current layer (used only if the current layer type is not set to LAYER_TYPE_NONE). Empty setLayerType (intType layer, ink paint paint) specifies the type of layer that supports this visualization. empty setLayoutDirection (layout intDirection) Set the direction of the layout for this view. empty setLayoutParams (ViewGroup.LayoutParams params) Set the layout parameters associated with this view. Final set of emptyLetrogar (left) Sets the left position of this view relative to its parent. Final set of emptyLeftTopRightBottom (int left, int top, int right, int bottom) Assign a size and position to this view. empty setLongClickable (boolean longClickable) allows or disables long-click events for this view. empty setMinimumHeight (int minHeight) Sets the minimum view height. empty setMinimumWidth (int minWidth) Sets the minimum width of the view. empty setNestedScrollingEnabled (boolean enabled) Enable or disable nested scrolling for this view. empty definedExtClusterForwardId (int nextClusterForwardId) Sets the view id to use as the root of the next keyboard navigation cluster. empty setExtFocusDownId (int nextFocusDownId) Sets the view id to use when the next focus is FOCUS_DOWN. empty setExtFocusForwardId (int nextFocusForwardId) Sets the view id to use when the next focus is FOCUS_FORWARD. Empty setExtFocusLeftId (int nextFocusLeftId) Sets the view id to use when the next focus is FOCUS_LEFT. empty setExtFocusRightId (int nextFocusRightId) Sets the view id to use when the next focus is FOCUS_RIGHT. empty setExtFocusUpId (int nextFocusUpId) Sets the view id to use when the next focus is FOCUS_UP. empty setOnApplyWindowInsetsListener (View.OnApplyWindowInsetsListener) Set an OnApplyWindowInsetsListener to assume the policy of applying window insets to this view. Empty setOnCapturedPointerListener (View.OnCapturedPointerListener l) Set a listener to receive callbacks when the pointer capture state of a view changes. empty setOnClickListener (View.OnClickListener l) Record a callback to be invoked when that view is clicked. empty setOnContextClickListener (View.OnContextClickListener l) Register a callback to be invoked when that view is clicked. Empty 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 an event listener call object drag to this View. void setOnFocusChangeListener (View.OnFocusChangeListener l) Register a callback to be invoked when the focus of this view has changed. void setOnGenericMotionListener (View.OnGenericMotionListener l) Register a callback to be invoked when a generic motion event is sent to this view. Empty setOnHoverListener (View.OnHoverListener l) Register a callback to be invoked when a hover event is sent to this view. Empty Empty l) Record a callback to be invoked when a hardware key is pressed in this view. Empty setOnLongClickListener (View.OnLongClickListener l) Register a callback to be invoked when this view is clicked and performed. empty setOnScrollChangeListener (View.OnScrollChangeListener l) Record a callback to be invoked when the x or y parchment positions of this view change. empty setOnSystemUiVisibilityChangeListener (View.OnSystemUiVisibilityChangeListener l) This method has been deprecated at API level 30. Use WindowInsets#isVisible(int) to find out about system bar visibility by setting an OnApplyWindowInsetsListener in this view. void setOnTouchListener (View.OnTouchListener l) Record a callback to be invoked when a touch event is sent to this view. Empty setOutlineAmbientShadowColor (color int) Sets the color of the ambient shadow that is drawn when the visualization has a Z value or positive elevation. Empty setOutlineProvider (ViewOutlineProvider provider) Defines the ViewOutlineProvider of the view, which generates the Outline that defines the shape of the shadow that casts and allows the outline clipping. Empty setOutlineSpotShadowColor (color int) Sets the spot shadow color that is drawn when the visualization has a Z value or positive elevation. empty setDeScrollMode (int overScrollMode) Set the over-scroll mode for this view. empty setPadding (int left, int top, int right, int bottom) sets the padding. empty setPaddingRelative (int start, int top, int end, int bottom) sets the relative fill. Empty setPivotX (pivotX float disk) Sets the x location of the point around which the view is rotated and scaled. Empty setPivotY (pivoty floating disk) Sets the y location of the point around which the view is rotated and scaled. Empty setSatoPointerIcon (PointerIcon pointerIcon) Set the pointer icon to the current view. Empty setPressed (boolean pressed) Sets the pressed state for this view. Final set of emptyRevealOnFocusHint (boolean revealOnFocus) Sets the preference of this view to reveal the behavior when it gains focus. Final set of emptyStetola (int on the right) Sets the right position of this view relative to its parent. Empty set Rotation (floating rotation) Defines the degrees that the visualization is rotated around the pivot point. Empty setRotationX (floating rotationX) Defines the degrees that the view is rotated around the horizontal axis through the pivot point. Empty setRotationY (floating rotationY) Defines the degrees that the view is rotated around the vertical axis through the pivot point. void setSaveEnabled (boolean enabled) Controls whether the state savings of this view is enabled (that is, whether your method onSaveInstanceState() will be called. Controls whether the entire hierarchy under this view will save its state when a state save traverse occurs from its parent. Empty setScaleX (floating scaleX) Defines the amount that the view is scaled in x around the pivot point, as a proportion of the non-escalating width of the view. Empty setScaleY (floating scaleY) Defines the amount that the visualization is sized in Y around the pivot pivot as a proportion of the non-escalating width of the view. Empty setScreenReaderFocusable (booleanaReaderFocusable screen) Defines whether this display should be a focusable element for screen readers and include non-focal views of your subtree when providing feedback. empty setScrollBarDefaultDelayBeforeFade (int scrollBarDefaultDelayBeforeFade) Set the delay before scroll bars disappear. empty setScrollBarFadeDuration (int scrollBarFadeDuration) Set the duration of scrollbar fading. empty setScrollBarSize(int scrollBarSize) Set the size of the scroll bar. empty setScrollBarStyle (int style) Specify the style of scroll bars. Empty setScrollContainer (boolean isScrollContainer) Change if this view is one of the sets of scrollable containers in your window. Empty setScrollIndicators (indicators int, mask of int) Sets the state of the scroll indicators specified by the mask. Empty definedScrollIndicators (indicators int) Sets the state of all parchment indicators. empty setScrollX(int value) Set the scrolled horizontal position of your view. empty setScrollY (int value) Set the vertical position rolled from your view. empty setScrollbarFadingEnabled (boolean fadeScrollbars) Set whether scroll bars will disappear when the display is not rolling. Selected empty set (selected Booleans) Changes the selection state of this view. Empty setSoundEffectsEnabled (booleanoEffectsEnabled sound) Set whether this view should have sound effects enabled for events such as click and tap. empty definedStateDescription (CharSequence stateDescription) Sets the description of the state of the view. empty setStateListAnimator (StateListAnimator stateListAnimator) attaches the StateListAnimator provided to this view. empty definedSStureExclusionRects(List<Rect> rects) Defines a list of areas within the post-layout coordinate space of this visualization where the system should not intercept touch or other pointing device gestures. SystemUiVisibility flags are depreced. Use windowinsetscontroller instead. Empty setStag (int key, object tag) sets a tag associated with this view and a key. Empty setStag (Object tag) sets the tag associated with this view. empty definedTextAlignment (int textAlignment) Set the alignment of the text. empty setTextDirection (text intDirection) Set the direction of the text. empty setTooltipText(CharSequence tooltipText) Sets the tooltip text that will be displayed in a small pop-up next to the view. SToto (top int) Final set sets the top position of this view relative to its parent. Empty setTouchDelegate (Delegate TouchDelegate) sets the TouchDelegate to this view. empty setTransitionAlpha (float alpha) This property is intended for use only by the Fade transition, which animates it produce a visual translucency that does not effect side effect (or is affected by) the actual alpha property. Final set of emptyTransitionName (String transitionIn the name) defines the name of the view to use to identify visualizations in transitions. empty setTransitionVisibility (visibility int) Changes the </Rect> </Rect> view without triggering any other changes. Empty setTranslationX (floating translationX) Sets the horizontal location of this view relative to the left position. Empty setTranslationY (floating translationY) Sets the vertical location of this visualization relative to its top position. void setTranslationZ (floating translationZ) Sets the depth location of this visualization relative to its elevation. Empty setVerticalFadingEdgeEnabled (vertical booleanaFadingEdgeEnabled) Set whether vertical edges should be faded when this view is scrolled vertically. VerticalScrollBarEnabled (vertical booleanACrollBarEnabled) Set whether the vertical scroll bar should be drawn or not. void setVerticalScrollbarPosition (position int) Set the position of the vertical scroll bar. empty setVerticalScrollbarThumbDrawable (Drawable drawable) Sets the drawable vertical scrollbar empty setVerticalScrollbarTrackDrawable (Drawable drawable) Sets the drawable drawable set (visibility int) vertical scrollbar empty Set the visibility state of this view. empty setWillNotCacheDrawing (boolean willNotCacheDrawing) This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. empty set WillNotDraw(boolean willNotDraw) If this view does not draw on its own, set this flag to allow for further optimizations. empty definedWindowInsetsAnimationCallback(WindowInsetsAnimation.Callback callback) Sets a windowInsetsAnimation.Callback to be notified about window animations that cause insets. empty setX (float x) Sets the visual position x of this view, in pixels. empty setY (float y) Sets the visual position y of this view, in pixels. Empty setZ (float z) Sets the visual position z of this view, in pixels. boolean showContextMenu() Shows the for this vision. boolean showContextMenu (float x, float y) Shows the context menu for this view anchored to the specified relative view coordinate. ActionMode startActionMode (ActionMode.Callback callback, type int) Start an action mode with the given type. ActionMode startActionMode (ActionMode.Callback callback) callback) an action mode with the standard type ActionMode#TYPE_PRIMARY. beginning of the emptyAnimation (animation animation) Start the specified animation now. early boolean finalDrag (data clipData, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) This method was deprecated at api level 24. Use startDragAndDrop for newer versions of the platform. final boolean startDragAndDrop (clipData data, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) Initiates a drag-and-drop operation. Boolean startNestedScroll (int axes) Start a nesteble parchment operation along the given axes. Embed ThisdScroll() Stop a nested parchment in progress. String toString() returns a sequence representation of the object. Matrix matrix transform modifies the input matrix in such a way that it maps local display coordinates to coordinates on the screen. Matrix matrix transform modifies the input matrix in such a way that it maps coordinates on the screen to view local coordinates. uncheck Drawable (Drawable who, Runnable what) cancels a scheduled action on a drawable. unselect Drawable (Drawable who) clear any events associated with the given drawable. Last empty updateDragShadow (View.DragShadowBuilder shadowBuilder) Updates the drag shadow for drag and drop operation in progress. boolean willNotCacheDrawing() This method has been depreed at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. boolean willNotDraw() Returns whether or not this View is based on its own. boolean awakenScrollBars (int startDelay, boolean invalidate) Trigger scroll bars to draw. boolean awakenScrollBars (int startDelay) Trigger scroll bars to draw. boolean awakenScrollBars() Trigger scroll bars to draw. int computeHorizontalExtentScroll() Calculate horizontal extension of the thumb of the horizontal scroll bar within the horizontal range. int computeHorizontalScrollOffset() Calculate the horizontal thumb offset of the horizontal scroll bar within the horizontal range. int computeHorizontalScrollRange() Calculate the horizontal range that the horizontal scroll bar horizontal int computeVerticalExtentScroll() Calculate the vertical thumb extension of the vertical scroll bar within the vertical range. int computeVerticalScrollOffset() Calculate the vertical thumb offset of the vertical scroll bar within the horizontal range. int computeVerticalScrollRange() Calculate the vertical range that the vertical scroll bar represents. Empty dispatchDraw (canvas screen) Called by draw to draw the child's views. boolean dispatchGenericFocusedEvent (motionEvent event) Dispatch a generic motion event to the currently focused view. dispatch booleangenericpointerevent (MotionEvent event) Send a generic motion event to the view under the first pointer. dispatch booleanhoverevent (MotionEvent event) dispatch a pair event. void dispatchRestoreState<Parcelable> (sparseArray container) called by restoreHierarchyState (android.util.SparseArray) to retrieve the state for this view and its children.<Parcelable> Empty DispatchSetActivated (boolean activated) Dispatch Set Enabled for all children of this view. empty dispatchSetPressed (boolean pressed) Dispatch setPressed for all children of this View. empty dispatchVisibilitySemtola (See changedVisver, visibility int) Dispatches a view visibility change in 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(Insets rect) This method has been deprethetised at API level 20. Starting with API 20 use dispatchApplyWindowInsets (android.view.WindowInsets) to apply insets to visualizations. Visualizations should replace onApplyWindowInsets (android.view.WindowInsets) or use the setOnApplyWindowInsetsListener (android.view.View.View.OnApplyWindowInsetsListener) to implement handling of your own insets. float getBottomFadingEdgeStrength() Returns the strength, or intensity, of the faded bottom edge. int getBottomPaddingOffset() Amount by which to extend the lower fading region. ContextMenu.ContextMenuInfo getContextMenuInfo() Visualizations should implement this if they have extra information to associate with the context menu. int getHorizontalScrollbarHeight() Returns the height of the horizontal scroll bar. float getLeftFadingEdgeStrength() Returns the strength, or intensity, of the left faded edge. int getLeftPaddingOffset() Quantity by which to extend the fading region to the left. 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 visualization should use. int getSuggestedMinimumWidth() returns the suggested minimum width that the view should use. Float </Parcelable> </Parcelable> the strength, or intensity, of the faded upper edge. int getTopPaddingOffset() Amount by which to extend the upper fading region. int getWindowAttachCount() boolean isPaddingOffsetRequired() If the View draws content within its fill and allows faded edges, it needs to support fill offsets. int static[] mergeDDEs (int[] baseState, int[] additionalState) Merges their own state values into additionalState in the base state of basestate values that were returned by onCreateDrawableState (int). empty onAnimationEnd() Invoked by a parent ViewGroup to notify the end of the animation currently associated with that view. empty onAnimationStart() Invoked by a parent ViewGroup to notify the start of the animation currently associated with that view. empty onAttachedToWindow() This is called when the view is attached to a window. empty onConfigurationChanged (New Config Configuration) called when the current configuration of the resources being used by the application have changed. empty in the CreateContextMenu (ContextMenu menu) Visualizations should implement this if the view itself adds items to the context menu. int[] onCreateDrawableState (int extraSpace) empty in DetachedFromWindow() This is called when the view is separated from a window. empty onDisplayHint (hint int) Gives this view a hint about whether it is displayed or not. void onDraw (Screen Screen) Implement this to make your drawing. Final empty in DrawScrollBars (canvas screen) Request the drawing of the horizontal and vertical scroll bar. empty onFinishInflate() End inflating an XML view. empty onFocusChanged (boolean gainFoba, direction int, Rect formerlyFocusedRect) Called by the display system when the focus state of this visualization changes. empty onLayout (boolean changed, int left, int top, int right, bottom int) Called layout when this view should assign a size and position to each of its children. empty in Measure (intMeasureSpec width, int heightMeasureSpec) Measure the visualization and its contents to determine the measured width and measured height. empty onOverScrolled (int scrollX, int scrollY, boolean clampedX, boolean clampedY) called by overScrollBy (int, int, int, int, int, int, int, boolean) to respond to the results of an over-scrolloperation. void onInstanceState (parcelable state) hook allowing a view to reaplua 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. empty onScrollChanged (int l, int t, int oldl, int oldt) That's called in response to an internal scroll in this view (i.e. the view went through your content). boolean onSetAlpha (int alpha) Invoked if there is a Transformation that involves alpha. empty onSizeChanged (int w, int h, int oldw, int oldh) This is called during the layout when the size of this view has has empty onVisibilityChanged (View changedView, int visibility) called when view visibility or a view ancestor has changed. empty in WindowVisibilityChanged (visibility int) called when the window it contains has changed 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 display with default behavior to scroll beyond normal content limits. Final set of empty MesuredDimension (int measuredWidth, int measuredHeight) This method must be called by onMeasure (int, int) to store the measured width and measured height. Boolean Drawable (Drawable who) check If your display subclass is displaying its own Drawable objects, it must override this function and return true to any Drawable that is displaying. From class java.lang.Object object() creates and returns a copy of this object. boolean equals (Object obj) Indicates if any other object is the same as this. Finalized empty() Called by the garbage collector on an object when garbage collection determines that there are no more references to the object. final class<?> getClass() Returns the runtime class of this object. int hashCode() returns a hash code value for the object. final void warning() Wakes up a single thread that is waiting on the monitor of this object. Final warning of the voidThe() Wakes up all the threads that are waiting on the monitor of this object. String toString() returns a sequence representation of the object. the final wait of the void (long timeout, int nanos) causes the current thread to wait until another thread invokes the notify() method or notifyAll() method for this object, or some other thread interrupts the current segment or a certain amount of real time has elapsed. the final empty wait (long time) causes the current thread to wait until another thread invokes the notify() method or notifyAll() method for this object, or a specified amount of time has elapsed. The final wait for the void() causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object. From the interface android.view.keyEvent.Callback boolean summary onKeyDown (int keyCode, keyEvent event) called when a key down event occurred. abstract boolean noKeyLongPress (int keyCode, keyEvent event) Called when a long press occurred. Abstract boolean noKeyMultiple (int keyCode, int count, KeyEvent event) Called when a user's interaction with an analog control, such as flinging a trackball, generates simulated events down/up to the same key multiple times in rapid succession. Abstract vulture in KeyUp (int keyCode, KeyEvent event) called when a key up event occurred. Whether or not this vision is a course for accessibility purposes. Can be a value as true or false. Related methods: setAccessibilityHeading (boolean) android:accessibilityLiveRegion Indica Indica services if the user should be notified when this view changes. It can be an integer value, like 100. Must be one of the following constant values. Assertive ConstantValueDescription2 Accessibility services must stop the ongoing speech to immediately announce changes to this view. accessibility services should not announce changes to this view. accessibility services polite1 should announce changes to this view. Related methods: setAccessibilityLiveRegion(int) android:accessibilityPaneTitle The title that this view should display to accessibility as a panel title. See View.setAccessibilityPaneTitle (CharSequence) It can be a string value, using '\\;' to escape characters such as '\' or '\\uxxxx' for a unicode character; Related methods: setAccessibilityPaneTitle (CharSequence) android:accessibilityTraversalAfter Sets the id of a visualization after which it is visited on accessibility traversal. A screen reader should visit the contents of the other view before the content of the view. It can be an integer value, like 100. Related methods: setAcessibilityTraversalAfter(int) android:accessibilityTraversalBefore Sets the id of a visualization before which it is visited in accessibility traverse. A screen reader should visit the contents of this view before the content of the preceding one. It can be an integer value, like 100. Related methods: setAccessibilityTraversalFore (int) alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque). It can be a floating-point value, such as 1.2. Related methods: android:autofillHints Describes the content of a view so that an autocomplete service can fill in the appropriate data. Multiple hints can be combined into a separate list of cígula or in a series of strings to mean, for example, emailAddress or postalAddress. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. It can be a

string value, using '\\;' to escape characters such as '\' or '\\uxxxx' for a unicode character; Related methods: android:autofilledHighlight Drawable to be drawn on the view to mark it as automatically populated Can be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: android:background A drawable to use as the background. This can be a reference to a fully drawable feature (such as a PNG image, 9-patch, XML state list description, etc.), or a solid color such as #ff000000 (black). Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. It can be a color value, in the form of #rgb, #argb, #rrggbb or #aarrggbb. Methods setBackgroundResource (int) android:backgroundTint Tint to apply to the background. It can be a color value, in the form of #rgb, #argb, #rrggbb or #aarrggbb. Related methods: setBackgroundTintList (ColorStateList) (ColorStateList) Blending mode used to apply the background tint. Must be one of the following constant values. ConstantValueDescription adds10Combines the color and alpha channels of tint and drawing, pinning the result to valid color values. Saturated (S + D) multiplies the color and alpha channels of the attractive with those of the tint. [Sa*Da, Sc*Dc] screenf[Sa+ Da - Sa*Da, Sc+Dc - Sc*Dc] src_atop9The hue is drawn above the drawable, but with the drawable alpha channel masking the result. [Da, Sc* Da + (1 - Sa) * Dc] src_in5The hue is masked by the drawable alpha channel. The colored channels are thrown away. [Sa*Da, Sc*Da] src_over3The tint is drawn on top of the drawable. [Sa + (1 - Sa)*Da, Rc = Sc + (1 - Sa)*Dc] Related methods: setBackgroundTintMode(PorterDuff.Mode) android:clickable Sets whether this view reacts to click events. It can be a Boolean value, such as true or false. Related methods: android:contentDescription Defines text that briefly describes the contents of the view. This property is primarily used for accessibility. Because some viewpoints do not have textual representation, this attribute can be used to provide such. It can be a string value, using '\\;' to escape characters such as '\' or '\\uxxxx' for a unicode character; Related methods: setContentDescription (CharSequence) android:contextClickable Defines whether this view reacts to context click events. It can be a Boolean value, such as true or false. Related methods: setContextClickable (boolean) android:defaultFocusHighlightEnabled If this View should use a default focus highlight when it focuses, but has R.attr.state_focused set in its background. It can be a Boolean value, such as true or false. Related methods: setDefaultFocusHighlightEnabled (boolean) 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 automatic. Depreive: The display drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. Must be one of the following constant values. ConstantValueDescription auto0The framework decides what quality level should be used for the drawing cache. Depreciated: The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. high quality high. When set to high quality, the drawing cache uses a higher color depth but uses more memory. Depreciated: The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. low quality1. When set to low quality, the drawing cache uses a lower color depth, thus losing precision in rendering gradients, uses less memory. Depreciated: The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. Related methods: setDrawingCacheQuality(int) android:duplicateParentState android:duplicateParentState this attribute is set to true, the view gets its drawing state (focused, pressed, etc.) from its direct parent and not from itself. It can be a Boolean value, such as true or false. Android:Altitude base z depth of view. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: Defines whether to fade scroll bars when they are not in use. It can be a Boolean value, such as true or false. Related methods: setScrollbarFadingEnabled(boolean) android:fadingEdgeLength Defines the length of faded edges. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: getVerticalFadingEdgeLength() android:filterTouchesWhenObscured Specifies if you filter touches when the preview 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 window. See the View security documentation for more details. It can be a Boolean value, such as true or false. Related methods: setFilterTouchesWhenObscured (boolean) android:fitsSystemWindows Boolean internal attribute to adjust the display layout based on system windows, such as the status bar. If true, adjust the fill of this view to leave room for system windows. It only has effect if this view is in an unincorporated activity. It can be a Boolean value, such as true or false. Related methods: setFitsSystemWindows(boolean) android:focusable Controls if a view can have focus. By default, this is automatic that allows the framework to determine whether a user can move focus to a view. When you set this attribute to true, the view is allowed to have focus. When you set it to false the view will not focus. This value does not affect the behavior of linking directly to View.requestFocus(), which will always prompt focus regardless of that view. It only impacts where focus navigation will attempt to move focus. It can be a Boolean value, such as true or false. Must be one of the following constant values. ConstantValueDescription auto10 Related methods: android:focusableInTouchMode Boolean that controls whether a view can have focus during touch mode. If this is true for a view, this view can gain focus when clicked, and can maintain focus if another view is clicked that does not have this attribute set to true. It can be boolean value, such as true or false. Related methods: setFocusableInTouchMode(boolean) android:focusedByDefault If this view is a default focus view. Only one view per keyboard navigation cluster can have this attribute set to true. View View It can be a Boolean value, such as true or false. Related methods: setFocusedByDefault (boolean) android:forceHasOverlappingRendering If this view has elements that can overlap when drawn. See View.forceHasOverlappingRendering(Boolean). It can be a Boolean value, such as true or false. Related methods: forceHasOverlappingRendering(boolean) android:foreground Sets the drawable to draw on the content. This can be used as an overlay. The drawable foreground participates in completing the content if the severity is set to fill. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. It can be a color value, in the form of #rgb, #argb, #rrggbb or #aarrggbb. Related methods: android:foregroundGravity sets gravity to apply to the drawable foreground. The default severity to fill. Must be one or more (separated by '|') of the following constant values. ConstantValueDescription bottom50Push object to the bottom of its container, not changing its size. center11The object in the center of its container on the vertical and horizontal axis, not changing its size. center_horizontal1Place object in the horizontal center of its container, not changing its size. center_vertical10Place object in the vertical center of your container, not changing its size. clip_horizontal8Additional option that can be set to have the child's left and/or right edges cut off at the boundaries of the container. The clip will be based on horizontal gravity: a left gravity will cut the right edge, a right gravity will cut the left edge, and neither of them will cut both edges. clip_vertical80Additional option that can be set to have the top and/or bottom edges of the child cut off at the boundaries of your container. The clip will be based on vertical gravity: a higher gravity will cut the lower edge, a lower gravity will cut the top edge, and neither of them will cut both edges. enchi77Srow the horizontal and vertical size of the object, if necessary, so that it completely fills its container. fill_horizontal7Grow the horizontal size of the object, if necessary, so that it completely fills its container. fill_vertical70Grow the vertical size of the object, if necessary, so that it completely fills its container. left3Push object to the left of your container, not changing its size. right5Push object to the right of your container, not changing its size. Top30Push object to the top of your container, not changing its size. Related methods: setForegroundGravity(int) android:foregroundTint Tint to apply to the foreground. It can be a color value, in the form of #rgb, #argb, #rrggbb or #aarrggbb. Related methods: setForegroundTintList (ColorStateList) android:foregroundIntMode Blending mode used to apply tonality in Plan. Must be one of the following constant values. ConstantValueDescription adds10Combines the color and alpha channels of tint and drawing, pinning the result to valid color values. Saturated (S + D) D) the color and alpha channels of the drawable with those of the tint. [Sa*Da, Sc*Dc] screenf[Sa+ Da - Sa*Da, Sc+Dc - Sc*Dc] src_atop9The hue is drawn above the drawable, but with the drawable alpha channel masking the result. [Da, Sc* Da + (1 - Sa) * Dc] src_in5The hue is masked by the drawable alpha channel. The colored channels are thrown away. [Sa*Da, Sc*Da] src_over3The tint is drawn on top of the drawable. [Sa + (1 - Sa)*Da, Rc = Sc + (1 - Sa)*Dc] Related methods: setForegroundTintMode(PorterDuff.Mode) android:hapticFeedbackAndnabled Boolean that controls whether a view should have haptic feedback enabled for events such as long presses. It can be a Boolean value, such as true or false. Related methods: setHapticFeedbackEnabled(boolean) android:id Provide an identifier name for this view, to retrieve it later with View.findViewById() or Activity.findViewById(). This should be a resource reference; you typically define this by using the @+ syntax to create a new ID feature. For example: android:id=@+id/my_id which allows you to retrieve the view later with findViewById (R.id.my_id). Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: android:importantForAccessibility Describes whether or not this view is important for accessibility. If important, the preview triggers accessibility events and is reported to accessibility services that query the screen. Note: Although not recommended, an accessibility service may decide to ignore this attribute and operate on all viewpoints of the preview tree. It can be an integer value, like 100. Must be one of the following constant values. ConstantValueDescription auto0The system determines whether the display is important for accessibility - standard (recommended). no2 Vision is not important for accessibility. inHideDescendants4Vision is not important for accessibility, nor any of your descending opinions. yes1Vision is important for accessibility. Related methods: setImportantForAccessibility(int) android:importantForAutofill Android System Tips if the display node associated with this View should be included in a display structure used for autocomplete purposes. Must be one or more (separated by '|') of the following constant values. ConstantValueDescription auto0Read the Android System uses its heuristics to determine if visualization is important for autocomplete. no2Hint the Android System that this view is *not* important for autocomplete, but your children (if any) will be traversed.. in ExcludeDescendants8Hint the Android System that this view is *not* important for autocomplete, and your children (if any) will not be traversed. yes1Hint the Android System that this view is important for filling and their children (if any) will be crossed.. simExcludeDescendants4Hint the Android System that this view is important for autocomplete, but your children (if any) will not be crossed. Related methods: related: android:importantForContentCapture suggests to the Android System whether the display node associated with this View should be used for content capture purposes. Must be one or more (separated by '|') of the following constant values. ConstantValueDescription auto0Let the Android System uses its heuristics to determine whether the display is important for capturing content. no2Hint the Android System that this view is *not* important for capturing content, but your children (if any) will be traversed.. in ExcludeDescendants8Hint the Android System that this view is *not* important for capturing content, and your children (if any) will not be traversed. yes1Hint the Android System that this view is important for capturing content, and your children (if any) will be traversed.. simExcludeDescendants4Hint the Android System that this view is important for capturing content, but your children (if any) will not be traversed. Related methods: setImportantForContentCapture(int) Set this if the view serves as a scroll container, which means that it can be resized to shrink its general window so that there is space for an input method. If not set, the default value is true if scrollbars have the vertical set of the scrollbar, something else is false. It can be a Boolean value, such as true or false. Related methods: setScrollContainer(boolean) android:keepScreenOn Controls if the display window should keep the screen on while visible. It can be a Boolean value, such as true or false. Related methods: android:keyboardNavigationCluster If this view is a root of a keyboard navigation cluster. See View.setKeyboardNavigationCluster(Boolean). It can be a Boolean value, such as true or false. Related methods: setKeyboardNavigationCluster (boolean) android:layerType Specifies the type of layer backing this view. The default value is none. See View.setLayerType (int, android.graphics.Paint) for more information. Must be one of the following constant values. Related methods: android:layoutDirection defines the direction of the layout drawing. This is typically associated with the writing direction of the language script used. Possible values are ltr to left to right, rtl to right to left, locale, and inherit from parent's view. If there is nothing to inherit, location is used. Locale back to en-US. ltr is the direction used in en-US. The default for this attribute is to inherit. Must be one of the following constant values. ConstantValueDescription inheritsfrom the parent. locallol. Ltr0Le-to-Right. RTL1 Right to Left. Related methods: android:longClickable Defines whether this view reacts to long-click events. It can be a Boolean value, such as true or false. Related methods: setLongClickable (boolean) android:minHeight Sets the minimum view height. It is not guaranteed that the vision able to reach this minimum height (for example, if its parent layout restricts it with less height available). It can be a dimension value, which is a one number of points attached with a unit as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: android:minWidth sets the minimum view width. It is not guaranteed that the visualization will be able to achieve this minimum width (for example, if its parent layout restricts it with less available width). It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: android:nextClusterForward sets the next keyboard navigation cluster. If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: setNextClusterForwardId(int) android:nextFocusDown Sets the next view to focus when the next focus is View.FOCUS_DOWN If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: android:nextFocusForward Sets the next view to focus on when the next focus is View.FOCUS_FORWARD If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: setNextFocusForwardId(int) android:nextFocusLeft Sets the next view to focus on when the next focus is View.FOCUS_LEFT. If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: android:nextFocusRight Sets the next view to focus on when the next focus is View.FOCUS_RIGHT If the reference refers to a view that does not exist or is part of an invisible hierarchy, a RuntimeException will result when the reference is accessed. