public class
TextView
extends View
implements

Summary: Nested Classes | XML Attrs | Inherited XML Attrs | Inherited Constants | Inherited Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]

Added in API level 1

ViewTreeObserver.OnPreDrawListener

java.lang.Object
↳ android.view.View
↳ android.widget.TextView

▶ Known Direct Subclasses
Button, CheckedTextView, Chronometer, DigitalClock, EditText, TextClock

▶ Known Indirect Subclasses
AutoCompleteTextView, CheckBox, CompoundButton, ExtractEditText, MultiAutoCompleteTextView, RadioButton, Switch, ToggleButton

Class Overview

Displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing; see `EditText (/reference/android/widget/EditText.html)` for a subclass that configures the text view for editing.

To allow users to copy some or all of the TextView's value and paste it somewhere else, set the XML attribute `android:textIsSelectable (/reference/android/R.styleable.html#TextView_textIsSelectable)` to "true" or call `setTextIsSelectable(true) (/reference/android/widget/TextView.html#setTextIsSelectable(boolean))`. The `textIsSelectable` flag allows users to make selection gestures in the TextView, which in turn triggers the system's built-in copy/paste controls.

XML attributes

See `TextView Attributes (/reference/android/R.styleable.html#TextView)`, `View Attributes (/reference/android/R.styleable.html#View)`

Summary

Nested Classes

enum TextView.BufferType

interface TextView.OnEditorActionListener	Interface definition for a callback to be invoked when an action is performed on the editor.
class TextView.SavedState	User interface state that is stored by TextView for implementing `onSaveInstanceState()`.

XML Attributes

Attribute Name	*Related Method*
android:autoLink	setAutoLinkMask(int)
android:autoText	setKeyListener(KeyListener)
android:bufferType	setText(CharSequence,TextView.BufferType)
android:capitalize	setKeyListener(KeyListener)
android:cursorVisible	setCursorVisible(boolean)
android:digits	setKeyListener(KeyListener)
android:drawableBottom	setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)
android:drawableEnd	setCompoundDrawablesRelativeWithIntrinsicBounds(int,int,int,i
android:drawableLeft	setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)
android:drawablePadding	setCompoundDrawablePadding(int)
android:drawableRight	setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)

android:drawableStart	setCompoundDrawablesRelativeWithIntrinsicBounds(int,int,int,i
android:drawableTop	setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)
android:editable	
android:editorExtras	setInputExtras(int)
android:ellipsize	setEllipsize(TextUtils.TruncateAt)
android:ems	setEms(int)
android:fontFamily	setTypeface(Typeface)
android:freezesText	setFreezesText(boolean)
android:gravity	setGravity(int)
android:height	setHeight(int)
android:hint	setHint(int)
android:imeActionId	setImeActionLabel(CharSequence,int)
android:imeActionLabel	setImeActionLabel(CharSequence,int)

android:imeOptions	setImeOptions(int)
android:includeFontPadding	setIncludeFontPadding(boolean)
android:inputMethod	setKeyListener(KeyListener)
android:inputType	setRawInputType(int)
android:lineSpacingExtra	setLineSpacing(float,float)
android:lineSpacingMultiplier	setLineSpacing(float,float)
android:lines	setLines(int)
android:linksClickable	setLinksClickable(boolean)
android:marqueeRepeatLimit	setMarqueeRepeatLimit(int)
android:maxEms	setMaxEms(int)
android:maxHeight	setMaxHeight(int)
android:maxLength	setFilters(InputFilter)
android:maxLines	setMaxLines(int)
android:maxWidth	setMaxWidth(int)

android:minEms	setMinEms(int)
android:minHeight	setMinHeight(int)
android:minLines	setMinLines(int)
android:minWidth	setMinWidth(int)
android:numeric	setKeyListener(KeyListener)
android:password	setTransformationMethod(TransformationMethod)
android:phoneNumber	setKeyListener(KeyListener)
android:privateImeOptions	setPrivateImeOptions(String)
android:scrollHorizontally	setHorizontallyScrolling(boolean)
android:selectAllOnFocus	setSelectAllOnFocus(boolean)
android:shadowColor	setShadowLayer(float,float,float,int)
android:shadowDx	setShadowLayer(float,float,float,int)
android:shadowDy	setShadowLayer(float,float,float,int)
android:shadowRadius	setShadowLayer(float,float,float,int)
android:singleLine	setTransformationMethod(TransformationMethod)

android:text	setText(CharSequence,TextView.BufferType)
android:textAllCaps	setAllCaps(boolean)
android:textAppearance	
android:textColor	setTextColor(int)
android:textColorHighlight	setHighlightColor(int)
android:textColorHint	setHintTextColor(int)
android:textColorLink	setLinkTextColor(int)
android:textIsSelectable	isTextSelectable()
android:textScaleX	setTextScaleX(float)
android:textSize	setTextSize(int,float)
android:textStyle	setTypeface(Typeface)
android:typeface	setTypeface(Typeface)
android:width	setWidth(int)

Inherited XML Attributes [Expand]

▸ From class android.view.View

Inherited Constants [Expand]

▸ From class android.view.View

Inherited Fields [Expand]

▸ From class android.view.View

Public Constructors

TextView (Context context)
TextView (Context context, AttributeSet attrs)
TextView (Context context, AttributeSet attrs, int defStyle)

Public Methods

void	addTextChangedListener (TextWatcher watcher) Adds a TextWatcher to the list of those whose methods are cal
final void	append (CharSequence text) Convenience method: Append the specified text to the TextVie BufferType.EDITABLE if it was not already editable.
void	append (CharSequence text, int start, int end) Convenience method: Append the specified text slice to the Te

BufferType.EDITABLE if it was not already editable.

void beginBatchEdit()

boolean bringPointIntoView(int offset)
Move the point, specified by the offset, into the view if it is nee

void cancelLongPress()
Cancels a pending long press.

void clearComposingText()
Use `BaseInputConnection.removeComposingSpans()` t
view.

void computeScroll()
Called by a parent to request that a child update its values for

void debug(int depth)
Prints information about this view in the log output, with the ta

boolean didTouchFocusSelect()
Returns true, only while processing a touch gesture, if the initia
text view and as a result its selection changed.

void endBatchEdit()

boolean extractText(ExtractedTextRequest request, ExtractedText outTex
If this TextView contains editable content, extract a portion of
outText.

void findViewsWithText(ArrayList<View> outViews, CharSequence se
Finds the Views that contain given text.

final int getAutoLinkMask()
Gets the autolink mask of the text.

int getBaseline()
Return the offset of the widget's text baseline from the widget'

int getCompoundDrawablePadding()
Returns the padding between the compound drawables and th

Drawable[] getCompoundDrawables()
Returns drawables for the left, top, right, and bottom borders.

Drawable[] getCompoundDrawablesRelative()
Returns drawables for the start, top, end, and bottom borders.

int getCompoundPaddingBottom()
Returns the bottom padding of the view, plus space for the bot

int getCompoundPaddingEnd()
Returns the end padding of the view, plus space for the end Dr

int getCompoundPaddingLeft()
Returns the left padding of the view, plus space for the left Dra

int getCompoundPaddingRight()
Returns the right padding of the view, plus space for the right D

int	getCompoundPaddingStart () Returns the start padding of the view, plus space for the start D
int	getCompoundPaddingTop () Returns the top padding of the view, plus space for the top Dra
final int	getCurrentHintTextColor () Return the current color selected to paint the hint text.
final int	getCurrentTextColor () Return the current color selected for normal text.
ActionMode.Callback	getCustomSelectionActionModeCallback () Retrieves the value set in `setCustomSelectionActionMod`
Editable	getEditableText () Return the text the TextView is displaying as an Editable objec
TextUtils.TruncateAt	getEllipsize () Returns where, if anywhere, words that are longer than the viev
CharSequence	getError () Returns the error message that was set to be displayed with s was set or if it the error was cleared by the widget after user in
int	getExtendedPaddingBottom () Returns the extended bottom padding of the view, including bo space to keep more than maxLines of text from showing.
int	getExtendedPaddingTop () Returns the extended top padding of the view, including both tl keep more than maxLines of text from showing.
InputFilter[]	getFilters () Returns the current list of input filters.
void	getFocusedRect (Rect r) When a view has focus and the user navigates away from it, th rectangle filled in by this method.
boolean	getFreezesText () Return whether this text view is including its entire text conten
int	getGravity () Returns the horizontal and vertical alignment of this TextView.
int	getHighlightColor ()
CharSequence	getHint () Returns the hint that is displayed when the text of the TextViev
final ColorStateList	getHintTextColors ()
int	getImeActionId () Get the IME action ID previous set with `setImeActionLabel`