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: android:nextFocusUp Sets the next view to focus on the next focus is View.FOCUS_UP If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, the will result when the reference is accessed. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: android:onClick Method name in the context of this view to invoke when the view is clicked. This name must match a public method that takes exactly one parameter of the view type. For example, if you specify android:onClick=sayHello, you must declare a public open sayHello (View v) method of your context (typically your Activity). It can be a string value, using '\\;' to escape characters such as '\' or '\\uxxxx' for a unicode character; android:outlineAmbientShadowColor sets the color of the ambient shadow that is drawn when the view has a Positive Z or Elevation value. By default, the shadow color is black. Generally, this color will be opaque, so the shadow intensity is consistent between different views with different colors. The opacity of the final ambient shadow is a function of the height of the shadow cast, the alpha channel of the AmbientShadowColor (typically opaque) contour, and the Theme attribute R.attr.ambientShadowAlpha. It can be a color value, in the form of #rgb, #argb, #rrggbb or #aarrggbb. Related methods: setOutlineAmbientShadowColor(int) android:outlineSpotShadowColor Sets the spot shadow color that is drawn when the display has a Z value or positive elevation. By default, the shadow color is black. Generally, this color will be opaque, so the shadow intensity is consistent between different views with different colors. The opacity of the endpoint shadow is a function of the height of the shadow castor, the alpha channel of the ShadowColor (typically opaque) contour, and the Theme attribute R.attr.spotShadowAlpha. It can be a color value, in the form of #rgb, #argb, #rrggbb or #aarrggbb. Related methods: setOutlineSpotShadowColor(int) android:padding Sets the fill, in pixels, of the four borders. Padding is set to space between the edges of the view and the contents of the view. This value takes precedence over any of the border-specific values (FillLeft, Top padding, Straight fill, FillBottom, Horizontal fill, and Vertical fill), but will not replace theStart fill or end fill, if defined. The size of a view will include its padding. If an R.attr.background is provided, the fill is initially set for this (0 if the drawable has no fill). Explicitly setting a fill value will override the corresponding fill found in the background. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: (int,int,int,int) Sets the fill, in pixels, of the bottom edge; see R.attr.fill. Can be a dimension value, which is a floating-point number attached with a unit as as Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: setPaddingRelative (int,int,int,int)android:fillEnd Sets the fill, in pixels, of the final edge; see R.attr.fill. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: setPaddingRelative (int,int,int,int) android:fillHorizontal Sets the padding, in pixels, of the left and right edges; see R.attr.fill. This value will take precedence over the FillLeft and FillRe, but not fillStart or FillEnd (if set). It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). android:PaddingLeft sets the fill, in pixels, of the left edge; see R.attr.fill. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: setPadding (int,int,int,int)android:fillS the fill sets, in pixels, from the right edge; see R.attr.fill. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: setPadding (int,int,int,int)android:fillStart Sets the fill, in pixels, of the initial border; see R.attr.fill. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: setPaddingRelative (int,int,int,int) android:Top fill Sets the fill, in pixels, of the top edge; see R.attr.fill. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: (int,int,int,int)android:fillVertical Sets the fill, in pixels, from the top and bottom edges; see R.attr.fill. This value takes precedence over the Top fill and the Bottom fill, if set. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels scaled based on preferred font size), in and mm (millimeters). android:requerFadingEdge Defines which edges should be faded in scrolling. Must be one or more (separated by '|') of the following constant values. ConstantValueDescription horizontal1000Fades horizontal edges only. no border is faded. vertical edges2000Fades only. Related methods: setVerticalFadingEdgeEnabled(boolean) android:rotation of the view rotation, in degrees. It can be a floating-point value, such as 1.2. Related methods: android:rotationX rotation of the view around the x-axis, in degrees. It can be a floating-point value, such as 1.2. Related methods: android:rotation rotation of the view around the y-axis, in degrees. It can be a floating-point value, such as 1.2. Related methods: android:saveEnabled If false, no state will be saved for this view when being frozen. The default is true, allowing the preview to be saved (however, it must also have an ID assigned to it for its state to be saved). Setting it to fake only disables the state for this view, not for your children who can still be saved. It can be a Boolean value, such as true or false. Related methods: android:scaleX scale of visualization in x direction. It can be a floating-point value, such as 1.2. Related methods: view android:scaleY scale in y direction. It can be a floating-point value, such as 1.2. Related methods: android:screenReaderFocusable if this view should be treated as a focusable unit by screen reader accessibility tools. See View.setReaderFocusable (Boolean). The default, false value lets the screen reader consider other signals, such as focusing or the presence of text, to decide what it focuses on. It can be a Boolean value, such as true or false. Related methods: setScreenReaderFocusable(boolean) Defines which scroll indicators should be displayed when the display can be scrolled. Multiple values can be combined using logical OR, for example top|bottom. Must be one or more (separated by '|') of the following constant values. ConstantValueDescription bottom2Displays bottom scrollindicator when vew can be scrolled down. End20Displays right parchment indicator when vew can be rolled in the final direction. Left4Displays left parchment indicator when vew can be rolled to the left. no scroll indicators are displayed. Right8Displays right scrolling indicator when vew can be scrolled right. Start 10Displays right parchment indicator when vew can be rolled in the initial direction. Top1Displays top scrolling indicator when the view can be scrolled up. Related methods: The initial horizontal offset, in pixels. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. The available units are: px (pixels), dp (pixels independent of sp (pixels sized based on preferred font size), in (inches) and mm (millimeters). The initial vertical scroll offset, in pixels. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available Available are: px (pixels), dp (density-independent pixels), sp (pixels scaled based on preferred font size), in (inches), and mm (millimeters). Defines whether the horizontal strip of the scroll bar should always be drawn. It can be a Boolean value, such as true or false. Defines whether the vertical scroll strip should always be drawn. It can be a Boolean value, such as true or false. Sets the delay in milliseconds that a scrollbar waits before disappearing. It can be an integer value, like 100. Related methods: setScrollBarDefaultDelayBeforeFade(int) Sets the delay in milliseconds that a scroll bar takes to disappear. It can be an integer value, like 100. Related methods: setScrollBarFadeDuration(int) Defines the width of vertical scroll bars and the height of horizontal scroll bars. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: Controls the style and position of the scrollbar. Scroll bars can be overlapored or inset. When inset, they add to the view fill. And scroll bars can be drawn within the fill area or at the edge of the view. For example, if a view has a drawing background and you want to draw the scroll bars within the fill specified by the drawable, you can use insideOverlay or insideInset. If you want them to appear on the edge of the view, ignoring the padding, then you can use outside of Overlay or outside of Inset. Must be one of the following constant values. ConstantValueDescription insideInset1000000Inside the padding and inset. insideOverlay0Inside the fill and overlap. offInset3000000Edge of vision and inset. overlay200000Edge of view and overlaid. Related methods: Sets the thumb of the drawn horizontal scroll bar. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: setHorizontalScrollbarThumbDrawable(Drawable) Sets the thumb of the drawn vertical scroll bar. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: setVerticalScrollbarThumbDrawable(Drawable) Defines the drawn horizontal scroll strip. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related methods: setHorizontalScrollbarTrackDrawable(Drawable) Sets the drawable vertical scroll strip. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. Related: setVerticalScrollbarTrackDrawable(Drawable) Defines which scroll bars should be displayed in scrolling or not. Must be one or more (separated by '|') of the following constant values. ConstantDescription of Values only the horizontal scroll bar. no scroll bar is displayed. Vertical200 displays vertical scrollbar only. android:soundEffectsEnabled Boolean that controls whether a display should have sound effects enabled for events like click and tap. It can be a Boolean value, such as true or false. Related methods: setSoundEffectsEnabled (boolean) android:stateListAnimator Sets the state-based animator to the View. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. android:tag Provide a tag for this view containing a String, to be retrieved later with View.getTag() or searched with View.findViewWithTag(). It is generally preferable to use IDs (via the android:id attribute) instead of tags because they are faster and allow verification of the build-time type. It can be a string value, using '\\;' to escape characters such as '\' or '\\uxxxx' for a unicode character; android:textAlignment Sets the alignment of the text. It can be an integer value, like 100. Must be one of the following constant values. ConstantValueDescription center4Cent paragraph, for example: ALIGN_CENTER. gravity1Default for root view. Gravity determines the alignment, ALIGN_NORMAL, ALIGN_CENTER, or ALIGN_OPPOSITE, which are relative to the text direction of each paragraph. inherited Default. textEnd3Align to the end of the paragraph, for example: ALIGN_OPPOSITE. textStart2Align to the beginning of the paragraph, for example: ALIGN_NORMAL. ViewEnd6Align to the end of the view, which is ALIGN_RIGHT if the layout resolved the visual?sDirection for LTR and ALIGN_LEFT otherwise. displayStart5Align to the beginning of the view, which is ALIGN_LEFT if the resolved layout of the visual?s is LTR and ALIGN_RIGHT otherwise. Related methods: android:textDirection Sets the direction of text. It can be an integer value, like 100. Must be one of the following constant values. ConstantValueDescription anyRtl2The direction of the paragraph is RTL if it contains any strong RTL characters, otherwise it is LTR if it contains some strong LTR character. If neither of the two is, the direction of the paragraph is the layout direction resolved by the visualization. firstStrong1Default for root view. The first strong directional character determines the direction of the paragraph. If there is no strong directional character, the direction of the paragraph is the layout direction resolved by the visualization. firstStrongLtr6The first strong directional character determines the direction of the paragraph. If there is no strong directional character, the direction of the paragraph is LTR. firstStrongRtl7The first strong directional character determines the direction of the paragraph. If there is no strong directional character, the direction of the paragraph is RTL. inherited Default. locale5The direction of the paragraph comes from the Locale system. LTR3The direction of the paragraph is left to right. rtl4The direction of the paragraph is from right to left. Methods android:theme specifies a replacement of the theme for a one When a theme override is set, the view is inflated using a context theme with the specified feature. During XML inflation, any childish view under the vision with an overlap theme will inherit the thematic context. Can it be a reference to another resource, in the form @[+][package:]type/name or a thematic attribute on the form ? [package:]type/name. android:tooltipText Sets text displayed in a small pop-up window on hover or long press. It can be a string value, using '\\;' to escape characters such as '\' or '\\uxxxx' for a unicode character; Related methods: setTooltipText(CharSequence) android:transformPivotX x pivot point location around which the view will rotate and scale. This xml attribute defines the pivotX property of the View. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: android:transformPivotY and location of the pivot point around which the view will rotate and scale. This xml attribute defines the pivot property of the view. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: android:transitionNamesa names a view so that it can be identified for transitions. The names must be unique in the view hierarchy. It can be a string value, using '\\;' to escape characters such as '\' or '\\uxxxx' for a unicode character; android:translationX in view x. This value is added after the layout to the left property of the view, which is set by its layout. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: android:translationY translation in y of the view. This value is added after the layout to the top view property, which is defined by its layout. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. Available units are: px (pixels), dp (density-independent pixels), sp (pixels sized based on preferred font size), in (inches), and mm (millimeter). Related methods: android:translationZ z translation of the display. This value is added to your elevation. It can be a dimension value, which is a floating-point number attached with a unit such as 14.5sp. The available units are: px (pixels), dp (pixels independent of sp (pixels sized based on preferred font size), in (inches) and mm (millimeters). Related methods: android:visibility Controls the initial visibility of the view. Must be one of the following constant values. ConstantDescription of Values hidden, as if the view had not been added. invisible1Not displayed, but taken into account during layout (space is left to it). visible visible on the screen; the default value. Related methods: Constants in the public static end region mode ACCESSIBILITY_LIVE_REGION_ASSERTIVE Live specifying that accessibility services must stop speech in progress to immediately announce changes in this view. Use with setAcessibilityLiveRegion(int). Constant value: 2 (0x00000002) public static end mode int ACCESSIBILITY_LIVE_REGION_NONE Living Region specifying that accessibility services should not automatically announce changes to this view. This is the default live region mode for most views. Use with setAcessibilityLiveRegion(int). Constant value: 0 (0x00000000) public static end mode int ACCESSIBILITY_LIVE_REGION_POLITE live region specifying that accessibility services should announce changes in this view. Use with setAcessibilityLiveRegion(int). Constant value: 1 (0x00000001) final static public String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE Tip indicating that this view can be automatically filled with a credit card expiration date. It should be used when the credit card expiration date is represented by only one view; if it is represented by more than one (for example, a view for the month and another view for the year), each of these views should use the unit-specific hint (AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DAY, AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH, or AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_YEAR). Can be used with setAutofillHints(java.lang.String[]) or android:autofillHint (in this case the value should be . AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE). When annotating a view with this tip, it is recommended to use a date autofill value to avoid ambiguity when the autocomplete service provides a value for it. To understand why a value can be ambiguous, consider April 2020, which could be represented as any of the following options: 04/2020 4/2020 2020/04 202 0/4 April/2020 Apr/2020 You set a date autocomplete value for the display, replacing the following methods: See setAutofillHints(java.lang.String) for more information on autofill tips. Constant value: End Static Public CreditCardExpirationDate String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH Hint indicating that this view can be automatically filled with a month of credit card expiration. Can be used with setAutofillHints(java.lang.String[]) or android:autofillHint (in this case the value should be . AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH). When annotating a view with this tip, it is recommended to use a text autofill value whose value is the numeric representation of the month, starting at 1 to ambiguity when the autocomplete service provides a value for it. To understand why a value might be ambiguous, consider January, which could be represented as any of 1: recommended form. 0: If you follow the Calendar#MONTH convention. January: full name, full name, English. jan: abbreviated name, in English. January: full name, in another language. Another recommended approach is to use a date autocomplete value - see AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE for more details. See setAutofillHints(java.lang.String) for more information on autocomplete tips. Constant value: Final static public creditCardExpirationMonth int AUTOFILL_TYPE_NONE autocomplete type for visualizations that cannot be filled automatically. Typically used when the view is read-only; for example, a text label. See also: Constant value: 0 (0x0000000) Added in API level 1 Depreed in API level 28 final public static int DRAWING_CACHE_QUALITY_AUTO This constant was depreed at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Allows automatic quality mode for the cache of the comatose. Constant value: 0 (0x00000000) Added in API level 1 Depreitated in API level 28 public static final int DRAWING_CACHE_QUALITY_HIGH This constant has been depreed at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have problems compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Allows high-quality mode for drawing caching. Constant value: 1048576 (0x00100000) Added in API level 1 Depreed in API level 28 int public static end This constant was deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Allows low-quality mode for drawing caching. Constant value: 524288 (0x00080000) final static public int FOCUS_BACKWARD Use with focoSearch(int). Move the focus to the previous selectable item. Constant value: 1 (0x000000001) final static public int FOCUS_FORWARD Use with focoSearch(int). Move the focus to the next selectable item. Constant value: 2 (0x000000002) final static public int FOCUS_LEFT Use with focoSearch(int). Move the focus to the left. Constant value: 17 (0x000000011) final static public int FOCUS_RIGHT Use with focoSearch(int). Move the focus to the right. Constant value: 66 (0x00000042) final static public int HAPTIC_FEEDBACK_ENABLED Display flag indicating whether this view should have plythfeedback enabled for events such as long presses. Constant value: 268435456 (0x100000000) public static end int IMPORTANT_FOR_ACCESSIBILITY_AUTO automatically determine whether a view is important for accessibility. Constant value: 0 (0x000000000) final static public int IMPORTANT_FOR_ACCESSIBILITY_NO The view is not important for accessibility. Constant value: 2 (0x00000002) final static public int IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS The vision is not important for accessibility, nor any of its descendant views. Constant value: 4 (0x00000004) final static public int IMPORTANT_FOR_ACCESSIBILITY_YES Vision is important for accessibility. Constant value: 1 (0x00000001) public static final int KEEP_SCREEN_ON Display flag indicating that the screen should remain on while the window containing this view is visible to the user. This effectively worries about automatically setting the WindowManager windowmanager.LayoutParams#FLAG_KEEP_SCREEN_ON. Constant value: 67108864 (0x040000000) LAYER_TYPE_HARDWARE end indicates the display has a hardware layer. A hardware layer is supported by a hardware-specific texture (usually Frame Buffer Objects or FBO in OpenGL hardware) and causes the display to render using Android Android hardware rendering but only if hardware acceleration is turned on to the display hierarchy. When hardware acceleration is turned off, hardware layers behave exactly like layers of software. A hardware layer is useful for applying a specific color filter and/or blending and/or translucency mode to a view and all its children. A hardware layer can be used to store a complex tree of view in a texture and reduce the complexity of drawing operations. For example, 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 rendering quality when rotation transformations are applied to a view. It can also be used to avoid possible clipping problems when applying 3D transformations to a view. Constant value: 2 (0x000000002) int final static public int LAYER_TYPE_SOFTWARE indicates that the view has a software layer. A software layer is supported by a bitmap and causes the preview to render using the Android software rendering pipeline, even if hardware acceleration is enabled. Software layers have several uses: When the application is not using hardware acceleration, a software layer is useful for applying a specific color filter and/or blending mode and/or translucency to a view and all of its children. When the application is using hardware acceleration, a software layer is useful for making drawing primitives unsupported by the accelerated hardware pipeline. It can also be used to store a complex tree view in a texture and reduce the complexity of drawing operations. For example, when animating a complex vision tree with a translation, a software layer can be used to render the view tree only once. Software layers should be avoided when affected tree verifies are updated frequently. Each update will require re rendering the software layer, which can be potentially slow (especially when hardware acceleration is turned on, since the layer will have to be loaded into a hardware texture after each update.) Constant value: 1 (0x00000001) the final public static int int LAYOUT_DIRECTION_INHERIT horizontal layout direction of this view is inherited from its parent. Use with setLayoutDirection(int). Constant value: 2 (0x00000002) int public static end LAYOUT_DIRECTION_LOCALE The horizontal layout direction of this view is deduced from the default language script for the location. Use with setLayoutDirection(int). Constant value: 3 (0x00000003) public static final int LAYOUT_DIRECTION_LTR Horizontal layout direction of this view is left to right. Use with setLayoutDirection(int). Constant value: 0 (0x00000000) final static public int Horizontal layout direction of this view is right to left. Use with setLayoutDirection(int). Constant value: 1 (0x00000001) int final static public int NO_ID used to mark a view that has no ID. Constant value: -1 (0xffffffffffff) int final static public int OVER_SCROLL_IF_CONTENT_SCROLLS Allow a user to scroll too much this view only if the content is large large to scroll significantly, as long as it's a view that can scroll. See also: getOverScrollMode()setOverScrollMode(int) Constant value: 1 (0x0000000001) int final static public int SCREEN_STATE_OFF Indicates that the screen has changed state and is now off. See also: onScreenStateChanged(int) Constant value: 0 (0x000000000) final static public int SCREEN_STATE_ON Indicates that the screen has changed state and is now turned on. See also: onScreenStateChanged(int) Constant value: 1 (0x000000001) final static public int SCROLLBARS_INSIDE_INSET The scrollbar style to display scrollbars within the padded area, increasing view padding. Scroll bars do not overlap the content area of the view. Constant value: 16777216 (0x01000000) int public static end SCROLLBARS_INSIDE_OVERLAY The scrollbar style to display scrollbars within the content area without increasing the fill. Scroll bars will be translucency overlaps in the contents of the view. Constant value: 0 (0x00000000) public static end input SCROLLBARS_OUTSIDE_INSET The scrollbar style to display scrollbars at the edge of the view, increasing view padding. Scroll bars only overlap the bottom, if any. Constant value: 50331648 (0x03000000) public static end input SCROLLBARS_OUTSIDE_OVERLAY The scrollbar style to display scrollbars at the edge of the view without increasing the padding. Scroll bars will be translucently overlaps. Constant value: 33554432 (0x02000000) int public static end SCROLLBAR_POSITION_DEFAULT Position the scroll bar in the default position as determined by the system. Constant value: 0 (0x00000000) public static end input int SCROLLBAR_POSITION_LEFT Position the scroll bar along the left edge. Constant value: 1 (0x00000001) public static end entry int SCROLLBAR_POSITION_RIGHT Position the scroll bar along the right edge. Constant value: 2 (0x000000002) int final static public int SCROLL_AXIS_HORIZONTAL indicates scrolling along the horizontal axis. Constant value: 1 (0x000000001) int final static public int SCROLL_AXIS_NONE indicates no axis of vision scrolling. Constant value: 0 (0x000000000) final static public int SCROLL_AXIS_VERTICAL Indicates scrolling along the vertical axis. Constant value: 2 (0x00000002) public static final int SOUND_EFFECTS_ENABLED Display flag indicating whether this view should have sound effects enabled for events such as click and tap. Constant value: 134217728 (0x08000000) Added at API level 16 Depreed in API level 30 public static final int SYSTEM_UI_FLAG_FULLSCREEN This constant has been depreed at API level 30. Use windowinsetsController#hide(int) with Type#statusBars() instead. Flag for setSystemUiVisibility(int): The preview has requested to enter normal fullscreen mode so that its contents can take over the screen and, at the allow the user to interact with the application. This has the same visual effect as WindowManager.LayoutParams.FLAG_FULLSCREEN, which means that non-critical screen decorations (such as (such as status bar) will be hidden while the user is in the View window, focusing the experience on that content. Unlike the window flag, if you are using the ActionBar in overlay mode with Window#FEATURE_ACTION_BAR_OVERLAY, then enabling this flag will also hide the action bar. This approach to going to the full screen is best used over the window flag when it is a transient state -- that is, the application does so at certain points in its interaction with the user where it wants to allow the user to focus on the content, but not as a continuous state. For situations where the app would simply like to stay full screen all the time (like a game that wants to take over the screen), the window flag 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 UI states. When using this flag, the application should provide some easy ease for the user to exit it. A common example would be in an e-book reader, where tapping the screen brings back any 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) Added in API level 14 Depreided api at API level 30 public static final int SYSTEM_UI_FLAG_HIDE_NAVIGATION This constant was depreed at API level 30. Use WindowInsetsController#hide(int) with Type#navigationBars() instead. Flag to set SystemUiVisibility(int): The view has requested that system navigation be temporarily hidden. This is an even less obtrusive state than what

asked for SYSTEM_UI_FLAG_LOW_PROFILE; on devices that draw essential navigation controls (Home, Back, and the like) on the SYSTEM_UI_FLAG_HIDE_NAVIGATION will cause them to disappear. This is useful (in conjunction with FLAG_FULLSCREEN and FLAG_LAYOUT_IN_SCREEN flags) to display content using each last pixel on the display. There is one limitation: Because navigation controls are so important, less user interaction will cause them to reappear immediately. When this happens, both this flag and SYSTEM_UI_FLAG_FULLSCREEN will be automatically cleared, so that both elements reappear at the same time. See also: setSystemUiVisibility(int) Constant value: 2 (0x00000002) Added to API level 19 Depredated api at API level 30 public static final int SYSTEM_UI_FLAG_IMMERSIVE_STICKY This constant was depreciated at API level 30. Use windowinsetscontroller#BEHAVIOR_SHOW_TRANSIENT_BARS_BY_SWIPE instead. Flag for setSystemUiVisibility(int): The view would like to remain interactive by hiding the status bar with SYSTEM_UI_FLAG_FULLSCREEN and/or hiding the navigation bar with SYSTEM_UI_FLAG_HIDE_NAVIGATION. Use this flag to create an experience while hiding the system bars. If this flag is not set, SYSTEM_UI_FLAG_HIDE_NAVIGATION form released by the system in any user interaction, and SYSTEM_UI_FLAG_FULLSCREEN is released by force by the if the user swipes from the top of the screen. When system bars are hidden in immersive mode, they can be temporarily revealed with system gestures, such as swiping from the top of the screen. These system transient bars will overrun application content, may have some degree of transparency, and will automatically hide after a short period of time. Since this flag is a modifier for SYSTEM_UI_FLAG_FULLSCREEN and SYSTEM_UI_FLAG_HIDE_NAVIGATION, it only has an effect when used in combination with one or both flags. Constant value: 4096 (0x00001000) Added to API level 16 Depreed in API level 30 public static final int SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN This constant has been depreed at API level 30. For floating windows, use LayoutParams#setFitInsetsTypes(int) with Type#statusBars() ()}. For non-floating windows that fill the screen, call window#setDecorFitsSystemWindows (boolean) with fake. Flag for setSystemUiVisibility(int): View would like your window to be set to if it had SYSTEM_UI_FLAG_FULLSCREEN, even if it was not. This allows you to avoid artifacts when entering and exiting this way, at the expense that part of your user interface can be covered by screen decorations when shown. You can run the layout of your built-in UI elements to account for non-fullscreen system UI through the fitSystemWindows method (android.graphics.Rect). Note: On displays that have a DisplayCutout, the window can still be placed differently than if SYSTEM_UI_FLAG_FULLSCREEN was configured, if the windowmanager window. LayoutParams#layoutInDisplayCutoutMode for WindowManager.LAYOUT_IN_DISPLAY_CUTOUT_MODE_DEFAULT. To avoid this, use any of the other modes. Constant value: 1024 (0x00000400) Added to API level 16 Depreed in API level 30 public static final int SYSTEM_UI_FLAG_LAYOUT_STABLE This constant has been deprecated at API level 30. Use WindowInsets#getInsetsIgnoringVisibility(int) instead to retrieve insets that do not change when system bars change the visibility state. Flag for setSystemUiVisibility(int): When using other layout flags, we would like a stable view of the data content insets to suit Windows Software (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 stock Android UI this is the space for the system bar, navigation bar and status bar, but not more transient elements, as an input method. The stable layout that the UI sees is based on the system UI modes to which you can switch. That is, if you specify SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN, then you will have a stable layout for changes to the SYSTEM_UI_FLAG_FULLSCREEN; if you SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN and SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION, then you can transition to SYSTEM_UI_FLAG_FULLSCREEN and SYSTEM_UI_FLAG_HIDE_NAVIGATION with a stable layout. (Note that you should avoid SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION by yourself.) If you have set the window flag window to hide the status bar (instead of using SYSTEM_UI_FLAG_FULLSCREEN), then a hidden status bar will be considered a stable state for purposes here. This allows your UI to continuously hide the status bar, while still using system UI flags to hide the action bar while still maintaining a stable layout. Note that changing the window's full-screen signage will never provide a stable layout for a clean transition. If you are using the ActionBar in overlay mode with Window#FEATURE_ACTION_BAR_OVERLAY, this flag will also affect the insets that you add to the data to the application. Constant value: 256 (0x000000000) Added to API level 14 Depredated in API level 30 public static final int SYSTEM_UI_FLAG_LOW_PROFILE This constant was deprethered at API level 30. Low profile mode is deprethetised. Hide system bars instead if the application needs to be in discrete mode. Use WindowInsetsController#hide(int) with Type#systemBars(). Flag for setSystemUiVisibility(int): The view has requested that the UI system interface enter a low-profile discrete mode. This is for use in games, book readers, video players or any other immersive application where the usual system chrome is considered very disturbing. In low profile mode, the status bar and/or navigation icons may darken. See also: setSystemUiVisibility(int) Constant value: 1 (0x000000000) Added in API level 16 Deprecated api at API level 30 public static end int SYSTEM_UI_LAYOUT_FLAGS This constant was depreciated at API level 30. System interface layout flags are deprethered. Flags that can impact the layout against the system interface. Constant value: 1536 (0x00000600) public static final int TEXT_ALIGNMENT_CENTER Center the paragraph, for example, ALIGN_CENTER. Use with setTextAlignment(int) Constant value: 4 (0x00000004) public static end int TEXT_ALIGNMENT_GRAVITY Default for root view. Gravity determines the alignment of text, ALIGN_NORMAL, ALIGN_CENTER, or ALIGN_OPPOSITE, which are relative to the text direction of each paragraph. Use with setTextAlignment(int) Constant value: 1 (0x000000001) public static end int TEXT_ALIGNMENT_INHERIT standard text alignment. The text alignment of this view is inherited from its parent. Use with setTextAlignment(int) Constant value: 0 (0x00000000) public static end int TEXT_ALIGNMENT_TEXT_END Align to the end of the paragraph, for example, ALIGN_OPPOSITE. Use with setTextAlignment(int) Constant value: 3 (0x000000003) int public static end int TEXT_ALIGNMENT_TEXT_START Align to the beginning of the paragraph, for example, ALIGN_NORMAL. Use with setTextAlignment(int) Constant value: 2 (0x000000000) public static end int TEXT_ALIGNMENT_VIEW_END Align to the end of the view, which is ALIGN_RIGHT if the layout resolved from the is LTR, and ALIGN_LEFT otherwise. Use with setTextAlignment(int) Constant value: 6 (0x000000006) public static end int TEXT_ALIGNMENT_VIEW_START Align to the beginning of the view, which is ALIGN_LEFT if the resolved layout of the Direction view is LTR, and ALIGN_RIGHT otherwise. Use with with Constant value: 5 (0x000000005) the final public static direction int TEXT_DIRECTION_ANY_RTL the text direction is using the any-RTL algorithm. The direction of the paragraph is RTL if it contains any strong RTL characters, otherwise it is LTR if it contains some strong LTR character. If neither of the two is the direction of the paragraph is the resolved layout direction of the view. Constant value: 2 (0x000000002) the final public static direction int TEXT_DIRECTION_FIRST_STRONG the text direction is using strong algorithm first. The first strong directional character determines the direction of the paragraph. If there is no strong directional character, the direction of the paragraph is the resolved layout direction of the view. Constant value: 1 (0x00000001) the final public static direction int TEXT_DIRECTION_FIRST_STRONG_LTR the text direction is using strong algorithm first. The first strong directional character determines the direction of the paragraph. If there is no strong directional character, the direction of the paragraph is LTR. Constant value: 6 (0x00000006) the final public static direction int TEXT_DIRECTION_FIRST_STRONG_RTL the text direction is using strong algorithm first. The first strong directional character determines the direction of the paragraph. If there is no strong directional character, the direction of the paragraph is RTL. Constant value: 7 (0x000000000) the final public static direction int TEXT_DIRECTION_INHERIT The direction of text is inherited through the ViewGroup Constant Value: 0 (0x000000000) the final static public direction int TEXT_DIRECTION_LOCALE The text direction comes from the Locale system. Constant value: 5 (0x000000005) the final public static direction int TEXT_DIRECTION_LTR The direction of the text is forced to LTR. Constant value: 3 (0x00000003) the final public static direction int TEXT_DIRECTION_RTL The text direction is forced to RTL. Constant value: 4 (0x000000004) protected static final String VIEW_LOG_TAG The registry tag used by this class with android.util.Log. Constant value: Display static protected fields int[] EMPTY_STATE_SET indicates that the view has no defined states. States are used with Drawable to change the view drawing, depending on its state. See also: DrawablegetDrawableState() int static end protected int[] SELECTED_STATE_SET indicates that the view is selected. States are used with Drawable to change the view drawing, depending on its state. See also: DrawablegetDrawableState() int static end protected WINDOW_FOCUSED_STATE_SET indicates that the view window has focus. States are used with Drawable to change the view drawing, depending on its state. See also: DrawablegetDrawableState() Public Constructors public View (Context) Simple-to-use constructor when creating a view from code. Context parameters Context: The context in which the view is running, through which you can access the current theme, resources, etc. public view (context, attrs AttributeSet) constructor that called when inflating an XML view. This is called when a view is being constructed from an XML file, providing attributes that have been specified in the XML file. This version uses a standard default style 0, therefore, the only attribute values applied are those of the Given Context theme and attribute set. The onFinishInflate() method will be called after all children have been added. Context of parameters Context: The context in which the view is running, through which you can access the current theme, resources, etc. attrs AttributeSet: The attributes of the XML tag that is inflating the visualization. This value can be null. See also: View (Context, AttributeSet, int) public view (context, attrs AttributeSet, int defStyleAttr) Run XML inflation and apply a class-specific base style from a themed attribute. This View constructor allows subclasses to use their own base style when they are inflating. For example, the constructor of a Button class would call this version of the super class constructor and provide R.attr.buttonStyle for defStyleAttr; this allows the theme button style to modify all base display attributes (in particular its background) as well as button class attributes. Context of parameters Context: The context in which the view is running, through which you can access the current theme, resources, etc. attrs AttributeSet: The attributes of the XML tag that is inflating the visualization. This value can be null. defStyleAttr int: An attribute in the current theme that contains a reference to a style feature that provides default values for the view. It can be 0 not to look for patterns. See also: View (Context, AttributeSet) public view (context context, attributeset attrs, int defStyleAttr, int defStyleRes) Run XML inflation and apply a class-specific base style from a thematic attribute or style feature. This View constructor allows subclasses to use their own base style when they are inflating. When determining the final value of a given attribute, there are four entries that come into play: Any attribute values in the given attribute set. The style feature specified in attributeset (called style). The default style specified by defStyleAttr. The default style specified by defStyleRes. The basic values of this theme. Each of these entries is considered in order, with the first listed taking precedence over the following. In other words, if in AttributeSet you <Button *= textcolor=#ff000000> provided, then the button text will always be black, regardless of what is specified in any of the styles. Context of parameters Context: The context in which the view is running, through which you can access the current theme, resources, etc. attrs AttributeSet: The attributes of the XML tag that is inflating the visualization. This value can be null. defStyleAttr int: An attribute in the current theme that contains a reference to a style feature that provides default values for the view. It can be 0 not to look for patterns. defStyleRes int: One of features of a style feature that provides default values for the view, used only if defStyleAttr is 0 or cannot be found in the theme. It can be 0 not to look for patterns. See also: </Button> </Button> AttributeSet, int) Public methods of public void add ChildrenForAccessibility (ArrayList<View> outChildren) Adds the children of this view relevant to accessibility to the list given as output. Since some visions are not important for accessibility, the additional views of children are not necessarily direct children of this view, but rather the first level of descendants important for accessibility. OutChildren ArrayList parameters: The output list that children will receive for accessibility. Public void addedExtraDataToAccessibilityNodeInfo (Accessibility InformationNodeInfo, String extraDataKey, Bundle Arguments) Adds extra data to an Accessibility Site based on an explicit request for additional data. This method only needs overloading if the node is marked as having extra data available. AccessibilityNodeInfo info parameters: The information to add the extra data to. Never null. This value cannot be null. extraDataKey String: A key specifying the type of extra data to add to the information. The extra data must be added to the Bundle returned by the AccessibilityNodeInfo#getExtras method of the information. Never null. This value cannot be null. Bundle arguments: A package holding any arguments relevant to this request. It can be null if the service did not provide arguments. This value can be null. See also: AccessibilityNodeInfo.setExtraData available (List) public void adductorFodes<View> (arraylist views, direction int) Add any focusable views that are descending from this view (possibly including this view if it is focusable) to the visualizations. If we're in touch mode, just add views that are also focusable in touch mode. public void addFocusables<View> (ArrayList views, int direction, int focusableMode) Adds any focusable views that are descending from that view (possibly including this view if it is focusable in itself) to the views. This method adds all focusable views, regardless of whether we're in contact mode or only focusable views in touch mode if we're in touch mode or only views that might have accessibility focus if accessibility is enabled depending on the focusable mode parameter. See also: FOCUSABLES_ALLFOCUSABLES_TOUCH_MODE empty public addsKeyboardNavigationClusters<View> (collection views, direction int) Adds any keyboard navigation cluster roots that are descending from this view (possibly including this view if it is a cluster root itself) to the visualizations. Parameters views collection: Keyboard navigation cluster roots found so far This value cannot be null. direction int: Direction to look public empty addOnLayoutChangeListener listener) Add a listener that will be called when the view boundaries change due to layout processing. Listener parameters View.OnLayoutChangeListener: The listener that will be called when the layout boundaries change. empty public addTouchables<View> (arrayList views) Add any touchable views that are descending from that view (possibly including this view if it is touchable itself) to the visualizations. ArrayList view parameters: Touch-sensitive views found</View> </View> </View> </View> </View> </View> </View> Visualexresto public ViewPropertyAnimator animates () This method returns a ViewPropertyAnimator object, which can be used to animate specific properties in this view. Returns ViewPropertyAnimator ViewPropertyAnimator The ViewPropertyAnimator associated with this View. Public empty announcementAccessibility (CharSequence text) Convenience method for sending an Accessibility EventEvent#TYPE_ANNOUNCEMENT Accessibility To suggest that an accessibility service advertise the specified text to its users. Note: The event generated with this API has no semantic meaning, and is appropriate only in exceptional situations. Applications can often achieve the correct behavior for accessibility by accurately providing the semantics of their UI. They should not specify what exactly is advertised to users. Text of CharSequence parameters: The ad text. Public void autofill (autofillValue value) automatically fills the contents of this view with the value. Visualizations support the Autofill Framework primarily by: Providing the metadata that defines what the view means and how it can be filled automatically. Implementing methods that automatically filter the view. onProvideAutofillStructure(android.view.ViewStructure, int) is responsible for the first, this method is responsible for the latter. This method does nothing by default, but when replaced normally: checks whether the supplied value matches the expected type (which is set by getAutofillType()). Checks to see if the view is editable - if it is not, it should return immediately. Call the appropriate getter method in AutofillValue to get the actual value. Pass the actual value to the equivalent setter in the view. For example, a text field view could implement the method this way: @Override autofill of public void (autofillValue value) { if (!value.isText() || |this.isEditable()) { return; } CharSequence text = value.getTextValue(); if (text != null) { this.setText(text); } If the value is updated asynchronously, the next call to AutofillManager#notifyValueChanged(View) should happen after the value has changed to the automatically populated value. If not, the view will not be considered automatically populated. Note: After this method is called, the value returned by getAutofillValue() must be equal to the value passed to it, otherwise the preview will not be highlighted as automatically populated. Value of autocomplete value parameters: value to be filled automatically. public empty autofill<AutofillValue> (SparseArray values) automatically fills the contents of virtual children within that view. Visualizations with virtual children support the Autofill Framework primarily by: Providing metadata that defines what virtual children mean and how they can be Automatically. Implementing methods that automatically fill virtual children. onProvideAutofillVirtualStructure (android.view.ViewStructure, int) is responsible for the first, this method is responsible for the latter - see autofill (android.view.autofill.AutofillValue) and onProvideAutofillVirtualStructure (android.view.ViewStructure, int) for more information about autofill. If a </AutofillValue> </AutofillValue> value is updated asynchronously, the next call to AutofillManager#notifyValueChanged (Preview, int, AutofillValue) must happen after the value has changed to the automatically filled value. If not, the child will not be considered automatically populated. Note: To indicate that a virtual view has been filled automatically, ?android:attr/autofilledHighlight must be drawn on it until the data changes. Related XML attributes: android:auto filledHigh light seseses value SparseArray: map of values to be filled automatically, keyed by child virtual identification. This value cannot be null. Added to the Deprecated Level 4 API in the Public Empty Level 28 API buildDrawingCache (boolean autoScale) This method was deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Forces the drawing cache to be constructed if the drawing cache is invalid. If you call buildDrawingCache() manually without calling setDrawingCacheEnabled(true), you must clear the cache by calling destroyDrawingCache() later. Note about autoscaling in compatibility mode: When autoscaling 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 the screen can show scale artifacts. To avoid such artifacts, you should call this method by setting autoscaling to true. Doing so, however, will generate a bitmap of a different size than the view. This implies that your application should 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 preview to render in software once, negatively impacting Performance. AutoScale boolean parameters See also: getDrawingCache()destroyDrawingCache() Added in api level 1 depreed in api level 28 of empty public buildDrawingCache () This method was deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers largely unnecessary and can easily result in a net loss of performance due to the cost of creating and upgrading the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Calling this method is equivalent to calling buildDrawingCache (false). See also: buildDrawingCache (boolean) public void buildLayer () Forces the layer creation of this view and this view to be rendered on your layer. If the layer type of this view is set to LAYER_TYPE_NONE, invoking this method has no effect. This method can, for example, be used to render a view on your layer before starting an animation. If this view is complex, rendering on the layer before starting the animation will avoid skipping frames. Play IllegallyStateException If this view is not attached to a window See also: setLayerType(int, android.graphics.Paint) public call booleanaOnClick () Call directly to any attached OnClickListener. Unlike performClick(), this only calls the listener, and does not do any associated click action, such as reporting an accessibility event. True boolean returns there was an assigned OnClickListener that was called, false otherwise returned. Public paplean caneResolveLayoutDirection () Make sure that the layout direction resolution can be made. Returns true Boolean if the resolution of the layout direction can be otherwise returned false. Public paplean caneResolveTextAlignment () Make sure that the text alignment resolution can be made. Returns true Boolean if the text alignment resolution can otherwise be returned false. Public nozzle ResolveTextDirection () Make sure that the resolution of the text direction can be made. Returns true Boolean if the text direction resolution can otherwise be made false return. canScrollcrollcrollalmente (direction int) Make sure that this view can be scrolled horizontally in a certain direction. Int direction parameters: Negative to check left scrolling, positive to check right scrolling. Returns true Boolean if this view can be scrolled in the specified direction, false otherwise. canScrollVerically (int direction) check that this view scrolled vertically in a certain direction. Int direction parameters: Negative to check scrolling up, positive to check scrolldown. Returns true Boolean if this view can be scrolled in the specified direction, false otherwise. cancellation of public voidLongPress () Cancels a long pending press. Your subclass can use this if you you want the context menu to rise if the user presses and keeps in the same place, but you don't want it to pone if they press and then move enough to cause scrolling. Public final cancellationPendingPendingEvents () Cancel any deferred high-level input events that were previously posted to the event queue. Many visualizations post high-level events, such as click handlers in the event queue to be deferred, in order to preserve a desired user experience - clearing visible pressed states before running, 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 CancelPendingInputEvents() and remove pending events from the queue. This will also cancel pending incoming events for any child preview. Note that this may not be enough as a debouncing strategy for clicks in all cases. This will not affect newer events published after this call that may occur as a result of lower-level input events still waiting in the queue. If you are trying to avoid doubly submitted events during some kind of asynchronous transaction, you should also take other steps to protect against unexpected double entries, for example. calling setEnabled (false) and reactivating the view when the transaction is completed, tracking transaction IDs already sent, etc. booleanaInputConnectionProxy (Preview) public check called by InputMethodManager when a view that is not the current incoming connection target is trying to make a call in the manager. The default implementation returns false; you can override this to return true to certain visualizations if you are running inputconnection proxying for them. Parameter view: The view that is making the inputmethodmanager call. Returns Boolean True Return to allow the call, false to reject. empty public clearAnimation () Cancels any animations for this view. Empty public (clearFocus) Called when this view wants to give up focus. If focus is released on FocusChanged (boolean, int, android.graphics.Rect) is called. Note: When not in contact mode, the framework will attempt to focus on the first focalable display of the top after releasing the focus. Therefore, if this View is the first top that can have focus, then all returns related to the compensation focus will be invoked after which the framework will focus on that view. int public static combinmeasuredstates (int curState, int newState) merges two states as returned by getMeasuredState(). CurState int parameters: The current state as returned from a view or the result of combining multiple views. newState int: The new state of to match. Returns int returns a new integer reflecting the combination of the two states. public void (computeScroll) Called by a parent to request that a child update their values to mScrollX and mScrollY if necessary. This is usually done if the child is animating a scroll using a Scroller object. Public public WindowInsets (WindowInsets in, Rect outLocalInsets) Computational insets that should be consumed by this vision and those that should propagate to those under it. Parameters in WindowInsets: Insets currently being processed by this view, probably received as a parameter for onApplyWindowInsets (android.view.WindowInsets). outLocalInsets Rect: A Rect that will receive the insets that must be consumed by this preview Returns WindowsInsets Insets that must be passed to previews under this Added API in api level 1 depreed in api level 28 empty public destroyDrawingCache () This method has been depreed in api level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Frees the resources used by the drawing cache. If you call buildDrawingCache() manually without calling setDrawingCacheEnabled(true), you should clear the cache with this method later. public windowsInsets dispatchApplyWindowInsets (Insets WindowInsets) request to apply the given window insets to this view or other view in your subarnão. This method should be called by clients who want to apply insets corresponding to areas obscured by window decorations or overlays. This can include status and navigation bars, action bars, input methods, and more. New entry categories may be added in the future. The method returns the provided insets minus those that were applied by this view or its children. Customers who want to provide custom behavior must override the onApplyWindowInsets method (android.view.WindowInsets) or alternatively provide an OnApplyWindowInsetsListener method through the setOnApplyWindowInsetsListener method. This method replaces the older fitSystemWindows method. WindowInsets parameter insets: Insets to apply insets returns The insets provided minus the insets that were consumed in booleanoCapturedPointerEvent public dispatch (event Pass a captured pointer event to the focused view. MotionEvent parameter events: The motion event to be dispatched. True boolean returns if the event was handled by the view, false otherwise. empty public dispatchDisplayHint (int hint) Dispatch a hint tip if this view is displayed. For example, when a View moves off the screen, it might receive a display hint indicating that the view is not displayed. Applications should not trust this tip because there is no guarantee that they will receive one. boolean public dispatchDragEvent (DragEvent event) Detects whether this view is enabled and has a drag event listener. If both are true, then it calls the drag event listener with the DragEvent it received. If the drag event listener returns true, then dispatchDragEvent() returns true. For all other cases, the method calls the drag event handler method onDragEvent() 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. DragEvent event parameters public dispatchDDwableHotspotChanged (float x, float y) Dispatches drawableHotspotChanged for all children in this View. Parameters x fluctuation: hotspot x coordinate y float: hotspot y coordinate See also: drawableHotspotChanged (float, float) public void dispatchFinishTemporaryDetach () Dispatch onFinishTemporaryDetach() for this View and its direct children if this is a View container. If you override this method, you must call up the superclass implementation. boolean public dispatchGenericMotionEvent (MotionEvent event) Dispatch a generic motion event. Generic motion events with the inputdevice#source class are SOURCE_CLASS_POINTER delivered to view under the pointer. All other generic motion events are delivered to focal vision. Hover events are handled specially and are delivered in onHoverEvent (android.view.MotionEvent). MotionEvent parameter events: The motion event to be dispatched. True boolean returns if the event was handled by the view, false otherwise. Public Boolean dispatchKeyEvent (KeyEvent event) Send a key event to the next view in the focus path. This path goes from the top of the vision tree to the currently focused view. If this view has focus, it will be sent to itself. Otherwise, it will dispatch the next node through the focus path. This method also triggers any key listener. KeyEvent parameter event: The key event to be dispatched. True boolean returns if the event was handled, false otherwise. Public Boolean dispatchKeyEventPreIme (KeyEvent event) dispatches a key event before being 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 to deal with the BACK key to update the application UI instead of allowing the IME to see it and close it. KeyEvent parameter event: The key event be dispatched. True boolean returns if the event was handled, false otherwise. Public Boolean dispatchKeyShortcutEvent (KeyEvent event) dispatches a key shortcut event. KeyEvent parameter event: The key event to be dispatched. Returns Returns True if the event was handled by the view, false otherwise. Public boolean dispatchNestedFling (floating speedX, floating speedY, boolean consumed) Dispatches a throw to a nested scrolling parent. This method should be used to indicate that a nested scrolling child has detected appropriate conditions for a case. This usually means that a touch roller has finished with a VelocityTracker in the direction of scrolling that meets or exceeds the #getScaledMinimumFlingVelocity() display setting along a scrollable axis. If a nested scrolling child view would typically launch, but is on the edge of its own content, it can use this method to delegate the case to its nested scrolling parent. The parent can optionally consume the case or observe a child case. Parameters speedX float: Horizontal throwing speed in pixels per second speedIf fluctuation: Vertical throwing speed in pixels per second consumed boolean: true if the child consumed the throw, false otherwise true boolean returns if the nested scrolling parent consumed or otherwise reacted to the booleanonestedRFling public dispatch (floating speedX, floating speedVo) Dispatch a pitch to a nested scrolling parent before being rendered by this view. Nested pre-pitch events are for nested events that touch trap is touching and what nested preparch is for nested parchment. DispatchNestedPreFling compensates for an opportunity for the parents' vision in a nested case to fully consume the case before the child's vision consumes it. If this method returns true, a nested parent view consumed the case, and this view should not scroll as a result. For a better user experience, only one view in a nested scroll chain should consume the case at a time. If a parent view consumed the case this method returns false. Custom preview implementations should explain this in two ways: If a custom view is called and needs to accommodate a fixed-page point, do not call the NestedPreFling dispatch; consume the case and resolve to a valid position independently. If a nested parent consumes the case, this view should not scroll at all, even to pin itself back to a valid idle position. Opinions should also not offer pitch speeds for views of parents nested along an axis where scrolling is not currently supported; a ScrollView should not offer a horizontal throw speed to your parents, since scrolling along that axis is not allowed and loading speed along that movement makes no sense. Speed of parametersX float: Horizontal pitch speed in pixels by second speedSpere buoy: Vertical pitch speed in pixels per second Returns true boolean if a nested scrolling parent the Boolean public dispatchNestedPrePerformAccessibilityAction (int action, Bundle arguments) Report an accessibility action to the parents of this view for delegate processing. PerformAccessibilityAction (int, android.os.Bundle) implementations can internally call this method to delegate an accessibility action to a supporting parent. If If You parent returns true of its ViewParent#on MethodNestedPrePerformAccessibilityAction (View, int, android.os.Bundle) this method will return faithful to mean that the action has been consumed. This method is useful for implementing nested scrolling child views. If isNestedScrollingEnabled() returns true and the action is a scrolling action, a custom view implementation can invoke this method to allow a parent to consume the parchment first. If this method returns true, the custom view should skip its own scrolling behavior. Parameters action int: Accessibility action for delegates arguments package: Optional action arguments returns true boolean if the action was consumed by a boolean parent public dispatchNestedPreScroll (int dx, int dy, int[] int[] offsetInWindow) Dispatch a step of a nested parchment in progress before this view consumes any part of it. Nested preparch events are for nested parchment events that touch interception is touching. dispatchNestedPreScroll provides an opportunity for viewing parents in a nested scroll operation to consume part or all of the parchment operation before the child's view consumes it. Dx int parameters: Horizontal distance of parchment in dy int pixels: Vertical distance of parchment in consumed pixels int: Output. If not null, the consumed[0] will contain the consumed dx component and consume[1] the consumed dy. This value can be null. offsetInWindow int: Optional. If not null, on the return, this will contain the offset in the local view coordinates of this view point before this operation to after it is completed. Preview implementations can use it to adjust the expected input coordinate tracking. This value can be null. True boolean returns if the parent consumed part or all of the parchment delta See also: dispatchNestedScroll (int, int, int, int[]) boolean onestedScroll public dispatch (int dxConsumed, int dyConsumed, int dxUnconsumed, int dyconsumed, int[] offsetInWindow) Dispatch a step of a nest parchment in progress. Visualization implementations that support nested scrolling should call this to report information about an in-progress scroll to the current nested scrolling parent. If a nested scroll is not in progress or nested scrolling is not enabled for this view, this method does nothing. Compatible view implementations should also call the NestedPreScroll dispatch before consuming a component of the scroll event itself. dxConsumed int parameters: Horizontal distance in pixels consumed by this view during this scrolling step dyConsumed int: Vertical distance in pixels consumed by this view during this scrolling step dxInconsumed int: Horizontal scrolling distance in pixels not consumed by this view int: Horizontal scrolling distance in pixels not consumed by this offsetInWindow int: Optional view. If not null, on the return, this will contain the offset in the local view coordinates of this view point before this operation to after it is completed. Preview implementations can use it to adjust the expected input coordinate tracking. This value can be True boolean returns if the event was dispatched, false if it could not be dispatched. See also: dispatchNestedPreScroll (int, int, int[], int[]) public empty dispatchProvideAutofillStructure (View structure, flags int) Dispatches the creation of a ViewStructures for autofill purposes below the hierarchy, when a Structure Wizard is being created as part of an autofill request. The default implementation does the following: Typically, this method should only be overridden by subclasses that provide a view hierarchy (such as ViewGroup) - other classes should replace the Android.ViewStructure (android.viewStructure, int) or proProvideAutofillVirtualStructure (android.viewStructure, int). When replaced, it should: call super.dispatchProvideAutofillStructure (structure, flags) or explicitly set the AutofillId in the structure (for example, by calling structure.setAutofillId(getAutofillId()). Decide how to deal with the AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS flag - when defined, all views of the structure should be considered important for autocomplete, regardless of what is portanteParaAutofill(). We encourage you to respect this flag to provide a better user experience - this flag is typically used when a user explicitly requested autocomplete. If the flag is not set, only visualizations marked as important for autocomplete should be included in the structure - skipping non-important visualizations optimizes overall autocomplete performance. Parameter structure Display structure: Fill in structured visualization data for autocomplete purposes. This value cannot be null. Flags Int: Optional flags. The value is 0 or AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS See also: AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS dispatch of public voidStartTemporaryDetach () Dispatch onStartTemporaryDetach() for this view and its direct children, if this is a View container. If you override this method, you must call up the superclass implementation. Public Boolean dispatchTouchEvent (MotionEvent event) Pass the motion event from the touch screen to the target view, or this view if it is the target. MotionEvent parameter events: The motion event to be dispatched. True boolean returns if the event was handled by the view, false otherwise. Public Boolean DispatchTrackballEvent (MotionEvent event) Pass a trackball motion event to the focused view. MotionEvent parameter events: The motion event to be dispatched. True boolean returns if the event was handled by the view, false otherwise. public boolean dispatchUnhandledMove (See focused, direction) This method is the last chance for focused vision and its ancestors to respond to a key Arrow. This is called when focused vision did not consume the key internally, nor could the visualization system find a new view in the direction requested to focus. Focused Parameters Vision: The currently focused vision. int direction: The focus of the direction wants to move. One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT and FOCUS_RIGHT. The value is is FOCUS_UP, FOCUS_RIGHT, FOCUS_DOWN returns boolean True if this view consumed this uncontrolled movement. Empty public dispatchWindowFocusChanged (boolean hasFocus) called when the window containing this view gains or loses the focus of the window. The ViewGroups must override to forward to their children. Parameters has Focus boolean: True if the window that contains this view now has focus, false otherwise. Public empty dispatchWindowVisibilityChanged (visibility int) Dispatches a change of window visibility by the view hierarchy. The ViewGroups must override to forward to their children. See also: noWindowVisibilityChanged(int) public void draw (screen screen) Manually render this view (and all your children) to the given screen. The view must have already made a complete layout before this function is called. When implementing a view, implement onDraw(android.graphics.Canvas) instead of overwriting this method. If you need to override this method, call the superclass