CharSequence	getImeActionLabel () Get the IME action label previous set with `setImeActionLab`
int	getImeOptions () Get the type of the IME editor.
boolean	getIncludeFontPadding () Gets whether the TextView includes extra top and bottom padc normal ascent and descent.
Bundle	getInputExtras (boolean create) Retrieve the input extras currently associated with the text vie
int	getInputType () Get the type of the editable content.
final KeyListener	getKeyListener ()
final Layout	getLayout ()
int	getLineBounds (int line, Rect bounds) Return the baseline for the specified line (0...getLineCount() - 1 bottom extents of the specified line in it.
int	getLineCount () Return the number of lines of text, or 0 if the internal Layout ha
int	getLineHeight ()
float	getLineSpacingExtra () Gets the line spacing extra space
float	getLineSpacingMultiplier () Gets the line spacing multiplier
final ColorStateList	getLinkTextColors ()
final boolean	getLinksClickable () Returns whether the movement method will automatically be s `setAutoLinkMask(int)` has been set to nonzero and links a
int	getMarqueeRepeatLimit () Gets the number of times the marquee animation is repeated.
int	getMaxEms ()
int	getMaxHeight ()
int	getMaxLines ()
int	getMaxWidth ()
int	getMinEms ()
int	getMinHeight ()
int	getMinLines ()
int	getMinWidth ()
final MovementMethod	getMovementMethod ()
int	getOffsetForPosition (float x, float y) Get the character offset closest to the specified absolute posit
TextPaint	getPaint ()

int	getPaintFlags ()
String	getPrivateImeOptions () Get the private type of the content.
int	getSelectionEnd () Convenience for `getSelectionEnd(CharSequence)`.
int	getSelectionStart () Convenience for `getSelectionStart(CharSequence)`.
int	getShadowColor ()
float	getShadowDx ()
float	getShadowDy ()
float	getShadowRadius () Gets the radius of the shadow layer.
CharSequence	getText () Return the text the TextView is displaying.
static int	getTextColor (Context context, TypedArray attrs, int def) Returns the default color from the TextView_textColor attribute from the TextAppearance_textColor from the TextView_textApp set directly.
final ColorStateList	getTextColors () Gets the text colors for the different states (normal, selected, f
static ColorStateList	getTextColors (Context context, TypedArray attrs) Returns the TextView_textColor attribute from the TypedArray, TextView_textAppearance attribute, if TextView_textColor was
Locale	getTextLocale () Get the default `Locale` of the text in this TextView.
float	getTextScaleX ()
float	getTextSize ()
int	getTotalPaddingBottom () Returns the total bottom padding of the view, including the bot than maxLines from showing, and the vertical offset for gravity
int	getTotalPaddingEnd () Returns the total end padding of the view, including the end Dr
int	getTotalPaddingLeft () Returns the total left padding of the view, including the left Dra
int	getTotalPaddingRight () Returns the total right padding of the view, including the right D
int	getTotalPaddingStart () Returns the total start padding of the view, including the start I
int	getTotalPaddingTop () Returns the total top padding of the view, including the top Dra maxLines from showing, and the vertical offset for gravity, if ar

final TransformationMethod	getTransformationMethod ()
Typeface	getTypeface ()
URLSpan[]	getUrls () Returns the list of URLSpans attached to the text (by Linkify
boolean	hasOverlappingRendering () Returns whether this View has content which overlaps.
boolean	hasSelection () Return true iff there is a selection inside this text view.
void	invalidateDrawable (Drawable drawable) Invalidates the specified Drawable.
boolean	isCursorVisible ()
boolean	isInputMethodTarget () Returns whether this text view is a current input method target
boolean	isSuggestionsEnabled () Return whether or not suggestions are enabled on this TextVie
boolean	isTextSelectable () Returns the state of the textIsSelectable flag (See setTe
void	jumpDrawablesToCurrentState () Call Drawable.jumpToCurrentState() on all Drawable ob
int	length () Returns the length, in characters, of the text managed by this T
boolean	moveCursorToVisibleOffset () Move the cursor, if needed, so that it is at an offset that is visib
void	onBeginBatchEdit () Called by the framework in response to a request to begin a ba beginBatchEdit().
boolean	onCheckIsTextEditor () Check whether the called view is a text editor, in which case it input window for it.
void	onCommitCompletion (CompletionInfo text) Called by the framework in response to a text completion from InputConnection.commitCompletion().
void	onCommitCorrection (CorrectionInfo info) Called by the framework in response to a text auto-correction (the current input method, provided by it calling commitCorre InputConnection.commitCorrection()}.
InputConnection	onCreateInputConnection (EditorInfo outAttrs) Create a new InputConnection for an InputMethod to interact v
boolean	onDragEvent (DragEvent event) Handles drag events sent by the system following a call to sta

void	onEditorAction (int actionCode) Called when an attached input method calls InputConnecti
void	onEndBatchEdit () Called by the framework in response to a request to end a batc endBatchEdit().
void	onFinishTemporaryDetach () Called after onStartTemporaryDetach() when the contain
boolean	onGenericMotionEvent (MotionEvent event) Implement this method to handle generic motion events.
void	onInitializeAccessibilityEvent (AccessibilityEvent event) Initializes an AccessibilityEvent with information about t
void	onInitializeAccessibilityNodeInfo (AccessibilityNodeInfo info) Initializes an AccessibilityNodeInfo with information abc
boolean	onKeyDown (int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyDown KEYCODE_DPAD_CENTER or KEYCODE_ENTER is released, if th
boolean	onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyMult event).
boolean	onKeyPreIme (int keyCode, KeyEvent event) Handle a key event before it is processed by any input method
boolean	onKeyShortcut (int keyCode, KeyEvent event) Called on the focused view when a key shortcut event is not ha
boolean	onKeyUp (int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyUp() KEYCODE_DPAD_CENTER or KEYCODE_ENTER is released.
void	onPopulateAccessibilityEvent (AccessibilityEvent event) Called from dispatchPopulateAccessibilityEvent(Ac View to populate the accessibility event with its text content.
boolean	onPreDraw () Callback method to be invoked when the view tree is about to b
boolean	onPrivateIMECommand (String action, Bundle data) Called by the framework in response to a private command fro InputConnection.performPrivateCommand().
void	onRestoreInstanceState (Parcelable state) Hook allowing a view to re-apply a representation of its interna onSaveInstanceState().
void	onRtlPropertiesChanged (int layoutDirection) Called when any RTL property (layout direction or text direction
Parcelable	onSaveInstanceState () Hook allowing a view to generate a representation of its interna instance with that same state.

void	onScreenStateChanged (int screenState) This method is called whenever the state of the screen this vie
void	onStartTemporaryDetach () This is called when a container is going to temporarily detach a `ViewGroup.detachViewFromParent`.
boolean	onTextContextMenuItem (int id) Called when a context menu option for the text view is selecte
boolean	onTouchEvent (MotionEvent event) Implement this method to handle touch screen motion events.
boolean	onTrackballEvent (MotionEvent event) Implement this method to handle trackball motion events.
void	onWindowFocusChanged (boolean hasWindowFocus) Called when the window containing this view gains or loses fo
boolean	performAccessibilityAction (int action, Bundle arguments) Performs the specified accessibility action on the view.
boolean	performLongClick () Call this view's OnLongClickListener, if it is defined.
void	removeTextChangedListener (TextWatcher watcher) Removes the specified TextWatcher from the list of those who text changes.
void	sendAccessibilityEvent (int eventType) Sends an accessibility event of the given type.
void	setAllCaps (boolean allCaps) Sets the properties of this field to transform input to ALL CAPS
final void	setAutoLinkMask (int mask) Sets the autolink mask of the text.
void	setCompoundDrawablePadding (int pad) Sets the size of the padding between the compound drawables
void	setCompoundDrawables (Drawable left, Drawable top, Drawable r Sets the Drawables (if any) to appear to the left of, above, to th
void	setCompoundDrawablesRelative (Drawable start, Drawable top, D Sets the Drawables (if any) to appear to the start of, above, to t
void	setCompoundDrawablesRelativeWithIntrinsicBounds (Drawable s Sets the Drawables (if any) to appear to the start of, above, to t
void	setCompoundDrawablesRelativeWithIntrinsicBounds (int start, in Sets the Drawables (if any) to appear to the start of, above, to t
void	setCompoundDrawablesWithIntrinsicBounds (Drawable left, Drav Sets the Drawables (if any) to appear to the left of, above, to th
void	setCompoundDrawablesWithIntrinsicBounds (int left, int top, int r Sets the Drawables (if any) to appear to the left of, above, to th