version. If you override this method, you must call up the superclass implementation. Parameters screen: The screen to which the display is rendered. public void drawableChanged (float x, float y) This function is called whenever the display hotspot changes and needs to be propagated to drawables or views of children managed by the view. Sending to child views is handled by dispatching DrawableHotspotChanged (float, float). Be sure to connect to the superclass when replacing this function. If you override this method, you must call up the superclass implementation. Parameters x fluctuation: hotspot x coordinate y float: hotspot y coordinate public View findFocus () Find the view in the hierarchy rooted in this view that currently has focus. View returns The view that currently has focus or null if no focused views can be found. Public end T findViewById (int id) Finds the first descending view with the given ID, the view itself if the ID matches getId(), or null if the ID is invalid (< 0) or there is no corresponding display in the hierarchy. Note: In most cases - depending on compiler support - the resulting view is automatically posted to the target class type. If the target class type is not constricted, an explicit cast may be required. Parameters id int: the ID to search for returns T a visualization with given id if found, or null otherwise See also: public end T findViewWithTag (Object Tag) Look for a child view with the given tag. If this view has the tag given, return this view. Object tag parameters: The tag to search for, using tag.equals(getTag(). Returns T The View that has the given tag in the nullSearch public view hierarchy or focus (direction int) Find the nearest view in the specified direction may have focus. That doesn't focus on that view. The most focused returns view in the specified direction or null if none can be found. Public empty strengthLayout () Forces this view to be set during the next layout pass. This method does not call requestLayout() or or about the father. int static public generateViewId () Generate an appropriate value for use in setId(int). This value will not collide with the ID values generated at the time of construction by AAPT to R.id. Returns int a public value generated ID CharSequence getAccessibilityClassName () Return the class name of this object to be used for accessibility purposes. Subclasses should only override this if they are implementing something that should be seen as a completely new view class when used by accessibility, unrelated to the class from which it is derived. This is used to populate the AccessibilityNodeInfo#setClassName. public int getAccessibilityTraversalAfter () gets the id of a view after which it is visited on accessibility traversal. Returns int The identity of a view this succeeds in traversing accessibility if specified, otherwise it NO_ID. See also: setAcessibilityTraversalAfter(int) public int getAccessibilityTraversalBefore () gets the id of a view before which it is visited at accessibility traversal. Returns int the id of a visualization that precedes the accessibility traverse if specified, otherwise it NO_ID. See also: setAcessibilityTraversalBefore(int) public float getAlpha () The opacity of the view. This is a value from 0 to 1, where 0 means that the view is completely transparent and 1 means that the view is completely opaque. By default, this is 1.0f. Returns fluctuate View opacity. public animation (getAnimation) Get the animation currently associated with this view. Animation returns the animation being played or programmed to play for this view. public IBinder getApplicationWindowToken () Retrieve a unique token by identifying the actual top-level window of the window to which this view is attached. That is, it is like getting WindowToken(), except if the window in which this view is a pane window (attached to another containing window), then the contention window token is returned instead. int[] getAttributeResolutionStack (attribute int) returns the ordered list of resource ID that are considered when resolving attribute values for this view. The list will include the layout feature ID if the View is inflated from XML. It will also include a set of explicit styles if specified in XML using style=.... Finally, it will include the default styles resolved from the theme. Note: This method will only return actual values if display attribute debugging is enabled in Android developer options. Int parameter attribute: Assign resource iD to which the resolution stack should be returned. Int[] ordered list of resource ID that are considered when resolving attribute values for this view. This value cannot be null. Public map<Integer, integer=> getAttributeSourceMap () Returns attribute resource ID mapping for iD source resource where the attribute value was set. The source resource ID can be a layout resource ID, whether the value was set in XML within the View tag or a style resource ID, if the attribute was set on a </Integer,> </Integer,> The value of the source resource will be one of the resource IDs of the getAttributeSourceResourceMap(). Note: This method will only return actual values if display attribute debugging is enabled in Android developer options. Map mapping<Integer, integer=> of returns from attribute resource ID to source resource id where the attribute value was set. This value cannot be null. int public getAutofillType () Describes the type of autofill in this view, so that an AutofillService can create the appropriate AutofillValue by automatically filtering the view. By default return AUTOFILL_TYPE_NONE, but visualizations must override it to properly support the Autofill Framework. public int getBaseline () Return the widget baseline compensation from the widget's upper bound. If this widget does not support baseline alignment, this method returns -1. Returns int the offset of the baseline within the boundaries of the widget or -1 if the baseline alignment is not supported in the end public position getBottom () Lower position of this view relative to its parent. Returns int The bottom of this view, in pixels. getCameraDistance (public fluctuation) gets the distance along the Camera's Z axis for this view. Returns float Distance along the Z axis. See also: public boolean getClipBounds (Rect outRect) fills an output rectangle with the clip boundaries of the view, returning true if it is successful or false if the preview clip limits are null. OutRect Rect parameters: Rectangle in which placing the clip boundaries of the view returns true boolean if successful or false if the clip boundaries of the view are public null Ret getClipBounds () Returns a copy of the current ClipBounds. Recurrent Returns A copy of the current clip limits if clip limits are set, otherwise null. Public end context (getContext) Returns the context in which the view is being retreated, through which it can access the current theme, resources, etc. Context returns The context of the view. final public boolean getDefaultFocusHighlightEnabled () /** Returns whether this View should use a default focus highlight when it focuses, but has R.attr.state_focused set on its background. Related XML attributes: android:defaultFocusHighlightEnabled Returns boolean True if this view should use a default focus highlight. public static int get DefaultSize (size int, int medeSpec) utility to return a default size. Uses the supplied size if MeasureSpec has not imposed restrictions. It's going to get bigger if allowed by MeasureSpec. Parameters size int: Default size for this display measureSpec int: Constraints imposed by the parent returns int The size of this view should be. public display (getDisplay) Receives the logical viewfinder to which the display window was attached. Display of The logical or null viewfinder if the display is not currently attached to a window. Added to the Deprecated Level 1 API in the Bitmap (Public Level 28 Level 28) API ( ) This method has been deprecated at API level 28. The vision drawing cache was largely rendered obsolete with the introduction of accelerated hardware

</Integer,> </Integer,> api 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Calling this method is equivalent to calling getDrawingCache (false). Bitmap returns An unsized bitmap representing this view or null if the cache is disabled. See also: Added to the Deprecated Level 4 API in the Bitmap public level 28 API getDrawingCache (boolean autoScale) This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Returns the bitmap in which this display drawing is cached. The returned bitmap is null when the cache is disabled. If the cache is enabled and the cache is not ready, this method will create it. Draw call (android.graphics.Canvas) will not be removed from the cache when caching is enabled. To benefit from caching, you must request the loot cache by calling this method and draw it on the screen if the returned bitmap is not null. Note about autoscaling in compatibility mode: When autoscaling is not enabled, this method will create a bitmap of the same than this view. Because this bitmap will be drawn scaled by the parent ViewGroup, the result on the screen can show scale artifacts. To avoid such artifacts, you should call this method by setting autoscaling to true. Doing so, however, will generate a bitmap of a different size than the view. This implies implying your application should be able to handle this size. Boolean auto-scaling parameters: Indicates whether the generated bitmap should scale based on the current density of the screen when the application is in compatibility mode. Bitmap returns a bitmap representing this view or null if the cache is disabled. Added to the Deprecated Level 1 API in api level 28 int public getDrawingCacheBackgroundColor () This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Int returns The background color used for the drawing cache bitmap See also: setDrawingCacheBackgroundColor(int) Added in api level 1 depreed at API level 28 int getDrawingCacheQuality () This method was deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Returns the quality of the drawing cache. Related XML attributes: empty android:drawingCacheQuality getDrawingRect (Rect outRect) Return the visible drawing boundaries of your view. Fills the output rectangle with the values of getScrollX(), getScrollY(), getWidth(), and getHeight(). These limits do not account for any transformation properties currently defined in the vision, such as setScaleX (float) or setRotation (float). Rect outRect parameters: The drawing boundaries (scrolled) of the view. long ago getDrawingTime () Return the time at which the drawing of the preview hierarchy began. Long returns the drawing start time in public milliseconds to float (getElevation) The base elevation of this view relative to its parent, in pixels. Returns float The base depth position of the view, in pixels. public int getExplicitStyle () Returns the resource ID for the specified style using style=... in attributeset support XML element or resources#ID_NULL otherwise, if not specified or not applicable. Each view can have an explicit style specified in the layout file. This style is used first during the resolution of the View attribute, then if an attribute is not defined there, the resource system looks at the default style and the theme as indentations. Note: This method will only return actual values if display attribute debugging is enabled in Android developer options. Returns int The resource ID for the specified style using style=... in attributeset support XML element or resources#ID_NULL otherwise, if not specified or not applicable. Public void getFocusedRect (Rect r) When a visualization has focus and the user navigates away from it, the next view is searched to start from the rectangle filled by this method. By default, the rectangle is the 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 must override this method and fill in a more specific rectangle. R Rectified parameters: The rectangle to fill, in the coordinates of this visualization. Drawable public getForeground () Returns the drawable used as the foreground of this View. The foreground drawn, if not null, is always drawn on top of the contents of the view. Drawable returns to Drawable or null if no foreground has been defined See also: boolean end public getGlobalVisibleRect (Rect r, Point globalOffset) If any part of this view is not cut off by any of your parents, then return that area in r in global coordinates (root). To convert r to local coordinates (without taking into account possible display rotations), compensate for -globalOffset (for example, r.offset (-globalOffset.x, -globalOffset.y)). If the view is completely cropped or translated, return false. R Rectify parameters: If true is returned, r holds the global coordinates of the visible part of this view. globalOffset Point: If true is returned, the globalOffset holds the dx,dy between this view and its root. globalOffet may be null. Returns true Boolean if r is not empty (that is.part view is visible at the root level. The public handler (getHandler) returns handler a handler to the segment that runs the View. This handler can be used to pump events into the UI event queue. public end int getHeight () Return the height of your vision. Returns int The height of vision, in pixels. empty public getHitRect (Rect outRect) Hit rectangle in the coordinates of the parents Parameters outRect Rect: The view impact rectangle. public int getHorizontalFadingEdgeLength () Returns the size of the faded horizontal edges used to indicate that more content in this view is visible. Related XML attributes: Returns int The pixel size of the faded horizontal border or 0 if the faded horizontal borders are not enabled for this view. int public getLabelFor () gets the ID of a vision for which this view serves as a label for accessibility purposes. Returns int The labeled display id. public ViewGroup.LayoutParams getLayoutParams () Get the LayoutParams associated with this view. All visualizations must have layout parameters. These parameters supply to the parent of this view specifying how it should be organized. There are many viewgroup.LayoutParams subclasses, and these correspond to the different ViewGroup subclasses that are responsible for organizing your children. This method can return null if this view is not attached to a parent ViewGroup or set LayoutParams (android.view.ViewGroup.LayoutParams) was not successfully invoked. When a view is attached to a parent ViewGroup, this method should not return null. Returns ViewGroup.LayoutParams The LayoutParams associated with this view, or null if no parameters have yet been set in public int getLeft () Left position of this view relative to its parent. Returns int The left edge of this view, in pixels. Public final boolean getLocalVisibleRect (Rect r) empty public getLocationInSurface (int[] location) Calculate the view coordinate within the surface. Calculates the coordinates of this view on its surface. The argument must be an array of two integers. After the method returns, the array contains the x and y location in that order. Int location parameters: An array of two integers in which to maintain coordinates this value cannot be null. public empty getLocationInWindow (int[] outLocation) calculates the coordinates of this view in your 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 outLocation int: A set of two integers in which to keep the coordinates the public void getLocationOnScreen (int[] outLocation) calculates 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 outLocation int: A set of two integers in which to keep public coordinates int getMeasuredHeightAndState () Return the total height measurement information for this visualization computed by the most recent call to measure (int, int). This result is a mask somewhat defined by MEASURED_SIZE_MASK and MEASURED_STATE_TOO_SMALL. This should only be used measurement and layout calculations. Use getHeight() to see how high a view is after the layout. Returns int The measured height of this view as a bit mask. Bit. int final getMeasuredWidthAndState () Return the total width measurement information for this visualization, as calculated by the most recent call to measure (int, int). This result is a mask somewhat defined by MEASURED_SIZE_MASK and MEASURED_STATE_TOO_SMALL. This should be used only during measurement and layout calculations. Use getWidth() to see how wide the view is after the layout. Returns int The measured width of this visualization as a bit mask. public int getNextClusterForwardId () gets the root id of the next keyboard navigation cluster. Related XML attributes: android:nextClusterForward Returns int The next ID of the keyboard navigation cluster or NO_ID whether the framework should decide automatically. public int getNextFocusDownId () Receives the view id to use when the next focus is FOCUS_DOWN. Related XML attributes: Returns int The next focus ID or NO_ID whether the framework should decide automatically. public int getNextFocusLeftId () Receives the view id to use when the next focus is FOCUS_LEFT. Related XML attributes: Returns int The next focus ID or NO_ID whether the framework should decide automatically. public int getNextFocusRightId () Receives the preview id to use when the next focus is FOCUS_RIGHT. Related XML attributes: Returns int The next focus ID or NO_ID whether the framework should decide automatically. public int getNextFocusUpId () Receives the preview id to use when the next focus is FOCUS_UP. Related XML attributes: Returns int The next focus ID or NO_ID whether the framework should decide automatically. Public ViewOverlay (getOverlay) Returns the overlay for this view, creating it if it does not already exist. Adding drawables to the overlay will cause them to appear whenever the view itself is redrawn. Objects in the overlay should be actively managed: remove them when they should no longer be displayed. The overlay will always be the same size as your host view. Note: Overlays don't work right at the moment with SurfaceView or TextureView; content in overlays for these types of views may not display correctly. Returns the ViewOverlay object to the ViewOverlay for this view. int public getPaddingBottom () Returns the bottom padding of this view. If there are inset and enabled scroll bars, this value can include the space required to display the scroll bars as well. Returns int the bottom padding in public pixels int getPaddingEnd () Returns the final fill of this view, depending on the direction of layout resolved. If there are inset and enabled scroll bars, this value can include the space required to display the scroll bars as well. Returns int the final fill in public pixels int getPaddingLeft () Returns the left fill of this view. If there are inset and enabled scroll bars, this can include the space required to display the scroll bars as well. Returns int the left fill in public pixels int getPaddingRight () Returns the right fill of this view. If there are inset and enabled scroll bars, this value can include the space required to the scroll bars too. Returns int the right fill in public pixels int getPaddingStart () Returns the initial fill of this view, depending on the layout direction resolved. If there are inset and enabled scroll bars, this value can include the space required to display the scroll bars as well. Returns int the initial fill in public pixels int getPaddingTop () Returns the top fill of this view. Returns int the top fill in public pixels end ViewParent (getParent) Receives the parent of this view. Note that the parent is a ViewParent and not necessarily a View. Returns See Parent Preview of this view. Public ViewParent getParentForAccessibility () Receives the parent for accessibility purposes. Note that the parent for accessibility does not need the immediate parent. It is the first predecessor that is important for accessibility. Returns Verparent The parent for accessibility purposes. final audience boolean getRevealOnFocusHint () Returns the preference of this view for revealing behavior when it gains focus. When this method returns true to a child view requesting focus, the opinions of ancestors responding to a change of focus in ViewParent#requestChildFocus(View, View) should make a better effort to make the newly focused child fully visible to the user. When it returns false, ancestor views should preferably not interrupt scrollplacement or other properties that affect user visibility as part of the focus shift. Returns true boolean if this visualization prefers to become fully visible when it gains focus, false if you prefer not to interrupt scroll placement See also: setRevealOnFocusHint(boolean) public end position int getRight () Correct position of this view relative to its parent. Returns int The right edge of this view, in pixels. public View (getRootView) Finds the highest view in the current view hierarchy. Returns Display the highest view that contains this view of public windows getRootWindowInsets () Provide original window sets that are sent to the view hierarchy. Insets are only available if the view is attached. Returns windowsInsets from the top of the view hierarchy or null if the View is highlighted public float (getScaleX) The amount that the view is scaled in x around the pivot point, as a proportion of the unscalded width of the view. A value of 1, the default, means that no escalation is applied. By default, this is 1.0f. Returns fluctuate The scale factor. See also: the public float (getScaleY) The amount that the view is sized in y around the pivot point, as a proportion of the non-escalating height of the view. A value of 1, the default, means that no escalation is applied. By default, this is 1.0f. Returns fluctuate The factor Scale. See also: public int getScrollBarFadeDuration () Returns the duration of the scrollbar fading. Related XML attributes: android:scrollbarFadeDuration returns int the duration of scrollbar fading, in milliseconds int public end getScrollX () Return the scrolled left position of this view. This is the left edge of the displayed part displayed his vision. You don't need to draw any more pixels to the left, as these are outside the frame of your view on the screen. Returns int The left edge of the displayed part of your view, in pixels. int public end getScrollY () Return the top scrolled position of this view. This is the top edge of the displayed part of your view. You don't need to draw any pixels above it, as these are outside the frame of your view on the screen. Returns int The top edge of the displayed part of your view, in pixels. Public int getSolidColor () Undo this if your vision is known to always be drawn on top of a solid color background, and need to draw faded edges. Returning a non-zero color allows the display system to optimize the drawing of faded edges. If you return a non-zero color, the alpha should be set to 0xFF. Returns int The known solid color background for this view or 0 if the color can vary the public int getSourceLayoutResId () A view can be inflated from an XML layout. For such a view, this method returns the resource ID of the source layout. Returns int The layout feature id if this view was inflated from XML, otherwise Resources#ID_NULL. Public object getTag (int key) returns the tag associated with this view and the specified key. Key parameters int: The key that identifies the Tag Returns Object stored in this view as a tag, or null if not defined See also: setTag(int, Object)getTag() public int getTextAlignment () Return the resolved text alignment. Related XML attributes: Returns int the alignment of resolved text. 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 Value is TEXT_ALIGNMENT_INHERIT, TEXT_ALIGNMENT_GRAVITY, TEXT_ALIGNMENT_CENTER, TEXT_ALIGNMENT_TEXT_START, TEXT_ALIGNMENT_TEXT_END, TEXT_ALIGNMENT_VIEW_START, or TEXT_ALIGNMENT_VIEW_END public end int getTop () Top position of this view relative to its parent. Returns int The top of this view, in pixels. public ArrayList<View> getTouchables () Find and return all touchable views that are descendants of this view, possibly including this view if it is touchable itself. Returns ArrayList<View> A list of publicly sensitive views getTransitionAlpha () This property is intended only for use by the Fade transition, which animates it to produce a visual translucency that does not effect the side effect (or is affected by) the actual alpha property. This value is composed with the other alpha value (and the AlphaAnimation value, when this is present) to produce a final visual translucency result, which is what is passed to the DisplayList. Public String (getTransitionName) Returns the name of the View to be used to identify views in transitions. The names must be unique in the view hierarchy. This is null if the View does not received a name. Sequence returns The name used from the View to be used to identify visualizations in transitions or null if no name was given. public</View> </View> </View> getTranslationX () The horizontal location of this view relative to the left position. This position is post-layout, in addition to where the layout of the object placed it. Returns float The horizontal position of this view relative to the left position, in pixels. the public float (getTranslationY) The vertical location of this view relative to its top position. This position is post-layout, in addition to where the layout of the object placed it. Returns float The vertical position of this view relative to its top position, in pixels. the public float (getTranslationZ) The depth location of this view in relation to its elevation. Returns fluctuate The depth of this view in relation to its elevation. public long getUniqueDrawingId () Get the handle used for this visualization by the drawing system. Returns for a long time A long one that uniquely identifies the drawing component of this view See also: int public getVerticalFadingEdgeLength () Returns the size of the faded vertical edges used to indicate that more content in this view is visible. Related XML attributes: Returns int The pixel size of the faded vertical border or 0 if the faded vertical borders are not enabled for this view. public int getVerticalScrollbarWidth () Returns the width of the vertical scroll bar. Returns int The pixel width of the vertical scroll bar or 0 if there is no vertical scroll bar. public ViewTreeObserver (getViewTreeObserver) Returns the ViewTreeObserver to the hierarchy of this view. The view tree watcher can be used to receive notifications when global events, such as layout, happen. The returned ViewTreeObserver observer is not guaranteed to remain valid throughout the life of this view. If the caller of this method maintains a long-running reference to viewtreeobserver, it should always check the return value of the ViewTreeObserver#isAlive(). Returns the ViewTreeObserver The ViewTreeObserver to the hierarchy of this view. public end int getWidth () Return the width of your view. Returns int The width of your view, in pixels. Added at API level 16 Depreed in api level 30 int public getWindowSystemUiVisibility () This method has been deprecated at API level 30. SystemUiVisibility flags are depreced. Use windowinsetscontroller instead. Returns the current visibility of the system UI that is currently set for the entire window. This is the combination of the setSystemUiVisibility(int) values provided by all views of the window. public int getWindowVisibility () Returns the current visibility of the window to which this view is attached (gone, invisible, or visible). Returns int Returns the current visibility of the view window. The value is visible, invisible, or public empty gone to getWindowVisibleDisplayFrame (Rect outRect) Retrieve the overall size of the visible display in which the window to which it is attached has been positioned. This takes into account screen decorations above the window, for both cases where the window itself is being positioned inside them or the window is being placed under then and the covered insets are are are are so that the window positions its contents inside. In fact, this informs the available area where content can be placed and remains visible to users. This function requires an IPC back to the window manager to retrieve the requested information, so it should not be used in performance-critical code as a drawing. Rect outRect parameters: Filled with visible display frame. If the view is not attached to a window, this is simply the size of the raw display. public floats (getX) The visual position x of this view, in pixels. This equates to the translationX property plus the current left property. Returns float The visual position x of this view, in pixels. public float (getY) The visual position y of this view, in pixels. This equates to the translationY property plus the current top property. Returns float The visual position of this view, in pixels. public floats (getZ) The visual position z of this view, in pixels. This equates to the translationZ property plus the current elevation property. Returns float The visual position z of this view, in pixels. public boolean hasExplicitFodable () Returns true if this view is focusable or if it contains an achievable view for which has explicitfodible returns() is true. A reachable hasExplicitFocusable() is a vision whose parents do not block the focus of descendants. Only visible views for which getFocusable() would return FOCUSABLE are considered focusable. This method preserves the pre-Build.VERSION_CODES#O behavior of hasFocusable() in which only explicitly defined visualizations with focus will make this method return true. A vision set to FOCUSABLE_AUTO that resolves to focus will not be. True Boolean returns if the view is focusable or if the view contains a focal view, false public boolean hasFocus () Returns true if that view has focus on itself, or is the ancestor of the view that has focus. Returns boolean true if this view has or contains focus, false otherwise. public boolean hasNestedScrollingParent () Returns true if this view has a nested scrolling parent. The presence of a nested scrolling parent indicates that this view initiated a nested scroll and was accepted by an ancestor view further above the visualization hierarchy. Boolean returns if this view has a public nested scrolling parent hasOnClickListeners () Return if this view has an OnClickListener attached. Returns true if there is a listener, false if there is none. public boolean hasOnLongClickListeners () Return if this view has an OnLongClickListener attached. Returns true if there is a listener, false if there is none. public boolean hasOverlappingRendering () Returns if this View has overlapping content. This function, intended to be replaced by specific display types, is an optimization when alpha is defined in a view. If the overlap in a view with alpha < 1, this view is drawn to an offscreen buffer and then composed in place, which can be expensive. If the view has no overlap overlay 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 without background or an ImageView only with the image in the foreground. The default implementation return is true; subclasses should replace if they have cases that can be optimized. Note: The return value of this method is ignored if the forcedOverlappingRendering (Boolean) is called. Returns true Boolean if the content in this view can overlap, false otherwise. the public boolean hasTransientState () Indicates whether the view is currently tracking the transient state that the application should not worry about saving and restoring, but that the frame should take special note to preserve when possible. A transient state view cannot be trivially reboundfrom an external data source, such as an adapter linking item view in a list. This can be because the view is performing an animation, crawling the selection of user content or similar. Returns true boolean if the view has public boolean of the transient state hasWindowFocus () 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 boolean True if this view is in a window that currently has window focus. Public static visual inflate (context context, int resource, viewgroup root) inflate a view of an XML resource. This convenience method involves the LayoutInflater class, which provides a full range of options for display inflation. Context parameters context: The context object for your activity or application. int resource: The resource ID to inflate root ViewGroup: A view group that will be the parent. Used to properly inflate the parameters layout_*. Added in the Deprecated API level 28 in invalid public void (Dirty Rectify) This method has been deprecated at API level 28. Switching to accelerated hardware rendering in API 14 has reduced the importance of dirty rectangle. In API 21, the given rectangle is ignored entirely in favor of an internally calculated area. Because of this, customers are encouraged to just call it invalidate(). Mark the area defined by dirty as needing to be drawn. If the view is visible, onDraw (android.graphics.Canvas) will be called at some point in the future. This should be called a UI thread. To connect from a non-UI thread, call postInvalidate(). WARNING: In API 19 or below, this method can be destructive to dirty. Dirty Parameters Retule: The rectangle representing the boundaries of the dirty region added in api level 1 depreed in API level 28 empty public invalidated (int l, int t, int r, int b) This method was api level 28. Switching to accelerated hardware rendering in API 14 has reduced the importance of dirty rectangle. In API 21, the given rectangle is ignored entirely in favor of an internally calculated area. Internally. customers are encouraged to just call it invalidate(). Mark the area defined by the ret (l,t,r,b) as needing to be drawn. The coordinates of the dirty retalire are relative to the view. If the view is visible, onDraw (android.graphics.Canvas) will be called at some point in the future. This should be called a UI thread. To connect from a non-UI thread, call postInvalidate(). Parameters l int: the left position of the dirty region t int: the top position of the dirty region r int: the right position of the dirty region b int: the lower position of the invalidated public empty dirty regionSays (Drawable drawable) Invalidates the specified Drawable. Drawable drawable parameters: the drawable to invalidate this value cannot be null. Boolean public isAccessibilityFocused () Returns whether this view is focused on accessibility. Returns boolean True if this view is focused on accessibility. public boolean (isAccessibilityHeading) Understands whether this view is a title for accessibility purposes. Related XML attributes: android:accessibilityHeading Returns true Boolean if the view is an otherwise false, false title. the public boolean isActivated () Indicates the activation state of this view. Returns true boolean if the display is turned on, otherwise false public boolean isAttachedToWindow () Returns true if this view is currently attached to a window. the boolean isClickable audience () Indicates whether this view reacts to click events or not. Related XML attributes: Returns true Boolean if the view is clickable, false otherwise see also: Boolean audience is Dirty () True if this view has changed since the last time it was drawn. Returns booleanThe dirty state of this view. Added to the Deprecated Api level 1 in the boolean public level 28 API isDrawingCacheEnabled () This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for ui reports or unit test is recommended. Indicates whether the drawing cache is enabled for this view. Returns true boolean if drawing cache is enabled See also: setDrawingCacheEnabled(boolean)getDrawing this vision. The interpretation of the enabled state varies by subclass. Returns Boolean True if this view is turned on, false otherwise. the public end boolean (isFocusable) returns if this View is currently able to have focus. Returns boolean true if this view can have focus, or false otherwise. the public