void	setCursorVisible (boolean visible) Set whether the cursor is visible.
void	setCustomSelectionActionModeCallback (ActionMode.Callback a If provided, this ActionMode.Callback will be used to create the this View.
final void	setEditableFactory (Editable.Factory factory) Sets the Factory used to create new Editables.
void	setEllipsize (TextUtils.TruncateAt where) Causes words in the text that are longer than the view is wide t
void	setEms (int ems) Makes the TextView exactly this many ems wide
void	setEnabled (boolean enabled) Set the enabled state of this view.
void	setError (CharSequence error) Sets the right-hand compound drawable of the TextView to the be displayed in a popup when the TextView has focus.
void	setError (CharSequence error, Drawable icon) Sets the right-hand compound drawable of the TextView to the will be displayed in a popup when the TextView has focus.
void	setExtractedText (ExtractedText text) Apply to this text view the given extracted text, as previously re `extractText(ExtractedTextRequest, ExtractedTex`
void	setFilters (InputFilter[] filters) Sets the list of input filters that will be used if the buffer is Edit
void	setFreezesText (boolean freezesText) Control whether this text view saves its entire text contents wh state such as cursor position.
void	setGravity (int gravity) Sets the horizontal alignment of the text and the vertical gravit the TextView beyond what is required for the text itself.
void	setHeight (int pixels) Makes the TextView exactly this many pixels tall.
void	setHighlightColor (int color) Sets the color used to display the selection highlight.
final void	setHint (CharSequence hint) Sets the text to be displayed when the text of the TextView is e
final void	setHint (int resid) Sets the text to be displayed when the text of the TextView is e
final void	setHintTextColor (ColorStateList colors) Sets the color of the hint text.
final void	setHintTextColor (int color) Sets the color of the hint text for all the states (disabled, focus

void	setHorizontallyScrolling (boolean whether) Sets whether the text should be allowed to be wider than the V
void	setImeActionLabel (CharSequence label, int actionId) Change the custom IME action associated with the text view, w `actionLabel` and `actionId` when it has focus.
void	setImeOptions (int imeOptions) Change the editor type integer associated with the text view, w when it has focus.
void	setIncludeFontPadding (boolean includepad) Set whether the TextView includes extra top and bottom paddi normal ascent and descent.
void	setInputExtras (int xmlResId) Set the extra input data of the text, which is the `TextBoxAttr` creating an input connection.
void	setInputType (int type) Set the type of the content with a constant as defined for `inpu`
void	setKeyListener (KeyListener input) Sets the key listener to be used with this TextView.
void	setLineSpacing (float add, float mult) Sets line spacing for this TextView.
void	setLines (int lines) Makes the TextView exactly this many lines tall.
final void	setLinkTextColor (ColorStateList colors) Sets the color of links in the text.
final void	setLinkTextColor (int color) Sets the color of links in the text.
final void	setLinksClickable (boolean whether) Sets whether the movement method will automatically be set t `setAutoLinkMask(int)` has been set to nonzero and links a
void	setMarqueeRepeatLimit (int marqueeLimit) Sets how many times to repeat the marquee animation.
void	setMaxEms (int maxems) Makes the TextView at most this many ems wide
void	setMaxHeight (int maxHeight) Makes the TextView at most this many pixels tall.
void	setMaxLines (int maxlines) Makes the TextView at most this many lines tall.
void	setMaxWidth (int maxpixels) Makes the TextView at most this many pixels wide
void	setMinEms (int minems) Makes the TextView at least this many ems wide

void	setMinHeight (int minHeight) Makes the TextView at least this many pixels tall.
void	setMinLines (int minlines) Makes the TextView at least this many lines tall.
void	setMinWidth (int minpixels) Makes the TextView at least this many pixels wide
final void	setMovementMethod (MovementMethod movement) Sets the movement method (arrow key handler) to be used for
void	setOnEditorActionListener (TextView.OnEditorActionListener l) Set a special listener to be called when an action is performed
void	setPadding (int left, int top, int right, int bottom) Sets the padding.
void	setPaddingRelative (int start, int top, int end, int bottom) Sets the relative padding.
void	setPaintFlags (int flags) Sets flags on the Paint being used to display the text and reflo
void	setPrivateImeOptions (String type) Set the private content type of the text, which is the EditorIn in when creating an input connection.
void	setRawInputType (int type) Directly change the content type integer of the text view, witho
void	setScroller (Scroller s)
void	setSelectAllOnFocus (boolean selectAllOnFocus) Set the TextView so that when it takes focus, all the text is sele
void	setSelected (boolean selected) Changes the selection state of this view.
void	setShadowLayer (float radius, float dx, float dy, int color) Gives the text a shadow of the specified radius and color, the s
void	setSingleLine () Sets the properties of this field (lines, horizontally scrolling, tra
void	setSingleLine (boolean singleLine) If true, sets the properties of this field (number of lines, horizor single-line input; if false, restores these to the default condition
final void	setSpannableFactory (Spannable.Factory factory) Sets the Factory used to create new Spannables.
final void	setText (int resid)
final void	setText (char[] text, int start, int len) Sets the TextView to display the specified slice of the specifiec
final void	setText (int resid, TextView.BufferType type)
final void	setText (CharSequence text) Sets the string value of the TextView.

	setText (CharSequence text, TextView.BufferType type) Sets the text that this TextView is to display (see `setText(Ch` in a styleable/spannable buffer and whether it is editable.
void	setTextAppearance (Context context, int resid) Sets the text color, size, style, hint color, and highlight color fro
void	setTextColor (ColorStateList colors) Sets the text color.
void	setTextColor (int color) Sets the text color for all the states (normal, selected, focused)
void	setTextIsSelectable (boolean selectable) Sets whether the content of this view is selectable by the user.
final void	setTextKeepState (CharSequence text) Like `setText(CharSequence)`, except that the cursor positi
final void	setTextKeepState (CharSequence text, TextView.BufferType type) Like `setText(CharSequence, android.widget.TextVi` position (if any) is retained in the new text.
void	setTextLocale (Locale locale) Set the default `Locale` of the text in this TextView to the giver
void	setTextScaleX (float size) Sets the extent by which text should be stretched horizontally.
void	setTextSize (float size) Set the default text size to the given value, interpreted as "scal
void	setTextSize (int unit, float size) Set the default text size to a given unit and value.
final void	setTransformationMethod (TransformationMethod method) Sets the transformation that is applied to the text that this Tex
void	setTypeface (Typeface tf, int style) Sets the typeface and style in which the text should be display Paint if the Typeface that you provided does not have all the bi
void	setTypeface (Typeface tf) Sets the typeface and style in which the text should be display
void	setWidth (int pixels) Makes the TextView exactly this many pixels wide.

Protected Methods

	computeHorizontalScrollRange () Compute the horizontal range that the horizontal scrollbar represents.
int	
	computeVerticalScrollExtent () Compute the vertical extent of the horizontal scrollbar's thumb within th vertical range.
int	

	computeVerticalScrollRange () Compute the vertical range that the vertical scrollbar represents.
int	
	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.
void	
	getBottomPaddingOffset () Amount by which to extend the bottom fading region.
int	
	getDefaultEditable () Subclasses override this to specify that they have a KeyListener by defa even if not specifically called for in the XML options.
boolean	
	getDefaultMovementMethod () Subclasses override this to specify a default movement method.
MovementMethod	
	getLeftFadingEdgeStrength () Returns the strength, or intensity, of the left faded edge.
float	
	getLeftPaddingOffset () Amount by which to extend the left fading region.
int	
	getRightFadingEdgeStrength () Returns the strength, or intensity, of the right faded edge.
float	
	getRightPaddingOffset () Amount by which to extend the right fading region.
int	
	getTopPaddingOffset () Amount by which to extend the top fading region.
int	
	isPaddingOffsetRequired () If the View draws content inside its padding and enables fading edges, it needs to support padding offsets.
boolean	
	onAttachedToWindow () This is called when the view is attached to a window.
void	
	onCreateDrawableState (int extraSpace) Generate the new `Drawable` state for this view.
int[]	
	onDetachedFromWindow () This is called when the view is detached from a window.
void	
	onDraw (Canvas canvas) Implement this to do your drawing.
void	
	onFocusChanged (boolean focused, int direction, Rect previouslyFocusedF Called by the view system when the focus state of this view changes.
void	
	onLayout (boolean changed, int left, int top, int right, int bottom) Called from layout when this view should assign a size and position to each of its children.
void	
	onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and

the measured height.

	onScrollChanged (int horiz, int vert, int oldHoriz, int oldVert)
void	This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents).
	onSelectionChanged (int selStart, int selEnd)
void	This method is called when the selection has changed, in case any subclasses would like to know.
	onTextChanged (CharSequence text, int start, int lengthBefore, int lengthAf
void	This method is called when the text is changed, in case any subclasses would like to know.
	onVisibilityChanged (View changedView, int visibility)
void	Called when the visibility of the view or an ancestor of the view is changed.
	setFrame (int l, int t, int r, int b)
boolean	Assign a size and position to this view.
	verifyDrawable (Drawable who)
boolean	If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying.

Inherited Methods [Expand]

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

XML Attributes

android:autoLink

Controls whether links such as urls and email addresses are automatically found and converted to clickable links. The default value is "none", disabling this feature.

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

Constant	Value	Description
`none`	0x00	Match no patterns (default).
`web`	0x01	Match Web URLs.
`email`	0x02	Match email addresses.

`phone`	0x04	Match phone numbers.
`map`	0x08	Match map addresses.
`all`	0x0f	Match all patterns (equivalent to web|email|phone|map).

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

Related Methods

setAutoLinkMask(int)

android:autoText

If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors. The default is "false".

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

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

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

Related Methods

setKeyListener(KeyListener)

android:bufferType

Determines the minimum type that getText() will return. The default is "normal". Note that EditText and LogTextBox always return Editable, even if you specify something less powerful here.

Must be one of the following constant values.

Constant	Value	Description
`normal`	0	Can return any CharSequence, possibly a Spanned one if the source text was Spanned.
`spannable`	1	Can only return Spannable.
`editable`	2	Can only return Spannable and Editable.

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

Related Methods

setText(CharSequence,TextView.BufferType)

android:capitalize

If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types. The default is "none".

Must be one of the following constant values.

Constant	Value	Description
`none`	0	Don't automatically capitalize anything.
`sentences`	1	Capitalize the first word of each sentence.
`words`	2	Capitalize the first letter of every word.
`characters`	3	Capitalize every character.

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

Related Methods

setKeyListener(KeyListener)

android:cursorVisible

Makes the cursor visible (the default) or invisible.

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

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

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

Related Methods

setCursorVisible(boolean)

android:digits

If set, specifies that this TextView has a numeric input method and that these specific characters are the ones that it will accept. If this is set, numeric is implied to be true. The default is false.

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

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

This corresponds to the global attribute resource symbol `digits`

(/reference/android/R.attr.html#digits).

Related Methods

setKeyListener(KeyListener)

android:drawableBottom

The drawable to be drawn below the text.

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

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

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

Related Methods

setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)

android:drawableEnd

The drawable to be drawn to the end of the text.

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

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

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

Related Methods

setCompoundDrawablesRelativeWithIntrinsicBounds(int,int,int,int)

android:drawableLeft

The drawable to be drawn to the left of the text.

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

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

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

Related Methods

setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)

android:drawablePadding

The padding between the drawables and the text.

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

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

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

Related Methods

setCompoundDrawablePadding(int)

android:drawableRight

The drawable to be drawn to the right of the text.

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

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

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

Related Methods

setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)

android:drawableStart

The drawable to be drawn to the start of the text.

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

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

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

Related Methods

setCompoundDrawablesRelativeWithIntrinsicBounds(int,int,int,int)

android:drawableTop

The drawable to be drawn above the text.

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

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

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

Related Methods

setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)

android:editable

If set, specifies that this TextView has an input method. It will be a textual one unless it has otherwise been specified. For TextView, this is false by default. For EditText, it is true by default.

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

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

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

Related Methods

android:editorExtras

Reference to an <input-extras> (/reference/android/R.styleable.html#InputExtras) XML resource containing additional data to supply to an input method, which is private to the implementation of the input method. This simply fills in the

EditorInfo.extras (/reference/android/view/inputmethod/EditorInfo.html#extras) field when the input method is connected.

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

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

Related Methods
setInputExtras(int)

android:ellipsize

If set, causes words that are longer than the view is wide to be ellipsized instead of broken in the middle. You will often also want to set scrollHorizontally or singleLine as well so that the text as a whole is also constrained to a single line instead of still allowed to be broken onto multiple lines.

Must be one of the following constant values.

Constant	Value	Description
none	0	
start	1	
middle	2	
end	3	
marquee	4	

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

Related Methods
setEllipsize(TextUtils.TruncateAt)

android:ems

Makes the TextView be exactly this many ems wide.

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

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

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

Related Methods

setEms(int)

android:fontFamily

Font family (named by string) for the text.

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

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

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

Related Methods

setTypeface(Typeface)

android:freezesText

If set, the text view will include its current complete text inside of its frozen icicle in addition to meta-data such as the current cursor position. By default this is disabled; it can be useful when the contents of a text view is not stored in a persistent place such as a content provider.

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

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

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

Related Methods

setFreezesText(boolean)

android:gravity

Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view.

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

Constant	Value	Description

top	0x30	Push object to the top of its container, not changing its size.
bottom	0x50	Push object to the bottom of its container, not changing its size.
left	0x03	Push object to the left of its container, not changing its size.
right	0x05	Push object to the right of its container, not changing its size.
center_vertical	0x10	Place object in the vertical center of its container, not changing its size.
fill_vertical	0x70	Grow the vertical size of the object if needed so it completely fills its container.
center_horizontal	0x01	Place object in the horizontal center of its container, not changing its size.
fill_horizontal	0x07	Grow the horizontal size of the object if needed so it completely fills its container.
center	0x11	Place the object in the center of its container in both the vertical and horizontal axis, not changing its size.
fill	0x77	Grow the horizontal and vertical size of the object if needed so it completely fills its container.
clip_vertical	0x80	Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges.
clip_horizontal	0x08	Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The

		clip will be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity will clip the left edge, and neither will clip both edges.
`start`	0x00800003	Push object to the beginning of its container, not changing its size.
`end`	0x00800005	Push object to the end of its container, not changing its size.

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

Related Methods

setGravity(int)

android:height

Makes the TextView be exactly this many pixels tall. You could get the same effect by specifying this number in the layout parameters.

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

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

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

Related Methods

setHeight(int)

android:hint

Hint text to display when the text is empty.

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

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

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

Related Methods
setHint(int)

android:imeActionId

Supply a value for EditorInfo.actionId (/reference/android/view/inputmethod/EditorInfo.html#actionId) used when an input method is connected to the text view.

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

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

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

Related Methods
setImeActionLabel(CharSequence,int)

android:imeActionLabel

Supply a value for EditorInfo.actionLabel (/reference/android/view/inputmethod/EditorInfo.html#actionLabel) used when an input method is connected to the text view.

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

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

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

Related Methods
setImeActionLabel(CharSequence,int)

android:imeOptions

Additional features you can enable in an IME associated with an editor to improve the integration with your application. The constants here correspond to those defined by imeOptions (/reference/android/view/inputmethod/EditorInfo.html#imeOptions).

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

Constant	Value	Description
normal	0x00000000	There are no special semantics associated with t editor.
actionUnspecified	0x00000000	There is no specific action associated with this e let the editor come up with its own if it can. Corresponds to IME_NULL.
actionNone	0x00000001	This editor has no action associated with it. Corresponds to IME_ACTION_NONE.
actionGo	0x00000002	The action key performs a "go" operation to take user to the target of the text they typed. Typically used, for example, when entering a URL. Corresp to IME_ACTION_GO.
actionSearch	0x00000003	The action key performs a "search" operation, tak the user to the results of searching for the text th have typed (in whatever context is appropriate). Corresponds to IME_ACTION_SEARCH.
actionSend	0x00000004	The action key performs a "send" operation, deliv the text to its target. This is typically used when composing a message. Corresponds to IME_ACTION_SEND.
actionNext	0x00000005	The action key performs a "next" operation, takin user to the next field that will accept text. Corresponds to IME_ACTION_NEXT.
actionDone	0x00000006	The action key performs a "done" operation, closi the soft input method. Corresponds to IME_ACTION_DONE.
actionPrevious	0x00000007	The action key performs a "previous" operation, t the user to the previous field that will accept text Corresponds to IME_ACTION_PREVIOUS.
flagNoFullscreen	0x2000000	Used to request that the IME never go into fullscr mode. Applications need to be aware that the fla not a guarantee, and not all IMEs will respect it. Corresponds to IME_FLAG_NO_FULLSCREEN (/reference/android/view/inputmethod /EditorInfo.html#IME_FLAG_NO_FULLSCREEN).
flagNavigatePrevious	0x4000000	Like flagNavigateNext, but specifies there is something interesting that a backward navigatio focus on. If the user selects the IME's facility to backward navigate, this will show up in the application as an actionPrevious at InputConnection.performEditorAction(Corresponds to IME_FLAG_NO_FULLSCREEN

		(/reference/android/view/inputmethod /EditorInfo.html#IME_FLAG_NO_FULLSCREEN).
flagNavigateNext	0x8000000	Used to specify that there is something interestir that a forward navigation can focus on. This is lik using actionNext, except allows the IME to be multiline (with an enter key) as well as provide forward navigation. Note that some IMEs may no able to do this, especially when running on a sma screen where there is little space. In that case it c not need to present a UI for this option. Like actionNext, if the user selects the IME's facility to forward navigate, this will show up in the applica at InputConnection.performEditorAction(. Corresponds to IME_FLAG_NAVIGATE_NEXT (/reference/android/view/inputmethod /EditorInfo.html#IME_FLAG_NAVIGATE_NEXT).
flagNoExtractUi	0x10000000	Used to specify that the IME does not need to sh its extracted text UI. For input methods that may fullscreen, often when in landscape mode, this al them to be smaller and let part of the application shown behind. Though there will likely be limited access to the application available from the user, can make the experience of a (mostly) fullscreen less jarring. Note that when this flag is specified IME may *not* be set up to be able to display text, s should only be used in situations where this is no needed. Corresponds to IME_FLAG_NO_EXTRACT_UI (/reference/android/view/inputmethod /EditorInfo.html#IME_FLAG_NO_EXTRACT_UI).
flagNoAccessoryAction	0x20000000	Used in conjunction with a custom action, this indicates that the action should not be available accessory button when the input method is full-screen. Note that by setting this flag, there ca cases where the action is simply never available the user. Setting this generally means that you th showing text being edited is more important thar action you have supplied. Corresponds to IME_FLAG_NO_ACCESSORY_ACTION (/refere /android/view/inputmethod /EditorInfo.html#IME_FLAG_NO_ACCESSORY_ACTION).