end boolean isFocusableInTouchMode () When a view is focusable, it may not want to focus when in touch mode. For example, a button would like to focus when the user is browsing through a D-pad so that the user can click on it, but once the user starts tapping the screen, the button should not have focus related XML attributes: android:focusableInTouchMode Returns booleanAnd the preview is focusable in touch mode. public boolean isFocused () Returns true if this view has focus Returns boolean True if this view has focus, false otherwise. the public final boolean isFocusedByDefault () returns whether this view should be focused when the focus is restored to the view hierarchy that contains that view. Focus is restored to a visualization hierarchy when the root of the hierarchy is added to a window or serves as a cluster navigation target. Related XML attributes: True Boolean returns if this view is the default-focus, false display otherwise see also: the public boolean isHardwareAccelerated () Indicates whether this view is connected to an accelerated hardware window or not. Even if this method returns true, it does not mean that all calls to draw (android.graphics.Canvas) will be made with a hardware-accelerated screen. For example, if this view is drawn on an offscreen Bitmap and its window is hardware-accelerated, Canvas.isHardwareAccelerated() is likely to return false, and this method returns true. Returns boolean True if the view is attached to a window and the window is accelerated by the hardware; false in any other case. the public boolean isHorizontalScrollBarEnabled () Indicate whether the horizontal scroll bar should be drawn or not. The scroll bar is not drawn by default. True Boolean returns if the horizontal scroll bar should be painted, false otherwise see also: setHorizontalScrollBarEnabled (boolean) boolean public isImportantForAccessibility () Calculates whether this view should be exposed for accessibility. In general, interactive visualizations or information are exposed while visualizations that serve only as containers are hidden. If an ancestor of this view has IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS, this method returns false. Otherwise, the value is computed according to the value getImportantForAccessibility() of the view: IMPORTANT_FOR_ACCESSIBILITY_NO or IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS, return IMPORTANT_FOR_ACCESSIBILITY_YES, return true, return true if the preview satisfies any of the following: It has an accessibility pane title, see setAcessibilityPaneTitle(CharSequence) returns boolean if the preview is exposed for accessibility. See also: also: public end boolean isImportantForAutofill () Suggests to the Android System whether the AssistStructure.ViewNode associated with this view is considered important for autocomplete purposes. Generally speaking, a view is important for autocomplete if: The preview can be automatically populated by an AutofillService. Display content can help an AutofillService determine how other visualizations can be filled automatically. For example, preview containers should typically return false for performance reasons (since important information is provided by their children), but if their properties have relevant information (for example, a resource id called credentials, it should return true. On the other hand, visualizations representing labels or editable fields should typically return true, but in some cases they can return false (for example, if they are part of a Captcha engine). The value returned by this method depends on the value returned by getImportantForAutofill(): When a view is considered important for autocomplete: The preview can automatically trigger an autocomplete request when focused. The contents of the view are included in the Display Structure used in an autocomplete request. On the other hand, when a view is considered not important for autocomplete: Boolean returns if the view is considered important for autocomplete. public boolean (isInEditMode) Indicates whether this view is currently in edit mode. A view is usually in edit mode when displayed within a developer tool. For example, if this view is being drawn by a visual ui constructor, this method should return true. Subclasses should check the return value of this method to provide different behaviors if their normal behavior can interfere with the host environment. For example: the class generates a segment in its constructor, the drawing code depends on device-specific features, and so on. This method is usually checked in the drawing code of custom widgets. Returns boolean True if this view is in edit mode, false otherwise. public boolean isInLayout () Returns whether the display hierarchy is currently going through a layout pass. This information is useful to avoid situations such as request callLayout() during a layout pass. Boolean returns if the display hierarchy is currently passing a boolean (isInTouchMode) public layout pass returns whether the device is currently in touch mode. The touch mode is entered when the user begins to interact with the device by touch, and affects several things as if the focus is always visible to the user. Boolean returns if the device in touch mode. the public end boolean (isKeyboardNavigationCluster) returns whether this view is a root of a keyboard navigation cluster. Related XML attributes: android:keyboardNavigationCluster Returns boolean True if this visualization is a root of a cluster or otherwise false. public boolean isLaidOut () Returns true if this view has gone through at least one layout since it was last attached or separated from a window. boolean public isLayoutDirectionResolved () () boolean true if the layout direction has been resolved. the public boolean isLayoutRequested () Indicates whether or not the layout of this view will be requested during the next hierarchy layout pass. Returns true Boolean if the layout will be forced during the next boolean public layout pass (NestedScrollingEnabled) Returns true if nested scrolling is enabled for this view. If nested scrolling is enabled and this implementation of the View to Support class, this view acts as a nested scrolling child view when applicable, forwarding data about the in-progress parchment operation to a compatible, cooperative nested scrolling parent. True boolean returns if nested scrolling is enabled See also: setNestedScrollingEnabled (boolean) public boolean isOpaque () Indicates whether this view is opaque. An opaque display ensures that it will draw all pixels superimposed on its boundaries using a fully opaque color. View subclasses should override this method whenever possible to indicate whether an instance is opaque. Opaque views are handled especially by the View hierarchy, possibly allowing it to perform optimizations during invalidated/hold passes. Returns boolean true if this view is guaranteed to be fully opaque, otherwise false. public boolean isPivotSet () Returns whether or not a pivot was defined by a call to define PivotX(float) or setPivotY (float). If no pivot has been defined, the pivot is the center of the view. True boolean returns if a pivot has been set, false if the default pivot is being used in public, the boolean isSaveFromParentEnabled () indicates whether the entire hierarchy under this view saves its state when a state save traversal occurs from its parent. The pattern is true; if false, these views will not be saved unless saveHierarchyState (android.util.SparseArray) is called directly in this view. Boolean returnsTrue returns if the save view state of the parent is enabled, another false thing. See also: setSaveFromParentEnabled (boolean) public boolean isScrollContainer () Indicates whether this view is one of the scrollable containers in your window. Related XML attributes: android:isScrollContainer Returns Boolean if this view is one of the sets of scrollable containers in its public window boolean isScrollbarFadingEnabled () Returns true if the scroll bars disappear when this view is not scrolling Related XML Attributes: Returns true boolean if the scroll bar is enabled for the chosen boolean isS audience () Indicates the selection state of this view. Returns true boolean if the view is selected, false public end boolean isShowingLayoutBounds () Returns true when the view is attached and the system developer setting to show the layout boundaries is or false otherwise. public boolean isShown () Returns the visibility of this view and all its ancestors Returns Boolean True if this view and all its ancestors are visible boolean public isAlignmentResolved () Returns true Boolean if the text alignment is resolved. boolean public public () Returns true Boolean if the text direction is resolved. Public boolean isVerticalScrollBarEnabled () Indicate whether the vertical scroll bar should be drawn or not. The scroll bar is not drawn by default. True boolean returns if the vertical scroll bar should be painted, false otherwise see also: setVerticalScrollEnabled (boolean) public boolean isVisibleToUserForAutofill (int virtualId) calculates whether this virtual autofill view is visible to the user. Note: By default, it returns true, but visualizations that provide a virtual hierarchy view must override it. Returns boolean if the preview is visible on the screen. NavigationClusterSearch public display keyboard (View currentCluster, direction int) Find the nearest keyboard navigation cluster in the specified direction. That doesn't focus on this cluster. Returns view The nearest keyboard navigation cluster in the specified direction or null if none can be found public empty layout (int l, int t, int r, int b) Assign a size and position to a view and to all its descendants This is the second phase of the layout engine. (The first is measurement). At this stage, each parent calls the layout of all their children to position them. This is usually done using the child's measurements that were stored in the measurement pass(). Derived classes should not override this method. Derived lessons with children should replace onLayout. In this method, they must call layout on each of their children. Parameters l int: Left position, relative to parent t int: Top position, relative to parent r int: Right position, relative to parent b int: Lower position, relative to the parent public end void measure (int widthMeasureSpec, int heightMeasureSpec) This is called to find out how large a view should be. Parents provide constraint information on the width and height parameters. The actual measurement work of a view is performed in onMeasure (int, int), called by this method. Therefore, only the onMeasure (int, int) can and should be replaced by subclasses. Parameters widthMeasureSpec int: Horizontal space requirements as imposed by parent heightMeasureSpec int: Vertical space requirements as enforced by the parent See also: offset of the public voidLeftAndRight (int offset) Compensate for the horizontal location of this visualization by the specified amount of pixels. Int cleared parameters: The number of pixels to compensate for viewing by public empty deviationTopAndBottom (int offset) Offset the vertical location of this view by the specified number of pixels. Int plywood parameters: The number of pixels to compensate for the display by public windows onApplyWindowInsets (Insets WindowInsets) called when the view should apply WindowInsets according to its internal policy. This method should be replaced by views that apply a different policy or beyond the default behavior. Clients who want to force a display subtree to apply insets should call dispatchApplyWindowInsets (android.view.WindowInsets). Customers can provide a to a view. If one is set, it is called during dispatch instead of this method. The listener can optionally call this method from its own implementation if it wants to apply the default policy of view insets in addition to its own. The implementations of this method should return the unchanged insets parameter or a new cloned WindowInsets from the insets provided with any consumed insets that this view applies to. This allows new input types added in future versions of the platform to go through existing implementations unchanged without being misconsumed. By default, if the property fits into a view, the Unhealthy property consumes the insets of the system window and applies them as padding for the view. WindowsInsets: Insets parameter insets to apply insets returns The insets provided with any applied insets consumed public void in the CancelPendingPendingEvents () called as a result of a call to cancelPendingInputEvents() in this view or a parent view. This method is responsible for removing any high-level pending 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(java.lang.Runnable), postDelayed(java.lang.Runnable, long), or Handler should override this method, call super.onCancelPendingPendingEvents() and remove these callbacks as appropriate. public boolean onCapturedPointerEvent (motionEvent event) implement this method to handle pointer events captured MotionEvent event parameters: The captured pointer event. True boolean returns if the event was handled, false otherwise. See also: boolean audience onCheckIsTextEditor () Make sure that 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 in CreateInputConnection (android.view.inputmethod.EditorInfo) to return true if a call in this method would return a non-null InputConnection, and they are actually a first-class editor that the user would typically start typing when entering a window containing its view. The default implementation always returns false. This does not mean that your onCreateInputConnection (android.view.inputmethod.EditorInfo) will not be called or the user will not be able to perform edits otherwise in your view; it's just a hint to the system that this is not the main purpose of this view. Returns boolean returns true if this view is a text editor, another false. Public inputconnection onCreateInputConnection (EditorInfo outAttrs) Create a new Input Connection for an InputMethod to interact with the visualization. The default implementation returns null because it does not support input methods. You can replace to implement this support. This is only required for visualizations that take focus and text input. When implementing this, you probably also want to implement in the CheckIsTextEditor() to indicate that you will return a non-null InputConnection. Also, be careful to fill in the object correctly and in its entirety, so that the connected IME can rely on its values. For example, editorinfo.initialSelStart and EditorInfo.initialSelEnd members must be filled with the correct cursor position for IMEs to function correctly with your application. Parameters outAttrs EditorInfo: Fill in attribute information about the connection. public boolean onDragEvent (DragEvent event) Handles drag events sent by the system after a call to startDragAndDrop(). When the system calls this method, it passes through a DragEvent object. A call to DragEvent.getAction() returns one of the action type constants defined in DragEvent. The method uses them to determine what is happening in the drag-and-drop operation. DragEvent parameter events: The DragEvent sent by the system. The DragEvent.getAction() method returns a constant type of action defined in DragEvent, indicating the type of drag event represented by this object. Returns true boolean if the method was successful, otherwise false. The method must return true in response to a type of DragEvent.ACTION_DRAG_STARTED to receive drag events for the current operation. The method should also return true in response to a type of DragEvent.ACTION_DROP if it consumed the drop, or false if it did not. For all other events, the return value is ignored. public void on drawforeground (screen screen) Draw any foreground content for this view. Foreground content can consist of scroll bars, a foreground, or other view-specific decorations. The foreground is drawn on top of the main display content. Parameters screen: Screen to draw boolean in public in TheFilterTouchEventForSecurity (MotionEvent event) Filter the touch event to apply security policies. MotionEvent event parameters: The motion event to filter. True boolean returns if the event should be dispatched, false if the event should be withdrawn. See also: get FilterTouchesWhenObscured() public boolean in GenericMotionEvent (MotionEvent event) Implement this method to handle generic motion events. Generic motion events describe joystick movements, mouse paito, track pad touches, scroll wheel movements, and other input events. The MotionEvent#getSource() of the motion event specifies the input class 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 the inputdevice#source class are SOURCE_CLASS_POINTER delivered to view under the pointer. All other generic motion events are delivered to focal vision. boolean inGenericMotionEvent (MotionEvent event) { if (event.isFromSource(InputDevice.SOURCE_CLASS_JOYSTICK)) { if (event.getAction() == { // process joystick movement... true return; } } if (event.isFromSource(InputDevice.SOURCE_CLASS_POINTER)) { switch (event.getAction()) { case MotionEvent.ACTION_HOVER_MOVE: // process hover mouse true return; case MotionEvent.ACTION_SCROLL: // process the movement of the scroll wheel... true return; } return super.onGenericMotionEvent(event); } MotionEvent parameter events: The generic motion event being processed. True boolean returns if the event was handled, false otherwise. public boolean onHoverEvent (MotionEvent event) implement this method to handle hovering events. This method is called whenever a pointer is hovering within, over, or outside the boundaries of a view, and the view is not currently being touched. Hover events are represented as pointer events with MotionEvent#ACTION_HOVER_ENTER, MotionEvent#ACTION_HOVER_MOVE, or MotionEvent#ACTION_HOVER_EXIT. The view receives a hover event with MotionEvent#ACTION_HOVER_ENTER when the pointer enters the boundaries of the view. The view receives a hover event with MotionEvent#ACTION_HOVER_MOVE action when the pointer has already entered the boundaries of the view and moved. The preview receives a hover event with MotionEvent#ACTION_HOVER_EXIT action when the pointer has left the boundaries of the view or when the pointer is about to descend due to a user click, touch, or similar action that causes the view to be touched. The view must implement this method to return true to indicate that it is dealing with the hover ing event, such as changing its draw state. The default implementation calls setHovered(boolean) to update the hovered view state of the view when a hover output event enters or hovers is received, if the view is enabled and clicked. The default implementation also sends accessibility events per hover. MotionEvent parameter events: The motion event that describes the hover. True boolean returns if the preview handled the hover event. See also: isHovered()setHovered(boolean)onHoverChanged(boolean) empty public onInitializeAccessibilityEvent (AccessibilityEvent event) Initializes an Accessibility Event with information about this view that is the source of the event. In other words, the origin of an accessibility event is the view whose change of state triggered the dismissal of the event. Example: Set the password property of an event, in addition to properties set by super implementation: public void onInitializeAccessibilityEvent (AccessibilityEvent Event) { super.onInitializeAccessibilityEvent (event); event.setPassword(true); } If an Accessibility Additional has been specified through the AccessibilityDelegate call set (android.view.View.AccessibilityDelegate) your AccessibilityDelegate#onInitializeAccessibilityEvent (View, AccessibilityEvent) is responsible for

handling that call. Note: Always call the super deployment before adding information to the event if the default implementation has basic information to add. If you override this method, you should call up to the of the superclass. AccessibilityEvent event parameters: The event to initialize. empty public onInitializeAccessibilityNodeInfo (Accessibility InformationNodeInfo) Initializes an AccessibilityNodeInfo with information about it The base implementation sets: AccessibilityNodeInfo#setParent(View), AccessibilityNodeInfo#setBoundsInParent(Rect), AccessibilityNodeInfo#setBoundsInScreen(Retct), AccessibilityNodeInfo#setPackageName(CharSequence), AccessibilityNodeInfo#setClassName(CharSequence), AccessibilityNodeInfo#setContentDescription(CharSequence), AccessibilityNodeInfo#setEn (boo AccessibilityNodeInfo#setClickable(boolean), AccessibilityNodeInfo#setFocusable(boolean), AccessibilityNodeInfo#setFo (boolean), AccessibilityNodeInfo #setLongClickable(boolean), AccessibilityNodeInfo#setSelected(boolean), AccessibilityNodeInfo#setContextClickable(boolean) Subclasses should replace this method, call the super implementation, and define additional attributes. If an Accessibility Addition has been specified through the AccessibilityDelegate call set (android.view.View.AccessibilityDelegate) your AccessibilityDelegate#onInitializeAccessibilityNodeInfo (View, AccessibilityNodeInfo) is responsible for handling that call. If you override this method, you must call up the superclass implementation. AccessibilityNodeInfo info parameters: The instance to initialize. public boolean onKeyDown (int keyCode, keyEvent event) Default implementation of KeyEvent.Callback#onKeyDown (int, KeyEvent): Run the display press when KeyEvent#KEYCODE_DPAD_CENTER or KeyEvent#KEYCODE_ENTER is launched, if the view is enabled and clickable. Key presses on software keyboards generally DO NOT trigger this listener, although some may choose to do so in some situations. Don't rely on this to capture software keystrokes. KeyCode parameters int: A key code that represents the pressed button, from the keyEvent of the KeyEvent event: the KeyEvent object that sets the action of the Boolean button 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 noKeyLongPress (int keyCode, keyEvent event) Default implementation of KeyEvent.Callback#onKeyLongPress (int, KeyEvent): always returns false (does not handle the event). Key presses on software keyboards generally DO NOT trigger this listener, although some may choose to do so in some situations. Don't rely on this to capture software keystrokes. KeyCode parameters int: The value in event.getKeyCode(). KeyEvent event: Description of the key event. Boolean 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 noKeyMultiple (int keyCode, int repeatCount, keyEvent event) Default implementation of KeyEvent.Callback#onKeyMultiple (int, int, KeyEvent): always returns false (does not handle the event). Key presses on software keyboards generally DO NOT trigger this listener, although some may choose to do so in some situations. Don't trust that. capture software key presses. KeyCode int parameters: A key code that represents the keybutton of the KeyEvent. repeat Account int: The number of times the was done. KeyEvent event: The KeyEvent object that defines the button action. Boolean 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) handles a key event before being 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 to deal with the BACK key to update the application UI instead of allowing the IME to see it and close it. KeyCode parameters int: The value in event.getKeyCode(). KeyEvent event: Description of the key event. Boolean 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 noKeyShortcut (int keyCode, keyEvent event) called in the focused view when a key shortcut event is not handled. Override this method to implement local key shortcuts to view. Key shortcuts can also be implemented by setting the MenuItem#setShortcut (char, char) property of menu items. KeyCode parameters int: The value in event.getKeyCode(). KeyEvent event: Description of the key event. Boolean 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) Default keyevent implementation.Callback#onKeyUp(int, KeyEvent): click the view when KeyEvent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER, or KeyEvent#KEYCODE_SPACE is released. Key presses on software keyboards generally DO NOT trigger this listener, although some may choose to do so in some situations. Don't rely on this to capture software keystrokes. KeyCode int parameters: A key code that represents the keybutton of the KeyEvent. KeyEvent event: The KeyEvent object that defines the button action. Boolean 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 in PointerCaptureChange (boolean hasCapture) called when the window has just acquired or lost the pointer capture. If you override this method, you must call up the superclass implementation. Parameters has Boolean Cap: True if the preview now has pointerCapture, false otherwise. public void onPopulateAccessibilityEvent (AccessibilityEvent event) called the dispatchPopulateAccessibilityEvent (android.view.accessibility.AccessibilityEvent) giving this View a chance to fill the accessibility event with its text content. Although this method is free to modify event attributes other than text content, this should typically be performed in the Example: Add sequence of formatted dates to an accessibility event, in addition to text added by super implementation: Public void event) { super.onPopulateAccessibilityEvent (event); final int flags = DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY; String selectedDateUtterance = DateUtils.formatDateTime(mContext, mCurrentDate.getTimeInMillis(), cues; event.getText().add(selectedDateUtterance); } If an Accessibility Delegate has been specified through the AccessibilityDelegate call set (android.view.View.AccessibilityDelegate) your AccessibilityDelegate#onPopulateAccessibilityEvent (View, AccessibilityEvent) is responsible for handling that call. Note: Always call the super deployment before adding information to the event if the default implementation has basic information to add. If you override this method, you must call up the superclass implementation. AccessibilityEvent event parameters: The accessibility event that you fill. public void in the ProvideAutofillStructure (Display structure, flags int) Populates a Display Structure to completely fill out an autocomplete request. The structure must contain at least the following properties: It is also recommended to set the following properties - the more properties the structure has, the greater the chances of an AutofillService correctly using the structure: Autofill Tips (ViewStructure#setAutofillHints(String[]). Autofill options (ViewStructure#setAutofillOptions(CharSequence[])) when the view can only be filled with predefined values (typically used when the autocomplete type is AUTOFILL_TYPE_LIST). Resource ID (ViewStructure#setId(int, String, String, String)). Class name (ViewStructure#setClassName(String)). Description of the content (ViewStructure#setContentDescription(CharSequence)). Visual properties such as visibility (ViewStructure#setVisibility(int)), dimensions (ViewStructure#setDimens(int, int, int, int, int), and opacity (ViewStructure#setOpaque(boolean)). For visualizations that represent text fields, text properties such as text itself (ViewStructure#setText(CharSequence)), text hints (ViewStructure#setHint(CharSequence), input type (ViewStructure#setInputType(int)), For visualizations representing HTML nodes, your web domain (ViewStructure#setWebDomain(String)), and HTML properties ((ViewStructure#setHtmlInfo(android.view.ViewStructure.HtmlInfo)). The default implementation of this method already defines most of these properties based on related display methods (for example, the autocomplete id is set using getAutofillId(), the autofill type set using getAutofillType(), etc.), and visualizations in the standard Android widget library also replace it to set its relevant properties (for example, TextView already sets text properties), so it is recommended to only replace this method (and call when: Note: The left and top values defined in the predecessor ViewStructure#setDimens(int, int, int, int, int) must be relative to the next view of the Predecessor ViewGroup#isImantForAutofill() included in the structure. Visualizations support the Autofill Framework primarily by: Providing metadata by defining what vision means and can be filled automatically. Notify the Android System when the display value has changed by calling AutofillManager#notifyValueChanged(View). Implementing methods that automatically filter the view. This method is responsible for the first; autofill (android.view.autofill.AutofillValue) is responsible for the latter. See also: AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS public void in the ProvideAutofillVirtualStructure (Preview Structure, flags int) Populaa a Display Structure containing virtual children to fill out an autocomplete request. This method should be used when the visualization manages a virtual structure under this view. For example, a view that draws input fields using draw (android.graphics.Canvas). When implementing this method, the subclasses must follow the rules below: Add virtual children by linking to the ViewStructure#newChild(int) or ViewStructure#asyncNewChild(int) methods, where the id is a single document identifying the children in the virtual structure. The child hierarchy can have multiple levels if necessary, but the ideal is to exclude intermediate levels that are irrelevant to autocomplete; that would improve autocomplete performance. Also implement autofill (android.util.SparseArray) to automatically filter virtual children. Set the autofill properties of the child structure as defined by the onProvideAutofillStructure (android.view.ViewStructure, int), using the ViewStructure#setAutofillId (AutofillId, int) to set your autocomplete id. Call AutofillManager.notifyViewEntered (View, int, Retct) and/or AutofillManager.notifyViewExited(View, int) when the focused virtual child has changed. Override isVisibleToUserForAutofill(int) to allow the platform to ask whether a particular virtual view is visible to the user in order to support the activation of salvation when all views of interest disappear. Call AutofillManager.notifyValueChanged (View, int, AutofillValue) when the value of a virtual child has changed. Call AutofillManager.notifyViewVisibilityChanged (View, int, boolean) when the visibility of a virtual child has changed. Call AutofillManager.notifyViewClicked when a virtual child is clicked. Call AutofillManager#commit() when the autocomplete context of the display structure has changed and the current context must be compromised (for example, when the user touched a SEND button on an HTML page). Call AutofillManager#cancel() when the autocomplete context of the display structure has changed and the current context should be canceled (for example, when the user touched a CANCEL button on an HTML page). Provide ways for users to request autocomplete by manually calling AutofillManager#requestAutofill(View, int, Rect). The left and top defined in ViewStructure#setDimens(int, int, int, int, int) must be relative to the next view of the ViewGroup#isImportantForAutofill() predecessor view included in the structure. Visualizations with virtual children support the Autofill Framework primarily by: Providing the metadata that defines what virtual children mean and how they can be filled automatically. Implementing methods that automatically filter virtual virtual This method is responsible for the first; autofill (android.util.SparseArray) is responsible for the latter. Parameter structure Visualization structure: Fill in with virtual children's data for autocomplete purposes. Flags Int: Optional flags. See also: AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS public void in the ProvideContentCaptureStructure (display structure structure structure, flags int) populates a View Structure for capturing content. This method is called after a view that is eligible for content capture (for example, if it is ImportantForAutofill(), an intelligence service is enabled for the user and the activity that provides the view is enabled for content capture) is established and is visible. The populated structure is then passed to the service through the ContentCaptureSession#notifyViewAppeared(ViewStructure). Note: Visualizations that manage a virtual structure under this view should populate only the node representing this view and immediately return, then asynchronously report (not necessarily on the UI segment) when child nodes appear, disappear, or have their text changed by calling ContentCaptureSession#notifyViewAppeared(ViewStructure), ContentCaptureSession#notifyViewDisappeared (AutofillId) respectively. The framework for a child must be created using ContentCaptureSession#newVirtualViewStructure(AutofillId, long), and autofillId for a child can be obtained through childStructure.getAutofillId() or ContentCaptureSession#newAutofillId (AutofillId, long). When the virtual view hierarchy represents a Web page, you should also: Note: The following methods of the framework are ignored: Parameter structure Preview structure: This value cannot be null. public empty int flags in the ProvideStructure (View Structure) called when the assistance structure is being retrieved from a view as part of Activity.onProvideAssistData. Parameter structure Visualization structure: Fill in with structured visualization data. The default implementation populates all the data that can be inferred from the view itself. Public void onProvideVirtualStructure (View Structure) Called when the assistance structure is being retrieved from a view as part of Activity.onProvideAssistData to generate additional virtual structure under this view. The defaullt implementation uses getAccessibilityNodeProvider() to try to generate this from the virtual accessibility nodes of the view, if any. You can replace this with a more optimal implementation by providing this data. Parameter structure View Formation of the public pointerIcon in The ResolvePointerIcon (MotionEvent event, int pointerIndex) Returns the pointer icon for the motion event or null if it does not specify the icon. The default implementation doesn't care about the types events, but some subclasses may use it (such as WebViews). public empty onRtlPropertiesChanged (int layoutDirection) called when any RTL property (layout direction or text direction or text alignment) has changed. Subclasses Subclasses override this method to take care of cached information that depends on the resolved layout direction or to inform the visualizations of children inheriting their layout direction. The default implementation does nothing. See also: LAYOUT_DIRECTION_LTRLAYOUT_DIRECTION_RTL empty public in ScreenStateChanged (intState screen) This method is called whenever the screen state this view is attached to changes. A state change will usually occur when the screen turns on or off (whether it happens automatically or the user does so manually.) the public boolean in the TouchEvent (MotionEvent event) implement this method to handle touch screen motion events. If this method is used to detect click actions, it is recommended that actions be performed by implementing and calling performClick(). This will ensure consistent system behavior, including: MotionEvent: The motion event event parameters. True boolean returns if the event was handled, false otherwise. public boolean onTrackballEvent (motionEvent event) Implement this method to handle trackball motion events. The relative movement of the trackball since the last event can be retrieved with MotionEvent#getX and MotionEvent#getY. These are normalized so that a movement of 1 corresponds to the user pressing a DPAD key (so they will often be fractionated values, representing the finer motion information available from a trackball). MotionEvent event parameters: The motion event. True boolean returns if the event was handled, false otherwise. Public void onVisibilityAggregated (boolean isVisible) called when user visibility of this view is potentially affected by a change in this view itself, an ancestral view, or the window to which that view is attached. If you override this method, you must call up the superclass implementation. Parameters are visible boolean: True if this view and all its ancestors are VISIBLE and the window of this view is also visible public void in TheWindowFocusChanged (boolean hasWindowFocus) called when the window containing this view gains or loses focus. Note that this is separate from the view focus: To receive key events, both your view and your window must have focus. If a window appears on top of its one that requires input focus, then its own window will lose focus, but the display focus remains unchanged. Parameters has Boolean WindowFocus: True if the window that contains this view now has focus, otherwise false. runs boolean public () Call the OnClickListener of this view if set. Performs all normal actions associated with the click: report accessibility event, play a sound, etc. True boolean returns there was an assigned OnClickListener that was called, false otherwise returned. runs public booleanoContextClick (float x, y) Call the OnContextClickListener of this view if set. Parameters x fluctuation: the x coordinate of the click y float context: the y coordinate of the click Returns boolean True context if there was an assigned OnContextClickListener assigned consumed the event, false otherwise. runs public boolean (ContextClick) Connect to the OnContextClickListener of this view if set. True boolean returns if there was an Assigned OnContextClickListener that consumed the event, otherwise false. public boolean performHapticFeedback (int feedbackConstant) BZZZTT!! 