flagNoEnterAction	0x40000000	Used in conjunction with a custom action, this indicates that the action should not be available in-line as a replacement for the "enter" key. Typica this is because the action has such a significant impact or is not recoverable enough that acciden hitting it should be avoided, such as sending a message. Note that TextView will automatically this flag for you on multi-line text views. Corresponds to IME_FLAG_NO_ENTER_ACTIO (/reference/android/view/inputmethod/EditorInfo.html#IME_FLAG_NO_ENTER_ACTION).
flagForceAscii	0x80000000	Used to request that the IME should be capable o inputting ASCII characters. The intention of this f to ensure that the user can type Roman alphabet characters in a TextView used for, typically, acc ID or password input. It is expected that IMEs normally are able to input ASCII even without bei told so (such IMEs already respect this flag in a sense), but there could be some cases they aren' when, for instance, only non-ASCII input languag like Arabic, Greek, Hebrew, Russian are enabled ir IME. Applications need to be aware that the flag a guarantee, and not all IMEs will respect it. How it is strongly recommended for IME authors to re this flag especially when their IME could end up v state that has only non-ASCII input languages enabled. Corresponds to IME_FLAG_FORCE_ASCII (/reference/android/view/inputmethod/EditorInfo.html#IME_FLAG_FORCE_ASCII).

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

Related Methods

setImeOptions(int)

android:includeFontPadding

Leave enough room for ascenders and descenders instead of using the font ascent and descent strictly. (Normally true).

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

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

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

Related Methods

setIncludeFontPadding(boolean)

android:inputMethod

If set, specifies that this TextView should use the specified input method (specified by fully-qualified class name).

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

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

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

Related Methods

setKeyListener(KeyListener)

android:inputType

The type of data being placed in a text field, used to help an input method decide how to let the user enter text. The constants here correspond to those defined by `InputType` (/reference/android/text/InputType.html). Generally you can select a single value, though some can be combined together as indicated. Setting this attribute to anything besides *none* also implies that the text is editable.

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

Constant	Value	Description
none	0x00000000	There is no content type. The text is not editable.
text	0x00000001	Just plain old text. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_NORMAL.
textCapCharacters	0x00001001	Can be combined with *text* and its variations to request capitalization of all characters. Corresponds to TYPE_TEXT_FLAG_CAP_CHARACTERS.

textCapWords	0x00002001	Can be combined with *text* and its variations to request capitalization of the first character of every word. Corresponds to TYPE_TEXT_FLAG_CAP_WORDS.
textCapSentences	0x00004001	Can be combined with *text* and its variations to request capitalization of the first character of every sentence. Corresponds to TYPE_TEXT_FLAG_CAP_SENTENCES.
textAutoCorrect	0x00008001	Can be combined with *text* and its variations to request auto-correction of text being input. Corresponds to TYPE_TEXT_FLAG_AUTO_CORRECT.
textAutoComplete	0x00010001	Can be combined with *text* and its variations to specify that this field will be doing its own auto-completion and talking with the input method appropriately. Corresponds to TYPE_TEXT_FLAG_AUTO_COMPLETE.
textMultiLine	0x00020001	Can be combined with *text* and its variations to allow multiple lines of text in the field. If this flag is not set, the text field will be constrained to a single line. Corresponds to TYPE_TEXT_FLAG_MULTI_LINE.
textImeMultiLine	0x00040001	Can be combined with *text* and its variations to indicate that though the regular text view should not be multiple lines, the IME should provide multiple lines if it can. Corresponds to TYPE_TEXT_FLAG_IME_MULTI_LINE.
textNoSuggestions	0x00080001	Can be combined with *text* and its variations to indicate that the IME should not show any dictionary-based word suggestions. Corresponds to TYPE_TEXT_FLAG_NO_SUGGESTIONS.
textUri	0x00000011	Text that will be used as a URI. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_URI.
textEmailAddress	0x00000021	Text that will be used as an e-mail address. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_EMAIL_ADDRESS.
textEmailSubject	0x00000031	Text that is being supplied as the subject of an e-mail. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_EMAIL_SUBJECT.
textShortMessage	0x00000041	Text that is the content of a short message. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_SHORT_MESSAGE.
textLongMessage	0x00000051	Text that is the content of a long message. Corresponds to TYPE_CLASS_TEXT |

		TYPE_TEXT_VARIATION_LONG_MESSAGE.
textPersonName	0x00000061	Text that is the name of a person. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_PERSON_NAME.
textPostalAddress	0x00000071	Text that is being supplied as a postal mailing address. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_POSTAL_ADDRESS.
textPassword	0x00000081	Text that is a password. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_PASSWORD.
textVisiblePassword	0x00000091	Text that is a password that should be visible. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_VISIBLE_PASSWORD.
textWebEditText	0x000000a1	Text that is being supplied as text in a web form. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_WEB_EDIT_TEXT.
textFilter	0x000000b1	Text that is filtering some other data. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_FILTER.
textPhonetic	0x000000c1	Text that is for phonetic pronunciation, such as a phonetic name field in a contact entry. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_PHONETIC.
textWebEmailAddress	0x000000d1	Text that will be used as an e-mail address on a web form. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_WEB_EMAIL_ADDRESS.
textWebPassword	0x000000e1	Text that will be used as a password on a web form. Corresponds to TYPE_CLASS_TEXT | TYPE_TEXT_VARIATION_WEB_PASSWORD.
number	0x00000002	A numeric only field. Corresponds to TYPE_CLASS_NUMBER | TYPE_NUMBER_VARIATION_NORMAL.
numberSigned	0x00001002	Can be combined with *number* and its other options to allow a signed number. Corresponds to TYPE_CLASS_NUMBER | TYPE_NUMBER_FLAG_SIGNED.
numberDecimal	0x00002002	Can be combined with *number* and its other options to allow a decimal (fractional) number. Corresponds to TYPE_CLASS_NUMBER | TYPE_NUMBER_FLAG_DECIMAL.
numberPassword	0x00000012	A numeric password field. Corresponds to TYPE_CLASS_NUMBER | TYPE_NUMBER_VARIATION_PASSWORD.

phone	0x00000003	For entering a phone number. Corresponds to TYPE_CLASS_PHONE.
datetime	0x00000004	For entering a date and time. Corresponds to TYPE_CLASS_DATETIME | TYPE_DATETIME_VARIATION_NORMAL.
date	0x00000014	For entering a date. Corresponds to TYPE_CLASS_DATETIME | TYPE_DATETIME_VARIATION_DATE.
time	0x00000024	For entering a time. Corresponds to TYPE_CLASS_DATETIME | TYPE_DATETIME_VARIATION_TIME.

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

Related Methods

setRawInputType(int)

android:lineSpacingExtra

Extra spacing between lines of text.

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

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

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

Related Methods

setLineSpacing(float,float)

android:lineSpacingMultiplier

Extra spacing between lines of text, as a multiplier.

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

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

This corresponds to the global attribute resource symbol

lineSpacingMultiplier (/reference/android/R.attr.html#lineSpacingMultiplier).

Related Methods
setLineSpacing(float,float)

android:lines

Makes the TextView be exactly this many lines tall.

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

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

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

Related Methods
setLines(int)

android:linksClickable

If set to false, keeps the movement method from being set to the link movement method even if autoLink causes links to be found.

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

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

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

Related Methods
setLinksClickable(boolean)

android:marqueeRepeatLimit

The number of times to repeat the marquee animation. Only applied if the TextView has marquee enabled.

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

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

May be one of the following constant values.

Constant	Value	Description
marquee_forever	-1	Indicates that marquee should repeat indefinitely.

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

Related Methods

setMarqueeRepeatLimit(int)

android:maxEms

Makes the TextView be at most this many ems wide.

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

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

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

Related Methods

setMaxEms(int)

android:maxHeight

Makes the TextView be at most this many pixels tall.

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

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

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

Related Methods

setMaxHeight(int)

android:maxLength

Set an input filter to constrain the text length to the specified number.

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

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

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

Related Methods
setFilters(InputFilter)

android:maxLines

Makes the TextView be at most this many lines tall. When used on an editable text, the `inputType` attribute's value must be combined with the `textMultiLine` flag for the maxLines attribute to apply.

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

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

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

Related Methods
setMaxLines(int)

android:maxWidth

Makes the TextView be at most this many pixels wide.

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

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

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

Related Methods
setMaxWidth(int)

android:minEms

Makes the TextView be at least this many ems wide.

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

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

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

Related Methods
setMinEms(int)

android:minHeight

Makes the TextView be at least this many pixels tall.

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

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

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

Related Methods
setMinHeight(int)

android:minLines

Makes the TextView be at least this many lines tall. When used on an editable text, the `inputType` attribute's value must be combined with the `textMultiLine` flag for the minLines attribute to apply.

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

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

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

Related Methods
setMinLines(int)

android:minWidth

Makes the TextView be at least this many pixels wide.

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

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

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

Related Methods
setMinWidth(int)

android:numeric

If set, specifies that this TextView has a numeric input method. The default is false.

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

Constant	Value	Description
`integer`	`0x01`	Input is numeric.
`signed`	`0x03`	Input is numeric, with sign allowed.
`decimal`	`0x05`	Input is numeric, with decimals allowed.

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

Related Methods
setKeyListener(KeyListener)

android:password

Whether the characters of the field are displayed as password dots instead of themselves.

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

This may also be a reference to a resource (in the form

"@[*package*:]*type*:*name*") or theme attribute (in the form "?[*package*:][*type*:]*name*") containing a value of this type.

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

Related Methods

setTransformationMethod(TransformationMethod)

android:phoneNumber

If set, specifies that this TextView has a phone number input method. The default is false.

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

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

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

Related Methods

setKeyListener(KeyListener)

android:privateImeOptions

An addition content type description to supply to the input method attached to the text view, which is private to the implementation of the input method. This simply fills in the EditorInfo.privateImeOptions (/reference/android/view/inputmethod/EditorInfo.html#privateImeOptions) field when the input method is connected.

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

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

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

Related Methods

setPrivateImeOptions(String)

android:scrollHorizontally

Whether the text is allowed to be wider than the view (and therefore can be scrolled horizontally).

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

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

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

Related Methods

setHorizontallyScrolling(boolean)

android:selectAllOnFocus

If the text is selectable, select it all when the view takes focus.

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

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

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

Related Methods

setSelectAllOnFocus(boolean)

android:shadowColor

Place a shadow of the specified color behind the text.

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

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

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

Related Methods

setShadowLayer(float,float,float,int)

android:shadowDx

Horizontal offset of the shadow.

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

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

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

Related Methods

setShadowLayer(float,float,float,int)

android:shadowDy

Vertical offset of the shadow.

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

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

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

Related Methods

setShadowLayer(float,float,float,int)

android:shadowRadius

Radius of the shadow.

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

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

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

Related Methods

setShadowLayer(float,float,float,int)

android:singleLine

Constrains the text to a single horizontally scrolling line instead of letting it wrap onto multiple lines, and advances focus instead of inserting a newline when you press the enter key. The default value is false (multi-line wrapped text mode) for non-editable text, but if you specify any value for inputType, the default is true (single-line input field mode).

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

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

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

Related Methods

setTransformationMethod(TransformationMethod)

android:text

Text to display.

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

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

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

Related Methods

setText(CharSequence,TextView.BufferType)

android:textAllCaps

Present the text in ALL CAPS. This may use a small-caps form when available.

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

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

This corresponds to the global attribute resource symbol

textAllCaps (/reference/android/R.attr.html#textAllCaps).

Related Methods
setAllCaps(boolean)

android:textAppearance

Base text color, typeface, size, and style.

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

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

Related Methods

android:textColor

Text color.

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

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

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

Related Methods
setTextColor(int)

android:textColorHighlight

Color of the text selection highlight.

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

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

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

Related Methods
setHighlightColor(int)

android:textColorHint

Color of the hint text.

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

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

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

Related Methods
setHintTextColor(int)

android:textColorLink

Text color for links.

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

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

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

Related Methods
setLinkTextColor(int)

android:textIsSelectable

Indicates that the content of a non-editable text can be selected.

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

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

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

Related Methods

isTextSelectable()

android:textScaleX

Sets the horizontal scaling factor for the text.

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

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

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

Related Methods

setTextScaleX(float)

android:textSize

Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).

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

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

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

Related Methods

setTextSize(int,float)

android:textStyle

Style (bold, italic, bolditalic) for the text.

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

Constant	Value	Description
normal	0	
bold	1	

`italic` `2`

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

Related Methods
setTypeface(Typeface)

android:typeface

Typeface (normal, sans, serif, monospace) for the text.

Must be one of the following constant values.

Constant	Value	Description
`normal`	0	
`sans`	1	
`serif`	2	
`monospace`	3	

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

Related Methods
setTypeface(Typeface)

android:width

Makes the TextView be exactly this many pixels wide. You could get the same effect by specifying this number in the layout parameters.

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

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

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

Related Methods
setWidth(int)

Public Constructors

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

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

public **TextView** (Context context, AttributeSet attrs, int defStyle) Added in API level 1

Public Methods

public void **addTextChangedListener** (TextWatcher watcher) Added in API level 1

Adds a TextWatcher to the list of those whose methods are called whenever this TextView's text changes.

In 1.0, the `afterTextChanged(Editable)` (/reference/android/text/TextWatcher.html#afterTextChanged(android.text.Editable)) method was erroneously not called after `setText(char[], int, int)` (/reference/android/widget/TextView.html#setText(char[], int, int)) calls. Now, doing `setText(char[], int, int)` (/reference/android/widget/TextView.html#setText(char[], int, int)) if there are any text changed listeners forces the buffer type to Editable if it would not otherwise be and does call this method.

public final void **append** (CharSequence text) Added in API level 1

Convenience method: Append the specified text to the TextView's display buffer, upgrading it to BufferType.EDITABLE if it was not already editable.

public void **append** (CharSequence text, int start, int end) Added in API level 1

Convenience method: Append the specified text slice to the TextView's display buffer, upgrading it to BufferType.EDITABLE if it was not already editable.

public void **beginBatchEdit** () Added in API level 3

public boolean **bringPointIntoView** (int offset) Added in API level 3

Move the point, specified by the offset, into the view if it is needed. This has to be called after layout. Returns true if anything changed.

public void **cancelLongPress** ()

Added in API level 1

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

public void **clearComposingText** ()

Added in API level 3

Use `BaseInputConnection.removeComposingSpans()` (/reference/android/view/inputmethod/BaseInputConnection.html#removeComposingSpans(android.text.Spannable)) to remove any IME composing state from this text view.

public void **computeScroll** ()

Added in API level 1

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

public void **debug** (int depth)

Added in API level 1

Prints information about this view in the log output, with the tag `VIEW_LOG_TAG` (/reference/android/view/View.html#VIEW_LOG_TAG). Each line in the output is preceded with an indentation defined by the `depth`.

Parameters

depth the indentation level

public boolean **didTouchFocusSelect** ()

Added in API level 3

Returns true, only while processing a touch gesture, if the initial touch down event caused focus to move to the text view and as a result its selection changed. Only valid while processing the touch gesture of interest, in an editable text view.

public void **endBatchEdit** ()

Added in API level 3

public boolean **extractText** (ExtractedTextRequest request, ExtractedText outText)

Added in API level 3

If this TextView contains editable content, extract a portion of it based on the information in *request* in to *outText*.

Returns

Returns true if the text was successfully extracted, else false.

public void **findViewsWithText** (ArrayList<View> outViews, CharSequence searched, int flags)

Added in API level 14

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

Parameters

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

public final int **getAutoLinkMask** ()

Added in API level 1

Gets the autolink mask of the text. See `Linkify.ALL (/reference/android/text/util/Linkify.html#ALL)` and peers for possible values.

Related XML Attributes

android:autoLink

public int **getBaseline** ()

Added in API level 1

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

Returns

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

public int **getCompoundDrawablePadding** ()

Added in API level 1

Returns the padding between the compound drawables and the text.

Related XML Attributes

android:drawablePadding

public Drawable[] **getCompoundDrawables** ()

Added in API level 1

Returns drawables for the left, top, right, and bottom borders.

Related XML Attributes

android:drawableLeft
android:drawableTop
android:drawableRight
android:drawableBottom

public Drawable[] **getCompoundDrawablesRelative** () Added in API level 17

Returns drawables for the start, top, end, and bottom borders.

Related XML Attributes

android:drawableStart
android:drawableTop
android:drawableEnd
android:drawableBottom

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

Returns the bottom padding of the view, plus space for the bottom Drawable if any.

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

Returns the end padding of the view, plus space for the end Drawable if any.

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

Returns the left padding of the view, plus space for the left Drawable if any.

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

Returns the right padding of the view, plus space for the right Drawable if any.

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

Returns the start padding of the view, plus space for the start Drawable if any.

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

Returns the top padding of the view, plus space for the top Drawable if any.

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

Return the current color selected to paint the hint text.

Returns

Returns the current hint text color.

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

Return the current color selected for normal text.

Returns

Returns the current text color.

public ActionMode.Callback **getCustomSelectionActionModeCallback** () Added in API level 11

Retrieves the value set in `setCustomSelectionActionModeCallback(ActionMode.Callback)` (/reference/android/widget/TextView.html#setCustomSelectionActionModeCallback(android.view.ActionMode.Callback)). Default is null.

Returns

The current custom selection callback.

public Editable **getEditableText** () Added in API level 3

Return the text the TextView is displaying as an Editable object. If the text is not editable, null is returned.