1! Provide the user with a plyth feedback for this view. The framework will provide home feedback for some embedded actions, such as long presses, but you may want to provide feedback for your own widget. Feedback will only be performed if isHapticFeedbackEnabled() is true. Public boolean performanceLongClick (float x, float y) Calls the OnLongClickListener from this view, if set. Invokes the context menu if the OnLongClickListener does not consume the event, anchoring it to a coordinate (x,y). Parameters x float: x anchor ing event coordinate, or Float#NaN to disable anchoring y float: y anchor ing event coordinate, or Float#NaN to disable anchoring Returns true Boolean if one of the above receivers consumed the event, otherwise public boolean performLongClick () Calls this view onLongClickListener, if set. Invokes the context menu if the OnLongClickListener does not consume the event. Returns boolean true if one of the above receivers consumed the event, false empty public playSoundEffect (int soundConstant) Play a sound effect for this view. The framework will play sound effects for some embedded actions, such as clicking, but you may want to play those effects in your widget, for example, for internal navigation. The sound effect will only play if the sound effects are activated by the user, and isSoundEffectsEnabled() is true. Boolean public post (runnable action) causes Runnable to be added to the message queue. The runnable will run on the UI thread. Runnable action parameters: The Runnable that will run. Boolean returns returns true if Runnable has been successfully placed in the message queue. Returns false about failure, usually because the looper processing the message queue is exiting. See also: postDelayed(Runnable, long)removeCallbacks (Runnable) postaalvoo publicDelayed (Runnable action, long delayMillis) Causes Runnable to be added to the message queue, to run after the specified time. The runnable will run on the UI thread. Runnable action parameters: The Runnable that will run. delaySound: The delay (in milliseconds) until Runnable runs. True Boolean returns if Runnable was successfully placed in the message queue. Returns false about failure, usually because the looper processing the message queue is exiting. Note that a true result does not mean that Runnable will be processed -- if the looper stops before the message delivery time the message will be withdrawn. See also: post(Runnable)removeCallbacks(Runnable) empty public postInvalidate () Cause Cause invalidate to happen in a subsequent loop through the event loop. Use it to invalidate the display of a non-UI thread. This method can only be invoked from outside the UI thread when this view is attached to a window. See also: invalidate()postInvalidateDelayed(long) empty public postInvalidate (int left, int top, int right, int bottom) Cause an invalidated specified area to happen in a subsequent loop through the event loop. Use it to invalidate the display of a non-UI thread. This method can only be invoked from outside the UI thread when this view is attached to a window. Parameters left int: The left coordinate of the rectangle to invalidate. Top int: The top coordinate of the rectangle to invalidate. right int: The right coordinate of the rectangle to invalidate. bottom int: The bottom coordinate of the rectangle to invalidate. Delayed public void (long delayMilliseconds, int left, int top, int right, int bottom) will cause an invalidated specified area to happen in a subsequent loop through the event loop. Waits for the specified time. This method can only be invoked from outside the UI thread when this view is attached to a window. Parameters are long delayed: the duration in milliseconds to delay invalidation by the left int: The left coordinate of the rectangle to invalidate. Top int: The top coordinate of the rectangle to invalidate. right int: The right coordinate of the rectangle to invalidate. bottom int: The bottom coordinate of the rectangle to invalidate. Public empty postInvalidatedDelayed (long delayMilliseconds) cause an invalidated happen in a subsequent loop through the event loop. Waits for the specified time. This method can only be invoked from outside the UI thread when this view is attached to a window. Parameters are delayedLong Mugundos: the duration in milliseconds to delay invalidation by View also: invalidate()postInvalidate() postInvalidatePublic void (left int, int superior, int right, int lower) Cause an invalidated of the specified area to happen in the next animation time step, typically the next display frame. This method can only be invoked from outside the UI thread when this view is attached to a window. Parameters left int: The left coordinate of the rectangle to invalidate. Top int: The top coordinate of the rectangle to invalidate. right int: The right coordinate of the rectangle to invalidate. bottom int: The bottom coordinate of the rectangle to invalidate. See also: invalidate (int, int, int, int)invalidate(Ret) public void postInvalidateOnAnimation () Cause an invalidated happen in the next animation time step, typically the next display frame. This method can only be invoked from outside the ui when this view is attached to a window. public empty postOnAnimationDelayed (Runnable action, long delayMillis) causes Runnable to run in the next animation time step, after the specified amount of time eats. O O will run on the UI segment. Runnable action parameters: The Runnable that will run. delaySound: The delay (in milliseconds) until Runnable runs. See also: postOnAnimation(Runnable)removeCallbacks (Runnable) public void refreshDrawableState () Call this to force a view to update its drawable state. This will cause DrawableStateChanged to be called to this view. Opinions interested in the new state should call getDrawableState. See also: drawableStateChanged()getDrawableState() empty public removeOnLayoutChangeListener (View.OnLayoutChangeListener listener) Remove a listener for layout changes. Listener parameters View.OnLayoutChangeListener: The listener for layout boundaries changes. Public End RequestFocus (direction int) Call this to try to focus on a specific view or one of your descendants and give you a hint about which direction the focus is going. A view will not really focus if it is not focusable (isFocusable() falsely returns), or if it is focusable and is not focusable in touch mode (isFocusableInTouchMode()) while the device is in touch mode. See also focusSearch(int), which is what you call to say you have focus, and you want your parent to look for the next one. This equates to the CallFocus request (int, android.graphics.Rect) with null set for the previously focused rectangle. Int direction parameters: One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT boolean returns if this view or one of their descendants really focused. Public end requestFocus () Call this to try to focus on a specific view or one of its descendants. A view will not really focus if it is not focused (isFocusable() returns false), or if it cannot be focused due to other conditions (not focusable in touch mode (isFocusableInTouchMode()) while the device is in touch mode, not visible, not activated, or has no size). See also focusSearch(int), which is what you call to say you have focus, and you want your parent to look for the next one. This equates to the CallFocus request (int, android.graphics.Rect) with FOCUS_DOWN null and null arguments. Boolean returns if this view or one of his descendants really focused. Public Boolean RequestFocus (direction int, Rect previouslyFocusedRect) Call this to try to focus on a specific view or one of its descendants and give you tips on the direction and a specific rectangle from where the focus comes from. The rectangle can help give you a bigger view of where the focus is coming from and therefore where to show the selection or change the focus internally. A view will not really focus if it is not focusable (isFocusable() falsely returns), or if it is focusable and is not focusable in touch mode (isFocusableInTouchMode()) while the device is in touch mode. A View will not focus if it is not visible. View will not focus one of your parents has equal to ViewGroup#FOCUS_BLOCK_DESCENDANTS. See also focusSearch(int), which is what you call to say you have focus, and you want your parent to look for the next one. You may want to override this method if your custom View has an internal view for your want to forward the request. Parameters sense int: One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT and FOCUS_RIGHT previouslyFocedRect Retct: The rectangle (in the coordinate system of this View) to give a finer hint about where the focus is coming from. It can be null if there is no hint. Boolean returns if this view or one of his descendants really focused. BooleanaFocusFromTouch (final public request) Call this to try to focus on a specific view or one of its descendants. This is a special variant of requestFocus() that will allow visualizations that are not focusable in touch mode to request focus when they are touched. Boolean returns if this view or one of his descendants really focused. See also: Public empty requestLayout () Call this when something has changed, which has invalidated the layout of this view. This will schedule a view tree layout pass. This should not be called while the view hierarchy is currently in a layout pass (isInLayout(). If the layout is happening, the request can be honored at the end of the current layout pass (and then the layout will run again) or after the current frame is drawn and the next layout occurs. Subclasses that override this method should call the superclass method to correctly handle possible request errors during layout. If you override this method, you must call up the superclass implementation. Public boolean requestRectangleOnScreen (straight rectangle) Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. View should call this if it keeps some notion of which part of its content is interesting. For example, a text editing view should call it when its cursor moves. The rectangle passed to this method must be in the view content coordinate space. It should not be affected by which part of the View is currently visible or its parchment position. Rectangle Rectangle Parameters Rectangle: The rectangle in the content coordinate space of the Boolean view returns if any parent rolled. public boolean requestRectangleOnScreen (rect rectangle, immediate boolean) Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. View should call this if it keeps some notion of which part of its content is interesting. For example, a text editing view should call it when its cursor moves. The rectangle passed to this method must be in the view content coordinate space. It should not be affected by which part of the View is currently or your parchment position. When the mate is set to true, scrolling is not animated. Rectangle Rectangle Rectangle parameters: The rectangle in the contents of the View coordinates the immediate boolean space: It is true to prohibit prohibit scrolling, false otherwise Boolean returns if any parent rolled. Public End Empty RequestInbufferedDispatch (MotionEvent event) Request the unreceived submission of the given flow of Motion Events to this View. Until this View receives a matching MotionEvent#ACTION_UP, ask the motionevents not to batch input system, but instead deliver them as soon as they become available. This method should only be called for touch events. This API is not intended for most applications. Buffered dispatch provides many benefits, and just requesting dispatch not offered on most MotionEvent streams will not improve your input latency. Side effects include: increased latency, nervous scrolls, and inability to take advantage of system reamration. Talk to your inbound professional to see if the InbufferedDispatch request (android.view.MotionEvent) is right for you. To receive events not offered for source classes of arbitrary input devices, use requestInbufferedDispatch(int), MotionEvent event parameters See also: RequestUnbufferedDispatch(int) public end T requireViewById (int id) Finds the first descending view with the given ID, the view itself if the ID matches getId(), or throws an IllegalArgumentException if the ID is invalid or there is a corresponding view in the hierarchy. Note: In most cases - depending on compiler support - the resulting view is automatically posted to the target class type. If the target class type is not constricted, an explicit cast may be required. Parameters id int: the ID to search for returns T a view with certain ID This value cannot be null. See also: public static int resolveSizeAndState (int size, int measureSpec, int childMeasuredState) Utility to reconcile a desired size and state, with restrictions imposed by a MeasureSpec. It will take the desired size, unless a different size is imposed by restrictions. The returned value is a composite integer, with the size resolved in the MEASURED_SIZE_MASK bits and optionally the MEASURED_STATE_TOO_SMALL bit set if the resulting size is smaller than the size that the view wants to be. Int size parameters: How big the view wants to be. measuresSpec int: Restrictions imposed by the parent. childMeasuredState int: Size information bit mask for view children. Public Boolean restoreDefaultFocus () Focuses on the default-focus view on the view hierarchy that has this view as a root. If the default focus view cannot be found, the call RequestFocus (int). Boolean returns If this visualization or one of its descendants actually had final empty public focus saveAttributeDataForStyleable (Context, int[] styleable, AttributeSet attrs, TypedArray t, int defStyleAttr, int defStyleRes) stores debug information about attributes. This should be in a constructor for each custom View that uses a custom style. If the custom view does not call it, then the custom attributes used by this view will not be visible in the layout inspection tools. Context of parameters Context: Context under which the vision is created. This value cannot be null. int stylized: A reference to the stylized array R.styleable.Foo This value cannot be null. attrs AttributeSet: AttributeSet used to construct this view. This value can be null. t TypedArray: Resolved TypedArray returned by a call to Resources#obtainAttributes (AttributeSet, int[]). This value cannot be null. defStyleAttr int: Default style attribute passed to the preview constructor. defStyleRes int: Default style feature passed to the preview builder. Drawable who, Runnable what, long when schedulean action on a drawable to occur at a specified time. Parameters that Drawable: The recipient of the action This value cannot be null. that Runnable: The action to perform on the drawable This value cannot be null. when long: the time at which the action should occur. Uses the SystemClock#uptimeMillis time base. parchment of the public voidBy (int x, int y) Move the scrolled position of your view. This will cause a call to onScrollChanged (int, int, int, int) and the view to be invalidated. Parameters x int: The amount of pixels to scroll horizontally y int: The amount of pixels to scroll through the vertically public empty parchmentTo (int x, int y) Set the scrolled position of your view. This will cause a call to onScrollChanged (int, int, int, int) and the view to be invalidated. Parameters x int: The x position to scroll to y int: the y position to scroll to the empty public setAccessibilityDeleggate (View.AccessibilityDelegate delegate) Defines a delegate to implement accessibility support via composition (as opposed to inheritance). For more details, see AccessibilityDelegate. Note: In platform versions prior to API 23, the methods of delegateing to visualizations in the android.widget.* package are called before the host methods. This prevents certain properties, such as the class name, from being modified, replacing accessibilitydelegate#onInitializeAccessibilityNodeInfo (View, AccessibilityNodeInfo), since any changes will be overwritten by the host class. Starting with API 23, delegate methods are called after host methods, which all properties must be modified without being overridden by the host class. Parameters delegate View.AccessibilityDelegate: The object to which calls from the accessibility method must be delegated This value can be null. See also: View.AccessibilityDelegate public void setAccessibilityHeading (boolean isHeading) Set if's a heading for a section of content of contenty to accessibility. Related XML attributes: android:accessibility Head parameters is Boolean: true if the view is a title, false otherwise. Public void setCompatibilitySSorregion (int mode) sets the live region mode for this view. This indicates to accessibility services whether they should notify the user about changes to the description or text of the content of the view, or in the descriptions of the content or text of the children of the view (where applicable). For example, on a login screen with a TextView that displays an incorrect incorrect notification, this view should be marked as a live region with ACCESSIBILITY_LIVE_REGION_POLITE. To turn off change notifications for this view, use 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. If the view changes stop the speech in progress and notify the user immediately, use ACCESSIBILITY_LIVE_REGION_ASSERTIVE. Related XML attributes: android:accessibilityLiveRegion public void setAccessibilityPaneTitle (CharSequence accessibilityPaneTitle) Visually distinct part of a window with window-like semantics are considered panebeses for accessibility purposes. An example is viewing content from a fragment that is overwritten. For accessibility services to understand the behavior of a window, panes must have descriptive titles. Views with dashboard titles produce Accessibility Events when they appear, disappear, or change the title. Related XML attributes: android:accessibilityPaneTitle Parameters accessibilityPaneTitle CharSequence: The title of the panel. The null setting indicates that this view is not a panel. This value can be null. public void setAccessibilityTraversalAfter (int afterId) Sets the id of a view after which it is visited at accessibility traversal. A screen reader should visit the contents of the other view before the content of the view. For example, if view B is set to be after display A, then a screen reader traverses all content from A before traversing all content from B, relative to the traversing strategy it is using. Visualizations that have not specified before/after relationships are traversed in order determined by the screen reader. Defining that this view is after a vision that is not important for accessibility or whether this view is not important for accessibility will have no effect because the screen reader is not aware of unimportant views. Related XML attributes: android:accessibilityTraversalAfter afterId int: The id of a visualization this succeeds in traversing accessibility. See also: setImportantForAccessibility(int) empty public setAccessibilityTraversalBefore (int beforeId) Sets the id of a view before which it is visited on accessibility traversal. A screen reader should visit the contents of this view before the content of the preceding one. For example, if view B is set to be before view A, then a screen reader traverses all content from B before traversing all content from A, relative to the traversing strategy it is using. Visualizations that have not specified before/after relationships are traversed in order determined by the screen reader. Define that this view is before a view that is not important for accessibility or if this view is not important for accessibility will have no effect, because the screen reader is not aware of unimportant views. Related XML attributes: android:accessibilityTraversalBefore Parameters before Id int: The id of a preview view precedes itself in the crossing of accessibility. See also: set ImportantForAccessibility (int) public void set Enabled (boolean enabled) Changes the enabled state of this view. A view can be activated or not. Note that activation is not the same as the selection. The selection is a transient property, representing the view (hierarchy) with which the user is currently interacting. Activation is a long-term state that the user can move views in and out. For example, in a single or multiple-select list view enabled, visualizations in the current selection set are enabled. (Um, yes, we are deeply sorry about the terminology here.) The activated state is propagated to the children of the vision in which it is connected. Boolean enabled parameters: True if the view is to be enabled, false public void setAlpha (float alpha) Sets the opacity of the view to a value of 0 to 1, where 0 means that the view is completely transparent and 1 means that the view is completely opaque. Note: Alpha fixation 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 transitioning, as in the case of faded animations. For a view with an alpha that changes frequently, such as during a faded animation, it is strongly recommended for performance reasons to override hasOverlappingRendering() to return false, if appropriate, or set a layer type in the view for the duration of the animation. In Build.VERSION_CODES. M and below, the default path to render a layerless view with alpha could add several milliseconds of rendering cost, even for simple or small visualizations. Starting with Build.VERSION_CODES. M, LAYER_TYPE_HARDWARE is automatically applied to the preview at the rendering level. If this view replaces the int (int) to return true, then this view is responsible for applying the opacity itself. In Build.VERSION_CODES. LOLLIPOP_MR1 and below, note that if the view is supported by a layer and is associated with a layer paint, setting an alpha value less than 1.0 will replace the alpha of the layer paint. Starting with Build.VERSION_CODES. M, setting a translucent alpha value will cut a View to its boundaries unless the view returns false from hasOverlappingRendering(). Related XML attributes: Alpha parameters float: The opacity of the view. The value is between 0.0 and 1.0 inclusive public empty setAnimation (animation animation) sets the next animation to play for this view. If you want the animation to play immediately, use startAnimation (android.view.animation.Animation) instead. This method allows fine control over start time and invalidation, but you must make sure that 1) the animation has a set of start, and 2) the parent of the view (which controls animations in your children) will be invalidated when the animation should begin. Animation parameter animation: The next animation, or null. public void setAutofillHints (String... (String... Defines tips that help an AutofillService determine how to automatically filter the visualization with user data. Typically, there is only one way to automatically fill a view, but there may be more than one. For example, if the app accepts a user name or email address to identify a user. These tips are not validated by the Android System, but passed as is to the service. Thus, they can have any value, but it is recommended to use AUTOFILL_HINT_ constants such as: AUTOFILL_HINT_USERNAME, AUTOFILL_HINT_PASSWORD, AUTOFILL_HINT_EMAIL_ADDRESS, AUTOFILL_HINT_NAME, AUTOFILL_HINT_PHONE, AUTOFILL_HINT_POSTAL_ADDRESS, AUTOFILL_HINT_POSTAL_CODE, AUTOFILL_HINT_CREDIT_CARD_NUMBER, AUTOFILL_HINT_CREDIT_CARD_SECURITY_CODE, AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE, AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DAY, AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH or AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_YEAR. Related XML Attributes: String autofillHints parameters: The autocomplete tips to define. If the array is emtpy, null is set. This value can be null. AutofillId id (AutofillId id) public void set defines the unique and logical identifier of this view in the activity for autofill purposes. The autocomplete id is created on demand, and this method should only be called when a view is reused after the ProvideAutofillStructure (android.view.ViewStructure, int) submission is called because this method creates a snapshot of the view that is transmitted to the autofill service. This method is typically used when preview subtrees are recycled to represent different contents* —in this case, the autocomplete id can be saved before the display content is swapped and restored later when it is swapped back. For example: EditText reusableView = ...; ViewGroup = parentView ...; AutofillManager afm = ...; Swap the view and change your contents AutofillId oldId = reusableView.getAutofillId(); CharSequence oldText = reusableView.getText(); parentView.removeView (reusableView); AutofillId newId = afm.getNextAutofillId(); reusableView.setText(New I am); reusableView.setAutofillId(newId); parentView.addView(reusableView); Later, swap the old content back in parentView.removeView(reusableView); reusableView.setAutofillId(oldId); reusableView.setText(oldText); parentView.addView(reusableView); AutofillId id parameters: A unique autocomplete ID in the Activity hosting the view or null to reset it. It is usually an id previously allocated to another view (and obtained through getAutofillId()), or a new value obtained through AutofillManager#getNextAutofillId(). This value can be null. Public Void SetBackground set set the background to a given Drawable or remove the background. If the background has padding, the this View will be set to fill the background. However, when a background is removed, the fill of this View is not touched. If the fill setting is desired, use setPadding (int, int, int, int). Drawable background parameters: The Drawable to use as the background, or null for for The public background empty setBackgroundColor (color int) sets the background color for this view. Int color parameters: The color of the public background void setBackgroundResource (int resid) Set the background for a given resource. The resource must refer to a Drawable object or 0 to remove the background. Related XML attributes: Parameters reside int: The resource identifier. Public End Void SetBottom (bottom int) Sets the lower position of this view relative to its parent. This method should be called by the layout system and should generally not be called otherwise, because the property can be changed at any time by the layout. Lower parameters: The bottom of this view, in pixels. CameraDistance Public Void Set Sets the distance along the Z axis (orthogonal to the X/Y plane on which visualizations are drawn) from the camera to this view. The distance from the camera affects 3D transformations, for example, rotations around the X and Y axis. The camera distance from the view plane can have an effect on the view's distortion perspective when it is rotated around the x or y axis. For example, a large distance will result in a large angle of view, and there won't be much view perspective distortion as it rotates. A short distance can cause much more perspective distortion in rotation, and can also result in some drawing artifacts if the rotated view ends partially behind the camera (which is why the recommendation is to use a distance at least until the view size if the view is rotated.) The distance is expressed in depth pixels. The default distance depends on the density of the screen. For example, on an average 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 at various densities, use the following formula: flotation 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 fluctuate: The distance in depth pixels, if negative the opposite value is used See also: setRotationX(float)setRotationY(float) set of public voidClickable (clickable boolean) Allows or disables click events for this view. When a view is clickable, it changes its state to pressed on each click. Subclasses must set the clickable view to react visually to user clicks. Related XML attributes: Boolean clickable parameters: faithful to make the display clickable, otherwise, the Public Void SetClipBounds (Rect clipBounds) defines a rectangular area in this view for which the view will be when it is drawn. Setting the null value will remove the clip boundaries and the display will be drawn normally, using its full boundaries. ClipBounds Retct: The rectangular area, in the local coordinates of this view, for which future drawing operations will be cut off. Public void setContentCaptureSession (ContentCaptureSession contentCaptureSession) defines the (optional) ContentCaptureSession associated with this view. This method should be called when you need to associate a ContentCaptureContext with the content capture events associated with that view or its view hierarchy (if it is a ViewGroup). For example, if your activity is associated with a web domain, you first need to define the context for the main DOM: ContentCaptureSession mainSession = rootView.getContentCaptureSession(); mainSession.setContentCaptureContext(ContentCaptureContext.forLocusId(Uri.parse(myUrl)); So if the page had an IFRAME, you would create a new session for it: ContentCaptureSession iframeSession = mainSession.createContentCaptureSssion(ContentCaptureContext.forLocusId(Uri.parse(iframeUrl)); iframeView.setContentCaptureSession(iframeSession); public void definedContentDescription (Content of 1edes) Defines the description of the contents of the View. A content description briefly describes the view and is primarily used for accessibility support to determine how a view should be presented to the user. In the case of a view without textual representation, such as ImageButton, a useful description of the content explains what the visualization does. For example, an image button with a phone icon used to make a call can use Call as your content description. An image of a floppy used to save a file can use Save. Related XML attributes: android:contentDescription Parameters contentDescription CharSequence: The content description. See also: Public void definedContextClickable (booleanoClickable context) Allows or disables the context by clicking for this view. This event can launch the listener. Related XML attributes: Boolean Clitable parameter context: faithful to make the view react to a context click, false otherwise See also: public void setDefaultFocusHighlightEnabled (booleanoFocushighlightEnabled default) Defines whether this view should use a default focus highlight when it focuses, but has R.attr.state_focused set in its background. Related XML attributes: android:defaultFocusHighlightD Default parametersFocusHighlightEnabled boolean: it's true to set this view to use a default, false focus highlight otherwise. Added to the Deprecated Api level 1 in the public void level 28 API setDrawingCacheBackgroundColor (color int) This method has been deprecated at API level 28. The preview drawing cache has become largely obsolete with the introduction of accelerated rendering hardware in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. No No cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Setting a solid background color for the drawing cache bitmaps will improve memory performance and usage. Note, however, this should only be used if this view is always drawn on top of a solid color. Int color parameters: The background color to use for the drawing cache bitmap added at API level 1 Deprecated in the public empty set API 28DrawingEnabled (boolean-enabled) This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Enables or disables drawing caching. When drawing caching is enabled, the next call to getDrawingCache() or buildDrawingCache() will draw the view on a bitmap. Draw call (android.graphics.Canvas) will not be removed from the cache when caching is enabled. To benefit from caching, you must request the loot cache by calling getDrawingCache() and draw it on the screen if the returned bitmap is not