See Also

`getText()`

public TextUtils.TruncateAt **getEllipsize** () Added in API level 1

Returns where, if anywhere, words that are longer than the view is wide should be ellipsized.

public CharSequence **getError** () Added in API level 1

Returns the error message that was set to be displayed with `setError(CharSequence)` (/reference/android/widget/TextView.html#setError(java.lang.CharSequence)), or `null` if no error was set or if it the error was cleared by the widget after user input.

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

Returns the extended bottom padding of the view, including both the

bottom Drawable if any and any extra space to keep more than maxLines of text from showing. It is only valid to call this after measuring.

public int **getExtendedPaddingTop** ()

Added in API level 1

Returns the extended top padding of the view, including both the top Drawable if any and any extra space to keep more than maxLines of text from showing. It is only valid to call this after measuring.

public InputFilter[] **getFilters** ()

Added in API level 1

Returns the current list of input filters.

Related XML Attributes

android:maxLength

public void **getFocusedRect** (Rect r)

Added in API level 1

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

Parameters

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

public boolean **getFreezesText** ()

Added in API level 1

Return whether this text view is including its entire text contents in frozen icicles.

Returns

Returns true if text is included, false if it isn't.

See Also

`setFreezesText(boolean)`

public int **getGravity** ()

Added in API level 1

Returns the horizontal and vertical alignment of this TextView.

Related XML Attributes

android:gravity

See Also
Gravity

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

Related XML Attributes
android:textColorHighlight

Returns
the color used to display the selection highlight

See Also
setHighlightColor(int)

public CharSequence **getHint** () Added in API level 1

Returns the hint that is displayed when the text of the TextView is empty.

Related XML Attributes
android:hint

public final ColorStateList **getHintTextColors** () Added in API level 1

Related XML Attributes
android:textColorHint

Returns
the color of the hint text, for the different states of this TextView.

See Also
setHintTextColor(ColorStateList)
setHintTextColor(int)
setTextColor(ColorStateList)
setLinkTextColor(ColorStateList)

public int **getImeActionId** () Added in API level 3

Get the IME action ID previous set with setImeActionLabel(CharSequence, int) (/reference/android/widget/TextView.html#setImeActionLabel(java.lang.CharSequence, int)).

See Also
setImeActionLabel(CharSequence, int)
EditorInfo

public CharSequence **getImeActionLabel** () Added in API level 3

Get the IME action label previous set with `setImeActionLabel(CharSequence, int)` (/reference/android/widget/TextView.html#setImeActionLabel(java.lang.CharSequence, int)).

See Also

`setImeActionLabel(CharSequence, int)`
`EditorInfo`

public int **getImeOptions** ()

Added in API level 3

Get the type of the IME editor.

See Also

`setImeOptions(int)`
`EditorInfo`

public boolean **getIncludeFontPadding** ()

Added in API level 16

Gets whether the TextView includes extra top and bottom padding to make room for accents that go above the normal ascent and descent.

Related XML Attributes

android:includeFontPadding

See Also

`setIncludeFontPadding(boolean)`

public Bundle **getInputExtras** (boolean create)

Added in API level 3

Retrieve the input extras currently associated with the text view, which can be viewed as well as modified.

Related XML Attributes

android:editorExtras

Parameters

create	If true, the extras will be created if they don't already exist. Otherwise, null will be returned if none have been created.

See Also

`setInputExtras(int)`
`extras`

public int **getInputType** ()

Added in API level 3

Get the type of the editable content.

See Also

setInputType(int)
InputType

public final KeyListener **getKeyListener** () Added in API level 1

Related XML Attributes

android:numeric
android:digits
android:phoneNumber
android:inputMethod
android:capitalize
android:autoText

Returns

the current key listener for this TextView. This will frequently be null for non-EditText TextViews.

public final Layout **getLayout** () Added in API level 1

Returns

the Layout that is currently being used to display the text. This can be null if the text or width has recently changes.

public int **getLineBounds** (int line, Rect bounds) Added in API level 1

Return the baseline for the specified line (0...getLineCount() - 1) If bounds is not null, return the top, left, right, bottom extents of the specified line in it. If the internal Layout has not been built, return 0 and set bounds to (0, 0, 0, 0)

Parameters

line	which line to examine (0..getLineCount() - 1)
bounds	Optional. If not null, it returns the extent of the line

Returns

the Y-coordinate of the baseline

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

Return the number of lines of text, or 0 if the internal Layout has not been built.

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

Returns

the height of one standard line in pixels. Note that markup within the text can cause individual lines to be taller or shorter than this

height, and the layout may contain additional first- or last-line padding.

public float **getLineSpacingExtra** ()

Added in API level 16

Gets the line spacing extra space

Related XML Attributes

android:lineSpacingExtra

Returns

the extra space that is added to the height of each lines of this TextView.

See Also

`setLineSpacing(float, float)`
`getLineSpacingMultiplier()`

public float **getLineSpacingMultiplier** ()

Added in API level 16

Gets the line spacing multiplier

Related XML Attributes

android:lineSpacingMultiplier

Returns

the value by which each line's height is multiplied to get its actual height.

See Also

`setLineSpacing(float, float)`
`getLineSpacingExtra()`

public final ColorStateList **getLinkTextColors** ()

Added in API level 1

Related XML Attributes

android:textColorLink

Returns

the list of colors used to paint the links in the text, for the different states of this TextView

See Also

`setLinkTextColor(ColorStateList)`
`setLinkTextColor(int)`

public final boolean **getLinksClickable** ()

Added in API level 1

Returns whether the movement method will automatically be set to `LinkMovementMethod (/reference/android/text/method`

/LinkMovementMethod.html) if setAutoLinkMask(int) (/reference/android/widget/TextView.html#setAutoLinkMask(int)) has been set to nonzero and links are detected in setText(char[], int, int) (/reference/android/widget/TextView.html#setText(char[], int, int)). The default is true.

Related XML Attributes

android:linksClickable

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

Gets the number of times the marquee animation is repeated. Only meaningful if the TextView has marquee enabled.

Related XML Attributes

android:marqueeRepeatLimit

Returns

the number of times the marquee animation is repeated. -1 if the animation repeats indefinitely

See Also

setMarqueeRepeatLimit(int)

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

Related XML Attributes

android:maxEms

Returns

the maximum width of the TextView, expressed in ems or -1 if the maximum width was set in pixels instead (using setMaxWidth(int) or setWidth(int)).

See Also

setMaxEms(int)
setEms(int)

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

Related XML Attributes

android:maxHeight

Returns

the maximum height of this TextView expressed in pixels, or -1 if the maximum height was set in number of lines instead using or #setLines(int).

See Also

setMaxHeight(int)

public int **getMaxLines** ()

Added in API level 16

Related XML Attributes

android:maxLines

Returns

the maximum number of lines displayed in this TextView, or -1 if the maximum height was set in pixels instead using or #setHeight(int).

See Also

setMaxLines(int)

public int **getMaxWidth** ()

Added in API level 16

Related XML Attributes

android:maxWidth

Returns

the maximum width of the TextView, in pixels or -1 if the maximum width was set in ems instead (using setMaxEms(int) or setEms(int)).

See Also

setMaxWidth(int)
setWidth(int)

public int **getMinEms** ()

Added in API level 16

Related XML Attributes

android:minEms

Returns

the minimum width of the TextView, expressed in ems or -1 if the minimum width was set in pixels instead (using setMinWidth(int) or setWidth(int)).

See Also

setMinEms(int)
setEms(int)

public int **getMinHeight** ()

Added in API level 16

Related XML Attributes

android:minHeight

Returns

the minimum height of this TextView expressed in pixels, or -1 if the minimum height was set in number of lines instead using or #setLines(int).

See Also

setMinHeight(int)

public int **getMinLines** ()

Added in API level 16

Related XML Attributes

android:minLines

Returns

the minimum number of lines displayed in this TextView, or -1 if the minimum height was set in pixels instead using or #setHeight(int).

See Also

setMinLines(int)

public int **getMinWidth** ()

Added in API level 16

Related XML Attributes

android:minWidth

Returns

the minimum width of the TextView, in pixels or -1 if the minimum width was set in ems instead (using setMinEms(int) or setEms(int)).

See Also

setMinWidth(int)
setWidth(int)

public final MovementMethod **getMovementMethod** ()

Added in API level 1

Returns

the movement method being used for this TextView. This will frequently be null for non-EditText TextViews.

public int **getOffsetForPosition** (float x, float y)

Added in API level 14

Get the character offset closest to the specified absolute position. A typical use case is to pass the result of getX() (/reference/android/view/MotionEvent.html#getX()) and getY() (/reference/android/view/MotionEvent.html#getY()) to this method.

Parameters

x The horizontal absolute position of a point on screen

y The vertical absolute position of a point on screen

Returns

the character offset for the character whose position is closest to the specified position. Returns -1 if there is no layout.

public TextPaint **getPaint** () Added in API level 1

Returns

the base paint used for the text. Please use this only to consult the Paint's properties and not to change them.