null. Enabling drawing caching is similar to setting a layer when hardware acceleration is turned off. When hardware acceleration is enabled, enabling drawing cache has no effect on rendering because the system uses a different acceleration mechanism that ignores the If you want to use a Bitmap for display, even when hardware acceleration is enabled, see setLayerType (int, android.graphics.Paint) for information about how to enable software and hardware layers. This API can be used to manually generate a bitmap copy of this setting the flag to true and calling getDrawingCache(). Boolean-enabled parameters: True to enable drawing caching, otherwise false added at API level Depreed at public empty api level 28 definedDrawingDrawingQuality (quality int) This method was depreed at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Set the quality of the drawing cache in this view. This value is only used when drawing caching is enabled Related XML attributes: android:drawingCacheQuality public void setDuplicateParentStateEnabled (boolean enabled) enables or disables duplication of the parent state in this view. When duplication is enabled, this view gets its default state from its parent and not its own internal properties. Note: In the current implementation, setting this property to true after the view has been added to a ViewGroup may have no effect. This property must always be used from XML or set to true before adding this view to a ViewGroup. Note: If the controller property of this view addStateFromChildren is enabled and this property is enabled, an exception is thrown. Note: If the child view uses and updates additional states unknown to parents, these states should not be affected by this method. Boolean-enabled parameters: True to allow duplication of the parent's draw state, false to disable it. See also: getDrawableState()isDuplicateParentStateEnabled() public void setElevation sets the base elevation of this view, in pixels. Related XML attributes: Elevation parameters float public void setEsseabled (boolean-enabled) Set the enabled state of this view. The interpretation of the enabled state varies by subclass. Boolean enabled parameters: True if display is enabled, false otherwise. Public void setFadingEdgeLength (length int) Set the size of the faded border used to indicate that more content in this view is available. It will not change if the fading edge is enabled; Use setVerticalFadingEdgeEnabled(boolean) or setHorizontalFadingEdgeEnabled (boolean) to allow faded edge to the or horizontal fading edges. Int length parameters: The pixel size of the faded border used to indicate that more content in this view is visible. Public void setFilterTouchesWhen obscured (boolean enabled) defines whether the structure should drop touches when the display window is obscured by another visible window. See the View security documentation for more details. Related XML attributes: android:filterTouchesWhenObscured Parameters activated boolean: True if touch filtering should be enabled. See also: getFilterTouchesWhenObscured() public void setFocusableInTouchMode (boolean focusableInTouchMode) 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 boolean: If true, this view may receive focus in touch mode. See also: Public void setFocusedByDefault (boolean isFocusedByDefault) Defines whether this view should be focused when the focus is restored to the view hierarchy that contains this view. Focus is restored to a visualization hierarchy when the root of the hierarchy is added to a window or serves as a cluster navigation target. Related XML attributes: Boolean isFocusedByDefault parameters: faithful to set this view to the default-focus view, otherwise false. See also: the public void setForceDarkAllowed (boolean allow) Defines whether or not to allow the dark force to apply to this view. Setting this to false will disable the auto-dark feature on whatever this view draws, including any descendants. Setting this to true will allow this view to be automatically made dark, however, a value of 'true' will not replace any 'false' value in your parent chain or prevent any 'false' in any of your children. The default behavior of dark force is also influenced by the IsLightTheme attribute of the Theme. If a theme is isLightTheme=false, then force dark is globally disabled for that theme. Parameters allow Boolean: Allow or not dark force. Foreground public void set (drawable foreground) Provide a Drawable that must be rendered on top of all content in the view. Related XML attributes: Drawable foreground parameters: The Drawable to be drawn on top of the child public void definedHapticFeedbackEnabled (boolean hapticFeedbackEnabled) Define whether this view should have haptic feedback for events such as long presses. You may want to disable the ptitic feedback if your vision already controls your own timetic feedback. Related XML attributes: android:hapticFeedbackEnabled Parameters hapticFeedbackEnabled boolean: if haptic feedback allowed this visualization. See also: Public void setHasTransientState (boolean hasTransientState) set whether this view is currently tracking the transient state that the framework should try to preserve when possible. This flag is counted by reference, so each call to setHasTransientState must be paired with a later call to setHasTransientState (false). A transient state view cannot be trivially reboundfrom an external data source, such as an adapter linking item view in a list. This can be because the view is performing an animation, crawling the selection of user content or similar. Parameters hasTransientState boolean: true if this view has transient state public void setHorizontalFadingEdgeEnabled (horizontal booleaneFadingEdgeEnabled) Set whether horizontal edges should be faded when this view is scrolled horizontally. Related XML attributes: android:requerFadingEdge HorizontalparametersFadingEdgeEnabled boolean: true if the horizontal edges should be faded when the view is scrolled horizontally See also: isHorizontalFadingEdgeEnabled() set (horizontalscrollScrollBarEnabled) Set whether the horizontal scroll bar should be drawn or not. The scroll bar is not drawn by default. Horizontal ParametersScrollBarEnabled boolean: true if the horizontal scrollbar should be painted See also: isHorizontalScrollBarEnabled() public void setHovered (boolean paired) sets whether the display is currently hovered. Calling this method also changes the view state. This allows the visualization to react when hovering using different drawable features to change its appearance. The onHoverChanged (boolean) method is called when the hovered state changes. Parameters hovered boolean: True if the view is hovered. See also: isHovered()onHoverChanged(boolean) set of public voidId (int id) Sets the identifier for this view. The identifier does not have to be unique in the hierarchy of this view. The identifier must be a positive number. Related XML attributes: Int id parameters: A number used to identify the view See also: NO_IDgetId()findViewById(int) public empty setImportanteForAutofill (int mode) Sets the mode to determine whether this visualization is considered important for autocomplete. The platform determines the importance of autocomplete automatically, but you can use this method to customize the behavior. For example: When view content is irrelevant to autocomplete (for example, a text field used in a Captcha challenge), it must be IMPORTANT_FOR_AUTOFILL_NO. When both the view and its children are irrelevant to autocomplete (for example, the root view of an activity containing a spreadhseet editor), it must be IMPORTANT_FOR_AUTOFILL_NO_EXCLUDE_DESCENDANTS. When the display content is relevant to autocomplete, but your children are not (for example, a credit card expiration date represented by a custom view that replaces the appropriate autocomplete methods and has 2 children representing the month and year), it should be IMPORTANT_FOR_AUTOFILL_YES_EXCLUDE_DESCENDANTS. Note: Setting how you IMPORTANT_FOR_AUTOFILL_NO IMPORTANT_FOR_AUTOFILL_NO_EXCLUDE_DESCENDANTS not guarantee that viewing (and your children) will always be considered unimportant; for example, when the user explicitly makes an autocomplete request, all considered important. See isImportantForAutofill() for more details on how the importance of View for autocomplete is used. Related XML attributes: android:importantForAutofill public void setKeyboardNavigationCluster (boolean isCluster) Define whether this visualization is a root of a keyboard navigation cluster. Related XML attributes: android:keyboardNavigationAlessescluster boolean isCluster parameters: If true, this visualization is a root of a cluster. Public void setLabelFor (int id) defines the id of a view for which this view serves as a label for accessibility purposes. Parameters id int: The labeled display id. public void setLayoutParams (ViewGroup.LayoutParams params) Set the layout parameters associated with this view. These parameters supply to the parent of this view specifying how it should be organized. There are many viewgroup.LayoutParams subclasses, and these correspond to the different ViewGroup subclasses that are responsible for organizing your children. Parameters params ViewGroup.LayoutParams: The layout parameters for this view cannot be null sets of public end emptyLion (int left) Sets the left position of this view relative to its parent. This method should be called by the layout system and should generally not be called otherwise, because the property can be changed at any time by the layout. Parameters left int: The left of this view, in pixels. Public end empty setLeftTopRightBottom (int left, int top, int right, bottom int) Assign a size and position to this view. This method should be used in animations only because it applies this position and size to the view only temporarily and can be changed at any time by the layout. Parameters left int: Left position, relative to the upper int of the parents: Upper position, relative to the right int of the parents: Lower position: Lower position, relative to the parent See also: #setRight(int), #setTop(int), #setBottom(int) set of public voidLongClickable (boolean longClickable) Allows or disables long click events for this view. When a display is long clickable, it reacts to the user holding the button for a duration longer than a touch. This event can launch the listener or a context menu. Related XML attributes: Long clickable boolean parameters: faithful to make the long view clickable, false otherwise See also: public void setMinimumHeight (int minHeight) Sets the minimum view height. It is not guaranteed that the view will be able to reach this minimum height (for example, if its parent layout restricts it with less height available). Related XML Attributes: MinHeight int Parameters: The minimum height that the view will try to be, in pixels See also: public void setMinimumWidth (int minWidth) Sets minimum width of the view. It is not guaranteed that the visualization will be able to achieve this minimum width (for example, if its parent layout restricts it with less available width). Related XML Attributes: MinWidth int Parameters: The minimum width that the view will try to be, in Pixels View View This dScrollingEnabled (boolean-enabled) enable or disable nested scrolling for this view. If this property is set to true, the view can start nested scrolling operations with a compatible parent view in the current hierarchy. If this view does not implement nested scrolling, this has no effect. Turning off nested scrolling while a nested scroll is in progress has the effect of stopping the nested parchment. Boolean-enabled parameters: True to enable nested scrolling, false to disable See also: isNestedScrollingEnabled() public empty setNextClusterForwardId (int nextClusterForwardId) Sets the preview id to use as the root of the next keyboard navigation cluster. Related XML attributes: android:nextClusterForward Parameters nextClusterForwardId int: The next cluster ID or NO_ID whether the framework should decide automatically. Public void setExtFocusDownId (int nextFocusDownId) Sets the view id to use when the next focus is FOCUS_DOWN. Related XML attributes: NextFocusDownId int: The next focus ID or NO_ID whether the framework should decide automatically. NextFocusForwardId (int nextFocusForwardId) sets the view id to use when the next focus is FOCUS_FORWARD. Related XML attributes: NextFocusForwardId int: The next focus ID or NO_ID whether the framework should decide automatically. Public void setExtFocusLeftId (int nextFocusLeftId) Sets the view id to use when the next focus is FOCUS_LEFT. Related XML attributes: NextFocusLeftId int: The next focus ID or NO_ID whether the framework should decide automatically. Public void setExtFocusRightId (int nextFocusRightId) Sets the view id to use when the next focus is FOCUS_RIGHT. Related XML attributes: NextFocusRightId int: The next focus ID or NO_ID whether the framework should decide automatically. Public void setExtFocusUpId (int nextFocusUpId) Sets the view id to use when the next focus is FOCUS_UP. Related XML attributes: NextFocusUpId int: The next focus ID or NO_ID whether the framework should decide automatically. Public void setOnClickListener (View.OnClickListener l) Register a callback to be invoked when this view is clicked. If this view is not clickable, it becomes clickable. L Ver.OnClickListener parameters: The callback that will execute this value may be null. See also: Public void setOnContextClickListener (View.OnContextClickListener l) Record a callback to be invoked when that view is clicked in context. If the view is not clickable in context, it becomes clickable in the context. L View.OnContextClickListener Parameters: The callback that will This value can be null. See also: setContextClickable(boolean) public void setOnCreateContextMenuListener (View.OnCreateContextMenuListener l) Register a callback to be invoked when the context menu for this view is being built. If this display is not long clickable, it becomes long clickable. Parameters l l The callback that will run the public void setOnFocusChangeListener (View.OnFocusChangeListener l) Register a callback to be invoked when the focus of this view has changed. L View.OnFocusChangeListener Parameters: The callback that will run. Public void setOnGenericMotionListener (View.OnGenericMotionListener l) Register a callback to be invoked when a generic motion event is sent to this viewpoint. L View.OnGenericMotionListener parameters: The generic motion listener attaching to this view the public void setOnHoverListener (View.OnHoverListener l) Register a callback to be invoked when a hover event is sent to this view. l View.OnHoverListener parameters: The hover listener to attach to this public empty view setOnKeyListener (View.OnKeyListener l) Register a callback to be invoked when a hardware key is pressed in this view. Key presses in software input methods generally do not trigger this listener's methods. L View.OnKeyListener parameters: The key listener to attach to this view the public void set OnLongClickListener (View.OnLongClickListener l) Register a callback to be invoked when this view is clicked and performed. If this display is not long clickable, it becomes long clickable. L View.OnLongClickListener parameters: The callback that will execute this value may be null. See also: setLongClickable (boolean) public empty setOnScrollChangeListener (View.OnScrollChangeListener l) Record a callback to be invoked when the x or y parchment positions of this view change. Note: Some visualizations handle view-independent scrolling and may have their own listeners separate for scroll-type events. For example, The ListView allows clients to register an AbsListView.OnScrollListener to hear changes in the scrolling position of the list. L View.OnScrollChangeListener parameters: The listener to notify when the position of the X or Y parchment changes. See also: Public void setOnTouchListener (View.OnTouchListener l) Record a callback to be invoked when a touch event is sent to this view. l View.OnTouchListener parameters: The touch listener attaching to this view the public void setOutlineAmbientShadowColorColor (color int) sets the color of the ambient shadow that is drawn when the view has a Z value or positive elevation. By default, the shadow color is black. Generally, this color will be opaque, so the shadow intensity is consistent between different views with different colors. The opacity of the final ambient shadow is a function of the height of the shadow cast, the alpha channel of the AmbientShadowColor (typically opaque) contour, and the Theme attribute R.attr.ambientShadowAlpha. Related XML attributes: android:outlineAmbientShadowColor Colored Int: The color that this Display will cast for your elevation shadow. Public void setOutlineProvider (Provider ViewOutlineProvider) Defines the ViewOutlineProvider of the view, which generates the Outline that defines the shape of the shadow that casts and allows the outline clipping. The ViewOutlineProvider Pattern, ViewOutlineProvider#BACKGROUND, queries the from the bottom of the drawable View, via Drawable#getOutline. Changing the contour provider with this method allows this behavior to be overridden. If the ViewOutlineProvider is null, if you query it for a false return outline, or if the contour produced is Contour#isEmpty(), the shadows are not cast. Only contours that return true contour#canClip() can be used for clipping. Parameters provider ViewOutlineProvider See also: setClipToOutline(boolean)getClipToOutline()getOutlineProvider() public empty setOutlineOutlineShadowColor (color int) Sets the color of the dot shadow that is drawn when the visualization has a Z value or positive elevation. By default, the shadow color is black. Generally, this color will be opaque, so the shadow intensity is consistent between different views with different colors. The opacity of the endpoint shadow is a function of the height of the shadow castor, the alpha channel of the ShadowColor (typically opaque) contour, and the Theme attribute R.attr.spotShadowAlpha. Related XML attributes: android:outlineSpotShadowColor Colored parameters int: The color that this view will cast for your elevation point shadow. OverScrollMode (int overScrollMode) set the over-scroll mode for this view. Valid over-scrolling modes are OVER_SCROLL_ALWAYS, OVER_SCROLL_IF_CONTENT_SCROLLS (allow excessive scrolling only if the display content is larger than the container) or OVER_SCROLL_NEVER. Setting the over-scrolling mode of a view will only have an effect if the view is able to scroll. OverScrollMode int parameters: The new over-scroll mode for this view. PivotX (pivotX floating disk) public void set Sets the x location of the point around which the view is rotated and scaled. By default, the pivot point is centered on the object. Setting this property disables this behavior and causes the visualization to use only the explicitly defined pivotX and pivotY values. Related XML attributes: PivotX float parameters: The x location of the pivot point. See also: getRotation()getScaleX()getScaleY()getPivotY() public void setPivotY (float disk) Sets the y location of the point around which the view is rotated and scaled. By default, the pivot point is centered on the object. Setting this property disables this behavior and causes the visualization to use only the explicitly defined pivotX and pivotY values. Related XML attributes: PivotY float parameters: The y location of the pivot point. See also: getRotation()getScaleX()getScaleY()getPivotY() public empty setPointerIcon (PointerIcon pointerIcon) Set the pointer icon to the current view. Passing null will restore the pointer icon to its default value. PointerIcon PointerIcon parameters: A pointerIcon instance that will be shown when the mouse hovers. set of public voidPressed sets the pressed state for this view. Boolean pressed parameters: Pass faithful to set the internal state of the View to pressed, or false to revert the internal state of the View from a previously defined state pressed. See also: also: Public end void setRevealOnFocusHint (boolean revealOnFocus) Sets the preference of this view to reveal the behavior when it gains focus. When set to truth, this is a sign for the ancestral views in the hierarchy that this view would rather be fully seen when it gains focus. For example, a text field in your typing objective. Other visualizations, such as scroll containers, may prefer to choose not to follow this behavior. The default value for visualizations is true, although subclasses can change this based on their preferred behavior. Parameters revealOnFocus boolean: true to the request for revelation about the focus on the ancestors, false otherwise See also: defined public final voidReto (int on the right) Defines the right position of this view in relation to its father. This method should be called by the layout system and should generally not be called otherwise, because the property can be changed at any time by the layout. Parameters int: The right of this view, in pixels. SaveEnabled (boolean enabled) public void set controls whether the state savings of this view is enabled (that is, whether your onSaveInstanceState() method is called. Note that even if freezing is enabled, the view must still have an id assigned to it (via setId(int)) for its state to be saved. This flag can only disable the salvation of this vision; any childish vision can still have its state saved. Related XML attributes: Boolean-enabled parameters: Set to false to disable state salvation or true (default) to allow this. See also: isSaveEnabled()setId(int)onSaveInstanceState() public set of emptyScaleX (floating scaleX) Defines the amount that the view is scaled in x around the pivot point, as a proportion of the unscaled width of the view. A value of 1 means that no escalation is applied. Related XML attributes: ScaleX parameters float: The scale factor. See also: The ScaleY (floating scale) public void set Defines the amount that the view scales around the pivot point, as a proportion of the view's non-escalating width. A value of 1 means that no escalation is applied. Related XML attributes: ScaleY parameters float: The scale factor. See also: The public void of the ScreenReaderFocusable set (booleanaReaderFocusable screen) Defines whether this view should be a focalable element for screen readers and include non-focal views of your subtree when providing feedback. Note: This is similar to using android:focusable, but does not affect the input focus behavior. Related XML attributes: android:screenReaderThe analysis parametersReaderFodiz: Whether the visualization should be treated as a unit by screen reader accessibility tools. public empty setScrollBarDefaultDelayBeforeFade scrollBarDefaultDelayBeforeFade) Set the delay before the scroll bars disappear. Related XML attributes: android:scrollbarDefaultDelayBeforeFade Parameters scrollBarDefaultDelayBeforeFade int: - The delay before scroll bars disappear the public void setScrollBarFadeDuration (int scrollBarFadeDuration) Set the fade duration of the scroll bar. Related XML attributes: related: ScrollBarFadeDuration int: - The duration of scrollbar fading, in milliseconds of empty public setScrollBarSize (int scrollBarSize) Set the size of the scrollbar. Related XML attributes: ScrollBarSize int: - the size of the scrollbar size of the size of the public void definedScrollBarStyle (int style) Specify the style of the scroll bars. Scroll bars can be overlapored or inset. When inset, they add to the view fill. And scroll bars can be drawn within the fill area or at the edge of the view. For example, if a view has a drawing background and you want to draw the scroll bars within the fill specified by drawable, you can use SCROLLBARS_INSIDE_OVERLAY or SCROLLBARS_INSIDE_INSET. If you want them to appear on the edge of the view, ignoring the padding, then you can use SCROLLBARS_OUTSIDE_OVERLAY or SCROLLBARS_OUTSIDE_INSET. Related XML attributes: Public void setScrollContainer (boolean isScrollContainer) Change if this view is one of the scrollable containers in your window. This will be used to determine whether the window can resize or should be used when a soft input area is open -- scrollable containers allow the window to use resize mode, since the container will shrink accordingly. Related XML attributes: android:isScrollContainer Parameters isScrollContainer boolean public void setScrollIndicators (indicators int, mask int) Sets the state of scroll indicators specified by the mask. To change all scroll indicators at once, see setScrollIndicators(int). When a scroll indicator is enabled, it appears if the view can scroll in the direction of the indicator. Various types of indicators can be turned on or off by passing the logical OR of the desired types. If multiple types are specified, all of them are set to the same enabled state. For example, to enable the top scroll indicatorExample: {@code setScrollIndicators Related XML Attributes: See also: setScrollIndicators(int)getScrollIndicators() public set voidScrollX (int value) Set the scrolled horizontal position of your view. This will cause a call to onScrollChanged (int, int, int, int) and the view to be invalidated. Parameters value int: The x position to scroll to the public void setScrollY (value int) Set the scrolled vertical position of your view. This will cause a call to onScrollChanged (int, int, int, int) and the view to be invalidated. Parameters value int: The y position to scroll to the public void setScrollbarFadingEnabled (boolean fadeScrollbars) Set whether scroll bars will disappear when the display is not scrolling. Related XML Attributes: Parameters DisappearBoolean Scrollbars: If You Allow Cluster Fading Public emptySelected (boolean selected) Changes the selection state of this view. A view can be selected or not. Note that the selection is not the same as focus. Visualizations are typically selected in the context of an AdapterView such as ListView or GridView; the selected view is the view that is highlighted. Selected Boolean parameters: true if the display should be selected, false false Public void setSoundEffectsEnabled (booleanoEffectsEnabled sound) Set whether this view should have sound effects enabled for events such as click and tap. You may want to disable sound effects for a display if you already play sounds, for example, a dial key that plays dtmf tones. Related XML attributes: android:soundEffectsEnabled Parameters soundEffectsEnabled boolean: whether sound effects are enabled for this preview. See also: isSoundEffectsEnabled()playSoundEffect(int) the empty public setStateDescription (CharSequence stateDescription) sets the description of the state of the view. A state description briefly describes the states of the view and is primarily used for accessibility support to determine how the states of a view should be presented to the user. It is an add-in for Boolean states (for example, checked/unverified) and is used for custom state description (for example, wi-fi, connected, three bars). The state description changes frequently, while the description of the content should change less frequently. The description of the state must be found. For android widgets that have default state descriptions, application developers can call this method to override state descriptions. Setting the state description to null restores the default behavior. State ParametersDescription CharSequence: The state description. This value can be null. See also: The public void defines StateListAnimator (StateListAnimator stateListAnimator) Attaches the StateListAnimator provided to this view. Any previously attached StateListAnimator will be highlighted. StateAnimator: The StateListAnimator to update the view See also: public void setSystemGestureExclusionRects (List<Rect> rects) Defines a list of areas within the post-layout coordinate space of this view where the system should not intercept touches or other pointing device gestures. This method should be called by onLayout (boolean, int, int, int, int) or onDraw (android.graphics.Canvas). Use this to tell the system which specific subareas of a view need to receive gesture input to function correctly in the presence of global system gestures that might conflict. For example, if the system wants to capture sliding gestures at the edge of the screen to provide system-level navigation functionality, a view as a navigation drawer container can mark the left (or initial) edge of itself as requiring gesture-capturing priority using this API. The system can then choose to relax its own gesture recognition to allow the application to consume the user's gesture. It is not necessary for an application to record exclusion restrains for wide-ranging regions, such as the entirety of a ScrollView, or for simple click-through targets of and drop, like Button. Checking an exclusion reswipe when interacting with a view requires a precision touch gesture in a small area in the X or Y dimension, such as a border slide or dragging a SeekBar thumb. Do not modify the list provided after calling this method. Note: the system will place a limit of </Rect> </Rect> vertical extension of the exclusions you take into account. The limit does not apply while the navigation bar is hidden, nor to the input method and intent#CATEGORY_HOME. Recamis List parameters: A list of regions of precision gestures that this visualization needs to function correctly This value cannot be null. Added to the Deprecated Level 11 API in the public empty level 30 API setSystemUiVisibility (visibility int) This method has been deprecated at API level 30. SystemUiVisibility flags are depreced. Use windowinsetscontroller instead. Request that the visibility of the status bar or other screen/window decorations be changed. This method is used to place the UI in temporary modes where user attention is more focused on application content, darkening or hiding system payments around. This is typically used in conjunction with Window#FEATURE_ACTION_BAR_OVERLAY, allowing application content to be placed behind the action bar (and with these flags other system applications) so that smooth transitions between hiding and showing them can be made. Two representative examples of using system interface visibility is the implementation of a content navigation application (such as a magazine reader) and a video playback application. The first code shows a typical implementation of a content navigation application. In this implementation, the application enters a content-oriented mode, hiding the status bar and action bar, and putting the navigation elements in lights-out mode. The user can then interact with the content while in this mode. Such an application should provide an easy way for the user to switch out of mode (such as checking information in the status bar or accessing notifications). In the implementation here, this is done simply by tapping on the content. 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; mLastSystemUiVis int; Runnable mNavHider = new Runnable() { @Override public void race() { setNavVisibility(false); } } Public content (Context context, Attrs AttributeSet) { 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(fake); mText.setOnClickListener(isso); mText.setTextIsSelectable(truth); addView(mText, new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT)); setOnSystemUiVisibilityChangeListener(this); } public void init (title TextView, SeekBar seek) { // This called by the activity of to provide the surrounding content browser // the content browser with which it will interact. mTitleView = title; mSeekView = seek; setNavVisibility (true); @Override } } visibility) { // Detect when we exit low profile mode, to also exit // full screen. We only do this when low-profile mode // is changing from its last state, and shutting down. int diff = mLastSystemUiVis ^ visibility; mLastSystemUiVis = visibility; if ((diff&SYSTEM_UI_FLAG_LOW_PROFILE) != 0&& (visibility&SYSTEM_UI_FLAG_LOW_PROFILE) == 0) { setNavVisibility(true); } empty @Override protected inWindowVisibilityChanged (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 empty protected 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 empty public onClick(View v) { // When the user clicks, we switch the visibility of the navigation elements. int curVis = getSystemUiVisibility(); setNavVisibility((((((((((curVis&SYSTEM_UI_FLAG_LOW_PROFILE) != 0); } void setBaseSystemUiVisibility(visibility int) { 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(); Clear any pending events to hide navigation if we are // changing visibility or making the UI visible. if (changed || visible) { Handler h = getHandler(); if (h != null) { h.removeCallbacks(mNavHider); } } Set the new visibility you want. setSystemUiVisibility(newVis); mTitleView.setVisibility (visible? VISIBLE : INVISIBLE); mSeekView.setVisibility (visible? VISIBLE : INVISIBLE); } } This second code sample shows a typical implementation of a video playback application. In this situation, while the video is playing the application would like to enter a full full screen mode, to use as much of the display as possible for the video. When in this state the user cannot interact with the application; The system intercepts the touch on the screen to take the UI out of full screen mode. See fitSystemWindows (android.graphics.Rect) for a sample layout that came with this code. Public static class content extends ImageView implements View.OnSystemUiVisibilityChangeListener, View.OnClickListener, ActionBar.OnMenuVisibilityListener { mActivity activity; TextView mTitleView; mPlayButton button; SeekBar mSeekView; boolean mAddedMenuListener; boolean mMenusOpen; boolean mPaused; boolean mNavVisible; mLastSystemUiVis int; Runnable mNavHider = new Runnable() { @Override public void race() { setNavVisibility(false); } } Public content (Context context, Attrs AttributeSet) { super (context, setOnSystemUiVisibilityChangeListener(isso); setOnClickListener(isso); } public void init (Activity activity, TextView title, Button playButton, SeekBar seek) { // This called by the called by the called by the activity to provide the status of the product // state of the video player with which it will interact. mActivity = activity; mTitleView = title; mPlayButton = playButton; mSeekView = seek; mPlayButton.setOnClickListener(this); setPlayPaused (true); } @Override empty protected onAttachedToWindow() { super.onAttachedToWindow(); if (mActivity != null) { mAddedMenuListener = true; mActivity.getActionBar().addOnMenuVisibilityListener(this); } } } } @Override empty protected noDetachedFromWindow() { super.onDetachedFromWindow(); if (mAddedMenuListener) { mActivity.getActionBar().removeOnMenuVisibilityListener(this); } } } } } @Override empty public noSystemUiVisibilityChange (visibility int) { // Detect when we exit hidden browsing mode, to clear our state // back to having full UI chrome. Only do this when // the state is changing and navigation 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 empty protected in WindowVisibilityChanged (int visibility) { super.onWindowVisibilityChanged(visibility); // When we become visible or invisible, game is paused. setPlayPaused(true); } @Override empty public onClick(View v) { if (v == mPlayButton) { // Clicking on the play/pause toggles its state. setPlayPaused(!mPaused); } else { // Clicking elsewhere makes navigation visible. setNavVisibility(true); } } } @Override empty public in MenuVisibilityChanged(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); } empty setNavVisibility (visible boolean) { 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, set a timer to be invisible. if (visible) { Handler h = getHandler(); if (h != null) { h.removeCallbacks(mNavHider); if (!mMenusOpen &&!mPaused) { // If the menus are open or playback is paused, we will not hide automatically. setSystemUiVisibility(newVis); mTitleTitle.setVisibility (visible? VISIBLE : INVISIBLE); mPlayButton.setVisibility (visible? VISIBLE : INVISIBLE); mSeekView.setVisibility (visible? VISIBLE : INVISIBLE); } } public empty setTag (key int, object tag) Sets a tag associated with this view and a key. A tag can be used to mark a view in your hierarchy and does not have to be unique within the hierarchy. Tags can also be used to store data within a view without the other data structure. The specified key must be an id declared in the application resources to ensure that it is unique (see id resource type). Key Keys as belonging to the Android framework or not associated with any package will cause an IllegalArgumentException to be released. Key parameters int: The key that identifies the Object tag: An object to mark the display with Throws IllegalArgumentException If the specified key is not valid See also: setTag(Object)getTag(int) public void setTag (Object tag) Sets the tag associated with this view. A tag can be used to mark a view in your 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. Object tag parameters: An object to mark the view with View also: getTag()setTag(int, Object) empty public definedTextAlignment (int textAlignment) Set the text alignment. Related XML attributes: TextAlignment int parameters: The text alignment to define. Must be a 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 made if the value is set to TEXT_ALIGNMENT_INHERIT. The resolution proceeds to the parental chain of the view to obtain the value. If there is no parent, then it returns the TEXT_ALIGNMENT_GRAVITY pattern. The value is TEXT_ALIGNMENT_INHERIT, TEXT_ALIGNMENT_GRAVITY, TEXT_ALIGNMENT_CENTER, TEXT_ALIGNMENT_TEXT_START, TEXT_ALIGNMENT_TEXT_END, TEXT_ALIGNMENT_VIEW_START, or the Public Empty SetTooltipText (CharSequence tooltipText) defines the tooltip text that will be displayed in TEXT_ALIGNMENT_VIEW_END a small pop-up next to the view. The tooltip will appear: Under long click unless otherwise handled (by OnLongClickListener or a context menu). On the mouse, after a brief delay, since the pointer has stopped moving Note: Do not override this method, as it will have no effect on the text displayed at the tooltip. Related XML attributes: TipText CharSequence tool parameters: Tooltip text or null if no tooltip is required This value can be null. See also: Public End Void SetTop (top int) Sets the top position of this view relative to its parent. This method should be called by the layout system and should generally not be called otherwise, because the property can be changed at any time by the layout. Top parameters: The top of this view, in pixels. Public empty setTouchDelegate (Delegate TouchDelegate) sets the TouchDelegate to this view. Parameters delete public void set TouchDelegateTransitionAlpha (float alpha) This property is intended only for use by the Fade transition, which animates it to produce a visual translucency that does not effect the side effect (or is affected by) the actual alpha property. This value is composed with the other alpha value (and the AlphaAnimation value, when this is present) produce a final result of visual translucency, which is what is passed to the DisplayList. Public End Empty SetTransitionName (StringName Transition) Defines the name of the view to use to identify visualizations in The names must be unique in the view hierarchy. Parameter transitionThe sequence of names: The name of the view to uniquely identify it for Transitions. Public void setTransitionVisibility (visibility int) Changes the visibility of this view without triggering other changes. This should only be used by animation frameworks, such as Transition, where visibility changes should not adjust focus or trigger a new layout. Application developers should use setVisibility(int) instead of ensuring that the hierarchy is updated correctly. Only call this method when a temporary visibility should be applied during an animation and the original visibility value is guaranteed to be reset after the animation is complete. Use setVisibility(int) in all other cases. See also: Public void setTranslationX (floating translationX) Sets the horizontal location of this view relative to the left position. This effectively positions the object after the layout, as well as where the object layout placed it. Related XML attributes: Translation of parametersX float: The horizontal position of this view relative to its left position, in pixels. Public void setTranslationY (floating translationY) Sets the vertical location of this view relative to its top position. This effectively positions the object after the layout, as well as where the object layout placed it. Related XML attributes: Translation parametersThe float: The vertical position of this visualization relative to its top position, in pixels. Public void setTranslationZ (floating translationZ) Defines the depth location of this view in relation to its elevation. Related XML Attributes: Translation of parametersZ float public empty setVerticalFadingEdgeEnabled (vertical booleanaFadingEdgeEnabled) Set whether vertical edges should be faded when this view is scrolled vertically. Related XML attributes: android:requerFadingEdge Parameters VerticalFadingEdgeEnabled boolean: true if the vertical edges should be faded when the view is scrolled vertically See also: isVerticalFadingEdgeEnabled() public void setVerticalScrollBarEnabled (verticalized booleanscrollBarEnabled) Set whether the vertical scroll bar should be drawn or not. The scroll bar is not drawn by default. VerticalParametersScrollBarEnabled boolean: true if the vertical scrollbar should be painted See also: isVerticalScrollBarEnabled() Added in API level 1 Depreitated in api level 28 set of public voidWillNotCacheDrawing (boolean willNotCacheDrawing) This method was depreed at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of and layer update. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part View hierarchy or individual views it is recommended to create a screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. When the drawing cache of a view is activated, the drawing is redirected to an offscreen bitmap. Some visualizations, such as an ImageView, should be able to work around this mechanism if they already draw a single bitmap, to avoid unnecessary memory usage. Parameters willNotCacheDrawing boolean: True if this view does not cache your drawing, false public void setWillNotDraw (boolean willNotDraw) If this view does not make any drawing on its own, set this flag to allow for new optimizations. By default, this flag is not set in the view, but can be set on some display subclasses, such as viewgroup. Normally, if you replace onDraw (android.graphics.Canvas) you should clear this flag. WillNotDraw boolean parameters: Whether or not this View draws in its own set of public voidX (float x) Sets the visual position x of this view, in pixels. This equates to setting the translationX property as the difference between the past x value and the current left property. Parameters x fluctuation: The visual position x of this view, in pixels. Public void setY (float y) Sets the visual position y of this view, in pixels. This equates to setting the translationY property as the difference between the past y-value and the current top property. Parameters y float: The visual position of this view, in pixels. Public void setZ (float z) Sets the visual position z of this view, in pixels. This equates to setting the translationZ property as the difference between the past z value and the current elevation property. Parameters z float: The visual position z of this view, in pixels. Public Boolean showContextMenu () Shows the context menu for this view. Returns true Boolean if the context menu was shown, false otherwise see also: showContextMenu (float, float) show public booleanoContextMenu (float x, float y) Shows the context menu for this view anchored in the coordinate relative to the specified view. Parameters x fluctuation: The X coordinate in pixels relative to the view to which the menu is to be anchored, or Float#NaN to disable anchoring y float: the Y coordinate in pixels relative to the view to which the menu should be anchored, or Float#NaN to disable anchoring Returns true Boolean if the context menu was shown, false otherwise, empty public startAnimation (Animation) Start the specified animation now. Animation parameters the animation to start now public end boolean startDragAndDrop (Data ClipData, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) Starts a drag and drop drop When the application calls this method, it passes a View.DragShadowBuilder object to the system. The system calls DragShadowBuilder#onProvideShadowMetrics (Point, Point) to get metrics for the drag shadow, and then calls the DragShadowBuilder#onDrawShadow(Canvas) object to draw the drag shadow itself. Once the system has the drag shadow, it starts the drag-and-drop operation by sending drag events to all display objects in your application that are currently visible. It does this by calling the drag listener of the view object (an implementation of onDrag() or by calling the view object method in DragEvent(). Both are approved on a DragEvent object that has a DragEvent.getAction() value of DragEvent.ACTION_DRAG_STARTED. Your application can invoke the startDragAndDrop on any attached display object. The View object does not have to be used in View.DragShadowBuilder, nor does it have to be related to the View user selected to drag. Returns true boolean if the method completes successfully, or false if it fails anywhere. False return means that the system was unable to drag because of another ongoing operation or some other reasons. Public Boolean startNestedScroll (int axes) Start a nesteble parchment operation along the given axes. A view starting a nested scroll promises to fulfill the following contract: The view will call startNestedScroll when starting a parchment operation. In the case of a touch scroll, this corresponds to the MotionEvent#ACTION_DOWN action. In the case of touch scrolling, the nested parchment will automatically terminate in the same way as the ViewParent#requestDisallowInterceptTouchEvent(boolean). In the case of programmatic scrolling, the caller must explicitly call stopNestedScroll() to indicate the end of the nested scroll. If StartNestedScroll works, a co-op parent is found. If he returns false, the interlocutor can ignore the rest of this contract until the next scroll. Calling startNestedScroll while a nested scroll is already in progress will return true. At each incremental step of the parchment, the caller must invoke the NestedPreScroll dispatch once it has calculated the requested scroll delta. If it returns true, the nested scrolling parent has at least partially consumed the parchment, and the caller must adjust the amount by which it scrolls. After applying the rest of the parchment delta, the caller must invoke the NestedScroll dispatch, passing both the consumed delta and the delta without consumption. A nested scrolling parent can treat these values differently. See View, int, int, int, int, int). The return is true if a cooperative parent was found and nested scrolling has been enabled for the current gesture. Public empty stopNestedScroll () Stop a nested scroll in progress. Call this method when a nested is not in progress is harmless. See also: Public String toString () Returns a sequence representation of the object. In general, the toString toString method a string that texturedly represents this object. The result should be a concise but informative, easy-to-read representation. It is recommended that all subclasses override this method. The toString method for class object returns a sequence consisting of the name of the class of which the object is an instance, the signal character '@', and the unsigned hexamymic representation of the object's hash code. In other words, this method returns a sequence equal to the value of: getClass().getName() + '@' + Integer.toHexString(hashCode()) Returns A string string representation of the object. Matrix matrix transforms the input matrix to such an end that it maps local display coordinates to coordinates on the screen. Array of parameters: Input array to modify This value cannot be null. Matrix matrix transforms the input matrix in such a way that it maps coordinates on the screen to see local coordinates. Array of parameters: Input array to modify This value cannot be null. public void without scheduleDuleDrawable (Drawable who, Runnable what) Cancels a scheduled action on a drawable. Parameters that Drawable: The recipient of the action This value cannot be null. that Runnable: The action to cancel this value cannot be null. public empty without drawable (Drawable) schedule that clear any events associated with the given drawable. This can be used when selecting a new Drawable in a view, so that the previous one is completely unscheduled. Parameters that Drawable: The Drawable to clear. See also:

Added to the Deprecated Api level 1 in the boolean public level 28 API willNotCacheDrawing () This method has been deprecated at API level 28. The preview drawing cache became largely obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss of performance due to the cost of creating and updating the layer. In rare cases where cache layers are useful, such as for alpha animations, setLayerType (int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small portion of the display hierarchy or individual views, it is recommended to create a Screen from a Bitmap or Picture and call draw (android.graphics.Canvas) in the View. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and contour clipping. For UI screenshots for feedback reports or unit testing is recommended. Returns whether or not this view can cache your drawing. Returns true boolean if this does not cache your drawing, otherwise false public boolean willNotDraw () Returns whether or not this view is based on its own. Returns true Boolean if this view has nothing to draw, false methods otherwise protected protected protected protected awakensScrollBars (int startDelay, boolean invalidate) Trigger scroll bars to draw. When invoked, this method starts an animation to fade the scroll bars after a fixed delay. If a subclass provides animated scrolling, the initial delay must be equal to the duration of the scroll animation. The animation starts only if at least one of the scroll bars is enabled, as specified by isHorizontalScrollBarEnabled() and isVerticalScrollBarEnabled(). When the animation starts, this method returns true, and otherwise false. If the animation starts, this method calls invalidate() if the invalidated parameter is set to true; in this case, the caller should not call it invalidate(). This method must be invoked every time a subclass directly updates the scroll parameters. Parameters startDelay int: the delay, in milliseconds, after which the animation must begin; when the delay is 0, the animation immediately begins to invalidate Boolean: If this method should call invalidate returns true boolean if the animation is played, false booleanus otherwise protected awakenScrollBars (int startDelay) Trigger the scroll bars to draw. When invoked, this method starts an animation to fade the scroll bars after a fixed delay. If a subclass provides animated scrolling, the initial delay must be equal to the duration of the scroll animation. The animation starts only if at least one of the scroll bars is enabled, as specified by isHorizontalScrollBarEnabled() and isVerticalScrollBarEnabled(). When the animation starts, this method returns true, and otherwise false. If the animation starts, this method calls invalidate(); in this case, the caller should not call it invalidate(). This method must be invoked every time a subclass directly updates the scroll parameters. Parameters startDelay int: the delay, in milliseconds, after which the animation must begin; When the delay is 0, the animation immediately starts true Boolean returns if the animation is played, false boolean otherwise protected boolean awakenBars () Trigger the scroll bars to draw. When invoked, this method starts an animation to fade the scroll bars after a standard delay. If a subclass provides animated scrolling, the initial delay must be equal to the duration of the scroll animation. The animation starts only if at least one of the scroll bars is enabled, as specified by isHorizontalScrollBarEnabled() and isVerticalScrollBarEnabled(). When the animation starts, this method returns true, and otherwise false. If the animation starts, this method calls invalidate(); in this case, the caller should not call it invalidate(). This method must be invoked every time a subclass directly updates the scroll parameters. This method is automatically invoked by scrollBy (int, and scrollTo (int, int). Returns true boolean if the animation is played, fake empty dispatch protected Otherwise called by lot to draw the child's visualizations. This can be replaced by derived classes to gain control before your children were drawn (but after your own vision has been drawn). Parameter screen: The screen on which to draw the boolean protected display dispatchGenericFocusedEvent (MotionEvent event) Dispatches a generic motion event to the currently focused view. Do not call this method directly. Call dispatchGenericMotionEvent (android.view.MotionEvent) instead. MotionEvent parameter events: The motion event to be dispatched. True boolean returns if the event was handled by the view, false otherwise. Protected Boolean dispatchGenericPointerEvent (MotionEvent event) Send a generic motion event to the view under the first pointer. Do not call this method directly. Call dispatchGenericMotionEvent (android.view.MotionEvent) instead. MotionEvent parameter events: The motion event to be dispatched. True boolean returns if the event was handled by the view, false otherwise. Empty dispatch protectedSetActivated (boolean enabled) Dispatch set enabled for all children of this view. Boolean enabled parameters: The new state enabled See also: empty dispatch protectedSetPressed (boolean pressed) Set dispatches presses for all children of this view. Boolean pressed parameters: The new state pressed See also: empty dispatch protectedSetS elected (boolean selected) Set of selected dispatches for all children of this view. Boolean selected parameters: The new selected state See also: Protected empty dispatchVisibility (View changed View, visibility int) Dispatch a view visibility change in the view hierarchy. The ViewGroups must override to forward to their children. Changed ParametersViewView: The view whose visibility has changed. It could be this or an ancestral vision. This value cannot be null. visibility int: The new visibility of the changedView: VISIBLE, INVISIBLE or DISAPPEARED. The value is VISIBLE, INVISIBLE, or The Protected Void of Gone 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. If the View has a StateListAnimator, it is also called to run required state change animations. Be sure to connect to the superclass when replacing this function. If you override this method, you must call up the superclass implementation. See also: Added to the Deprecated Level 1 API in the boolean protected boolean fitSystemWindows (Rectified) API This method has been deprecated at API level 20. Starting with API 20 use dispatchApplyWindowInsets (android.view.WindowInsets) to apply insets to visualizations. Visualizations should replace the onApplyWindowInsets (android.view.WindowInsets) or use the setOnApplyWindowInsetsListener (android.view.View.View.View.ViewInsetsListener) to implement the of their own insets. Called by the view hierarchy when the content insets for a window have changed, to allow it to adjust its contents to fit within those windows. Content insets say that the space that the status bar, input method, and other system windows break the application window. You usually don't need need to with this function, since the default window decoration given to applications takes care of applying it to the window content. If you use SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN or SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION this will not be the case, and its contents may be placed under these system elements. You can then use this method within your view hierarchy if you have parts of your UI that you would like to ensure that they are not being covered. The default implementation of this method simply applies the content insets to the view fill, consuming that content (modifying the insets to be 0) and returning true. This behavior is out by default, but can be activated through setFitsSystemWindows(boolean). Crossing this function through the hierarchy is the first depth. The same content insets object is propagated by the hierarchy, so that any changes made to it will be seen by all subsequent views (potentially including those above in the hierarchy, since this is a depth crossing). The first vision that returns true will abort the entire crossing. 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 contents at all edges. If you need a more complicated layout (such as two system adjustment windows of different visualizations, one at the top of the window and one at the bottom), you can override the method and handle the insets as you like. Note that the insets provided by the structure are always relative to the distant edges of the window, not counting the location of the view call within that window. (In fact, when this method is called, you still don't know where the layout will place the view, as it is done before the layout happens.) Note: Unlike many display methods, there is no sending phase for this call. If you are replacing it in a ViewGroup and want to allow the call to continue to your children, you should be sure to call the super implementation. Here is a sample layout that makes use of mounting system windows to have controls for a video preview placed inside the window decoration that it hides and shows. This can be used with code such as the second sample (video player) shown in setSystemUiVisibility(int). <FrameLayout xmlns: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></view> <FrameLayout android:layout_width=match_parent android:layout_height=match_parent android:fitssystemwindows=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:id=@+id/title android:layout_width=match_parent android:layout_height=wrap_content android:layout_gravity=top|center_horizontal android:textcolor=#ffffffff></TextView android:id=@+id/title android:layout_width=match_parent android:layout_height=wrap_content android:layout_gravity=top|center_horizontal android:textColor=#ffffffff ></FrameLayout></FrameLayout> ></FrameLayout></FrameLayout> android:textAppearance=?android:attr/textAppearanceLarge android:gravity=left android:endding=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></Button> <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></SeekBar> Parameters insets Rect: Insets of current window content. Before i Build.VERSION_CODES. JELLY_BEAN you should not modify the insets or else you and Android will be unhappy. True Boolean returns if this view applied the insets and should not continue propagating further down the hierarchy, false otherwise. protected float (getBottomFadingEdgeStrength) Returns the strength, or intensity, of the faded bottom edge. The force is a value between 0.0 (no fading) and 1.0 (total fade). The default implementation returns 0.0 or 1.0, but no value between them. Subclasses should override this method to provide a smoother fade transition when scrolling occurs. The returns fluctuate the background intensity disappear as a float between 0.0f and 1.0f protected int getHorizontalScrollbarHeight () Returns the height of the horizontal scroll bar. Returns int The pixel height of the horizontal scroll bar or 0 if there is no horizontal scroll bar. Floating protected getLeftFadingEdgeStrength () Returns the strength, or intensity, of the left faded edge. The force is a value between 0.0 (no fading) and 1.0 (total fade). The default implementation returns 0.0 or 1.0, but no value between them. Subclasses should override this method to provide a smoother fade transition when scrolling occurs. The returns fluctuate the intensity of the left fade as a float between 0.0f and 1.0f protected floating getRightFadingEdgeStrength () Returns the strength, or intensity, of the faded right edge. The force is a value between 0.0 (no fading) and 1.0 (total fade). The default implementation returns 0.0 or 1.0, but no value between them. Subclasses should override this method to provide a smoother fade transition when scrolling occurs. Returns fluctuate the intensity of the right fade as a float between 0.0f and 1.0f protected int getSuggestedMinimumHeight () Returns the suggested minimum height that the view should use. This returns the maximum of the minimum view height and the minimum bottom height (Drawable.getMinimumHeight()). When used in onMeasure (int, int), the caller must still ensure that the returned height is within the parent's requirements. Return int The suggested minimum view height. int protected getSuggestedMinimumWidth () Returns the suggested minimum width that the view should use. This returns the maximum of the minimum display width and the minimum width of the background When used in int), the caller must still ensure that the returned width is within the parent's requirements. Returns int The suggested minimum width of the view. protected float (getTopFadingEdgeStrength) Returns the strength, or intensity, of the faded top edge. The force is a value between 0.0 (no fading) and 1.0 (total fade). The default implementation returns 0.0 or 1.0, but no value between them. Subclasses should override this method to provide a smoother fade transition when scrolling occurs. The returns fluctuate the intensity of the top disappear as a float between 0.0f and 1.0f int protected getWindowAttachCount () Returns int The number of times this view was attached to a window protected by booddingsaquindoSOrequired () If the View draws content within its upholstery and allows faded edges, it needs to support upholstery offsets. Fill offsets are added to faded edges to extend the length of the fade so that it covers pixels drawn within the upholstery. The subclasses in this class should override this method if they need to draw content within the fill. Returns boolean true if the offset fill should be applied, false otherwise. int static protected[] mergeDEres (int[] baseState, int[] additionalState) merges their own state values into additionalState in the base state state of the basestate state that were returned by onCreateDrawableState (int). Base parameters: The base state values returned by the onCreateDrawableState (int), which will be modified to also maintain their own additional state values. additional int: The additional values of the state that you would like to add to the base state; this array is not modified. Returns int[] As a convenience, the basestate array that you originally passed to the function is returned. See also: noCreateDrawableState (int) empty protected onAnimationEnd () 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(); If you override this method, you must call up the superclass implementation. See also: empty setAnimation (android.view.animation.Animation)getAnimation() protected onAnimationStart () 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(); If you override this method, you must call up the superclass implementation. See also: empty setAnimation (android.view.animation.Animation)getAnimation() protected in AttachedToWindow () 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), however, it can be called at any before the first onDraw -- including before or after onMeasure (int, int). If you override this method, you must call up the superclass implementation. See also: Empty protected in Scanned Configuration (New Config Configuration) Called when when when the current configuration of the resources used by the application has changed. You can use it to decide when to reload features that can be changed based on orientation and other configuration characteristics. You only need to use this if you are not relying on the normal activity engine to recreate the activity instance after a configuration change. New Settings Config Parameters: The new feature configuration. Protected void in the CreateContextMenu (ContextMenu menu) Visualizations should implement this if the view itself adds items to the context menu. ContextMenu menu parameters: The context menu to fill int protected[] onCreateDrawableState (int extraSpace) generate the new Drawable state for this view. This is called by the display system when the cached Drawable state is determined to be invalid. To retrieve the current state, you must use getDrawableState(). ExtraSpace int parameters: If non-zero, this is the number of extra entries you would like in the returned array in which you can put your own states. Returns int[] Returns an array that maintains the current Drawable view state. See also: mergeDes (int[], int[]) empty protected inDetachedFromWindow () This is called when the view is separated from a window. At this point there is no more surface to draw. If you override this method, you must call up the superclass implementation. See also: Empty protected in DisplayHint (hint int) Gives this view a hint about whether it is displayed or not. For example, when a View moves off the screen, it might receive a display hint indicating that the view is not displayed. Applications should not trust this tip because there is no guarantee that they will receive one. empty protected in Draw (screen screen) Implement this to make your drawing. Parameters screen: The screen on which the background will be drawn protected final empty in DrawScrollBars (Screen Screen) Request the drawing of the horizontal and vertical scroll bar. Scroll bars are painted only if they are agreed first. Parameter screen: The screen on which to draw the scroll bars See also: empty protected in FinishInflate () Finish inflating an XML view. This is called the last phase of inflation, after all children's visions have been added. Even if the subclass overlaps with FinishInflate, they should always be sure to call the super method, so that we are called. If you override this method, you must call up the superclass implementation. empty protected in FocusChanged (gain booleanfoba, direction int, Rect formerlyFocusedRect) Called by the display system when the focus state of this view changes. When the focus change event is caused by directional navigation, the direction and previouslyfocusedRect provide an insight into where the focus is coming from. When replacing, be sure to call up the super class to standard focus handling occurs. If you override this method, you must call up the superclass superclass Parameters gain Boolean Focus: True if the View has focus; false otherwise. int direction: The direction focus has changed when the Focus() request is called to give this view focus. The 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 pattern. The value is FOCUS_BACKWARD, FOCUS_FORWARD, FOCUS_LEFT, FOCUS_UP, FOCUS_RIGHT, or FOCUS_DOWN previouslyFocusedRect Retct: The rectangle, in the coordinate system of this view, of the previously focused view. If applicable, this will be passed as finer information about where the focus comes from (in addition to direction). It will be null otherwise. This value can be null. Empty protected in the layout (boolean changed, int left, int top, int right, bottom int) Called 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 the layout of each of your children. Boolean changed parameters: This is a new size or position for this left view int: Left position, relative to the top of the parents: Top position, relative to the right int of the parents: Right position, relative to the bottom of the parent: Lower position, relative to the empty protected by the parents naMeasure (width of the intMeasureSpec, intMeasureSpec height) Measure the view and its contents to determine the measured width and the measured height. This method is invoked by measure (int, int) and must be replaced by subclasses to provide accurate and efficient measurement of its content. CONTRACT: When overwriting this method, you must call setMeasuredDimension (int, int) to store the measured width and height of this view. Failure to do so will trigger a full-release IllegalStateException (int, int). Calling the superclass onMeasure (int, int) is a valid use. The implementation of the measure pattern base class for the background size, unless a larger size is allowed by MeasureSpec. Subclasses should replace onMeasure (int, int) to provide better measurements of their content. If this method is overridden, it is the responsibility of the subclass to ensure that the measured height and width are at least the minimum height and width of the view (getSuggestedMinimumHeight() and getSuggestedMinimumWidth()). Width ParameterMeasureSpec int: Horizontal space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec. heightMeasureSpec int: Vertical space requirements imposed by the parent. The requirements are encoded with View.MeasureSpec. empty protected 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-scrolloperation operation. scrollX int parameters: New X scroll value in scrollY int pixels: New y scroll value in boolean sx pinned pixels: True if scrollX was pinned on Fixed over-scroll scroll ing limitY boolean: True if scrollY has been pinned to an over-scroll over-scroll limit Parcelable onSaveInstanceState () Hook allowing a view to generate a representation of its internal state that can be used later to create a new instance with that same state. This state should contain only information that is not persistent or cannot be rebuilt later. For example, you will never store your current position on the screen because this will be calculated again when a new instance of the view is placed in your display hierarchy. Some examples of things you can store here: the current cursor position in a text view (but usually not the text itself, since it is stored in a content provider or other persistent storage), the item currently selected in a list view. If you override this method, you must call up the superclass implementation. Parcelable returns a Parcelable object containing the current dynamic state of the view or null if there is nothing interesting to save. empty protected onScrollChanged (int l, int t, int oldl, int oldt) That is called in response to an internal scroll in this view (i.e. the view rolled its own content). This is typically as a result of scrollBy (int, int) or scrollTo (int, int) being called. L int parameters: Origin of the current horizontal parchment. t int: Origin of the current vertical parchment. oldl int: Origin of the anterior horizontal parchment. oldt int: Origin of the anterior vertical parchment. protected boolean onSetAlpha (int alpha) invoked if there is a Transformation that involves alpha. Subclass that can draw with the specified alpha must return true, and then respect that alpha when its onDraw() is called. If this returns false, then the view can be redirected to an offscreen buffer to fulfill the request, which will be fine, but may be slower than if the subclass handles it internally. The default implementation returns false. Alpha int parameters: The alpha (0.255) to apply to the view drawing returns true Boolean if the view can draw with the specified alpha. empty protected onSizeChanged (int w, int h, int oldw, int oldh) This is called during the layout when the size of this view has changed. If you have just been added to the preview hierarchy, you are called with the old values of 0. Parameters w int: Current width of this view. h int: Current height of this view. oldw int: Old width of this view. oldh int: Ancient height of this view. Empty protected overSspouted visibility (View changedView, visibility int) Called when visibility of the view or an ancestor of the view has changed. Changed ParametersViewView: The view whose visibility has changed. It could be this or an ancestral vision. This value cannot be null. visibility int: The new visibility, one of VISIBLE, INVISIBLE or Gone. The value is visible, invisible, or empty protected by Gone in The Altered Visibility int) Call when the window containing changes its visibility (between GONE, INVISIBLE, and VISIBLE). Note that this tells you whether or not your window is being visible to the window manager; it doesn't say whether or or your window is obscured by other windows on the screen, even if it is visible. protected boolean onScrollBy (int deltaX, int deltaY, int scrollX, int scrollY, int scrollRangeX, int parchmentRangeY, int maxOverScrollX, int maxOverScrollY, boolean isTouchEvent) Scroll the display with default behavior to scroll beyond normal content limits. Opinions that call this method should override onOverScrolled (int, int, boolean, boolean) to respond to the results of an over-scrolling operation. Visualizations can use this method to handle any touch-based scrolling or launch. deltaX int: Change in X in pixels deltaY int: Change in Y pixels in scrollX int: Current value of parchment X in pixels before applying deltaX scrollY int: Current value of parchment Y in pixels before applying delta Parchment Desarmes: Maximum range of content scrolling along the xrangey int axis scroll: Maximum range of content scrolling along the Y axis maxOverScrollX int : Number of pixels to overscroll in any direction along the X axis. maxOverScrollY int: Number of pixels to overscroll in any direction along the Y axis. isTouchEvent boolean: true if this parchment operation is the result of a touch event. True Boolean returns if the scroll has been fixed to an over-scrolling boundary along any axis, otherwise false. Soudible who if your display subclass is displaying its own Drawable objects, it must override this function and return true to any Drawable it is displaying. This allows animations for these drawables to be programmed. Be sure to call up the super class when replacing this function. If you override this method, you must call up the superclass implementation. Parameters that Drawable: The Drawable to check. Return true if it is one that you are viewing, then return the result of calling to the super class. This value cannot be null. Boolean boolean returns if true of what drawable is being displayed in the view; more false and is not allowed to animate. Animate.