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

Returns

the flags on the Paint being used to display the text.

See Also

`getFlags()`

public String **getPrivateImeOptions** () Added in API level 3

Get the private type of the content.

See Also

`setPrivateImeOptions(String)`
`privateImeOptions`

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

Convenience for `getSelectionEnd(CharSequence) (/reference/android/text/Selection.html#getSelectionEnd(java.lang.CharSequence))`.

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

Convenience for `getSelectionStart(CharSequence) (/reference/android/text/Selection.html#getSelectionStart(java.lang.CharSequence))`.

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

Related XML Attributes

android:shadowColor

Returns

the color of the shadow layer

See Also

setShadowLayer(float, float, float, int)

public float **getShadowDx** ()

Added in API level 16

Related XML Attributes

android:shadowDx

Returns

the horizontal offset of the shadow layer

See Also

setShadowLayer(float, float, float, int)

public float **getShadowDy** ()

Added in API level 16

Related XML Attributes

android:shadowDy

Returns

the vertical offset of the shadow layer

See Also

setShadowLayer(float, float, float, int)

public float **getShadowRadius** ()

Added in API level 16

Gets the radius of the shadow layer.

Related XML Attributes

android:shadowRadius

Returns

the radius of the shadow layer. If 0, the shadow layer is not visible

See Also

setShadowLayer(float, float, float, int)

public CharSequence **getText** ()

Added in API level 1

Return the text the TextView is displaying. If setText() was called with an argument of BufferType.SPANNABLE or BufferType.EDITABLE, you can cast the return value from this method to Spannable or Editable, respectively. Note: The content of the return value should not be modified. If you want a modifiable one, you should make your own copy first.

Related XML Attributes

android:text

public static int **getTextColor** (Context context,

TypedArray attrs, int def) Added in API level 1

Returns the default color from the TextView_textColor attribute from the AttributeSet, if set, or the default color from the TextAppearance_textColor from the TextView_textAppearance attribute, if TextView_textColor was not set directly.

public final ColorStateList **getTextColors** () Added in API level 1

Gets the text colors for the different states (normal, selected, focused) of the TextView.

Related XML Attributes

android:textColor

See Also

`setTextColor(ColorStateList)`
`setTextColor(int)`

public static ColorStateList **getTextColors** (Context context, TypedArray attrs) Added in API level 1

Returns the TextView_textColor attribute from the TypedArray, if set, or the TextAppearance_textColor from the TextView_textAppearance attribute, if TextView_textColor was not set directly.

public Locale **getTextLocale** () Added in API level 17

Get the default `Locale (/reference/java/util/Locale.html)` of the text in this TextView.

Returns

the default `Locale` of the text in this TextView.

public float **getTextScaleX** () Added in API level 1

Returns

the extent by which text is currently being stretched horizontally. This will usually be 1.

public float **getTextSize** () Added in API level 1

Returns

the size (in pixels) of the default text size in this TextView.

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

Returns the total bottom padding of the view, including the bottom

Drawable if any, the extra space to keep more than maxLines from showing, and the vertical offset for gravity, if any.

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

Returns the total end padding of the view, including the end Drawable if any.

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

Returns the total left padding of the view, including the left Drawable if any.

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

Returns the total right padding of the view, including the right Drawable if any.

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

Returns the total start padding of the view, including the start Drawable if any.

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

Returns the total top padding of the view, including the top Drawable if any, the extra space to keep more than maxLines from showing, and the vertical offset for gravity, if any.

public final TransformationMethod **getTransformationMethod** () Added in API level 1

Related XML Attributes

android:password
android:singleLine

Returns

the current transformation method for this TextView. This will frequently be null except for single-line and password fields.

public Typeface **getTypeface** () Added in API level 1

Related XML Attributes

android:fontFamily
android:typeface
android:textStyle

Returns

the current typeface and style in which the text is being displayed.

See Also
setTypeface(Typeface)

public URLSpan[] **getUrls** () Added in API level 1

Returns the list of URLSpans attached to the text (by Linkify (/reference/android/text/util/Linkify.html) or otherwise) if any. You can call getURL() (/reference/android/text/style/URLSpan.html#getURL()) on them to find where they link to or use getSpanStart(Object) (/reference/android/text/Spanned.html#getSpanStart(java.lang.Object)) and getSpanEnd(Object) (/reference/android/text/Spanned.html#getSpanEnd(java.lang.Object)) to find the region of the text they are attached to.

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

Returns whether this View has content which overlaps.

This function, intended to be overridden by specific View types, is an optimization when alpha is set on a view. If rendering overlaps in a view with alpha < 1, that view is drawn to an offscreen buffer and then composited into place, which can be expensive. If the view has no overlapping rendering, the view can draw each primitive with the appropriate alpha value directly. An example of overlapping rendering is a TextView with a background image, such as a Button. An example of non-overlapping rendering is a TextView with no background, or an ImageView with only the foreground image. The default implementation returns true; subclasses should override if they have cases which can be optimized.

The current implementation of the saveLayer and saveLayerAlpha methods in Canvas (/reference/android/graphics/Canvas.html) necessitates that a View return true if it uses the methods internally without passing the CLIP_TO_LAYER_SAVE_FLAG (/reference/android/graphics/Canvas.html#CLIP_TO_LAYER_SAVE_FLAG).

Returns
true if the content in this view might overlap, false otherwise.

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

Return true iff there is a selection inside this text view.

public void **invalidateDrawable** (Drawable drawable) Added in API level 1

Invalidates the specified Drawable.

Parameters

drawable the drawable to invalidate

public boolean **isCursorVisible** ()

Added in API level 16

Related XML Attributes

android:cursorVisible

Returns

whether or not the cursor is visible (assuming this TextView is editable)

See Also

`setCursorVisible(boolean)`

public boolean **isInputMethodTarget** ()

Added in API level 3

Returns whether this text view is a current input method target. The default implementation just checks with `InputMethodManager` (/reference/android/view/inputmethod/InputMethodManager.html).

public boolean **isSuggestionsEnabled** ()

Added in API level 14

Return whether or not suggestions are enabled on this TextView. The suggestions are generated by the IME or by the spell checker as the user types. This is done by adding `SuggestionSpan` (/reference/android/text/style/SuggestionSpan.html)s to the text. When suggestions are enabled (default), this list of suggestions will be displayed when the user asks for them on these parts of the text. This value depends on the inputType of this TextView. The class of the input type must be `TYPE_CLASS_TEXT` (/reference/android/text/InputType.html#TYPE_CLASS_TEXT). In addition, the type variation must be one of `TYPE_TEXT_VARIATION_NORMAL` (/reference/android/text/InputType.html#TYPE_TEXT_VARIATION_NORMAL), `TYPE_TEXT_VARIATION_EMAIL_SUBJECT` (/reference/android/text/InputType.html#TYPE_TEXT_VARIATION_EMAIL_SUBJECT), `TYPE_TEXT_VARIATION_LONG_MESSAGE` (/reference/android/text/InputType.html#TYPE_TEXT_VARIATION_LONG_MESSAGE), `TYPE_TEXT_VARIATION_SHORT_MESSAGE` (/reference/android/text/InputType.html#TYPE_TEXT_VARIATION_SHORT_MESSAGE) or `TYPE_TEXT_VARIATION_WEB_EDIT_TEXT` (/reference/android/text/InputType.html#TYPE_TEXT_VARIATION_WEB_EDIT_TEXT). And finally, the `TYPE_TEXT_FLAG_NO_SUGGESTIONS` (/reference/android/text/InputType.html#TYPE_TEXT_FLAG_NO_SUGGESTIONS) flag must *not* be

set.

Returns

true if the suggestions popup window is enabled, based on the inputType.

public boolean **isTextSelectable** ()

Added in API level 11

Returns the state of the `textIsSelectable` flag (See `setTextIsSelectable()` (/reference/android/widget/TextView.html#setTextIsSelectable(boolean))). Although you have to set this flag to allow users to select and copy text in a non-editable TextView, the content of an `EditText` (/reference/android/widget/EditText.html) can always be selected, independently of the value of this flag.

Related XML Attributes

android:textIsSelectable

Returns

True if the text displayed in this TextView can be selected by the user.

public void **jumpDrawablesToCurrentState** ()

Added in API level 11

Call `Drawable.jumpToCurrentState()` (/reference/android/graphics/drawable/Drawable.html#jumpToCurrentState()) on all Drawable objects associated with this view.

public int **length** ()

Added in API level 1

Returns the length, in characters, of the text managed by this TextView

public boolean **moveCursorToVisibleOffset** ()

Added in API level 3

Move the cursor, if needed, so that it is at an offset that is visible to the user. This will not move the cursor if it represents more than one character (a selection range). This will only work if the TextView contains spannable text; otherwise it will do nothing.

Returns

True if the cursor was actually moved, false otherwise.

public void **onBeginBatchEdit** ()

Added in API level 3

Called by the framework in response to a request to begin a batch of edit operations through a call to link `beginBatchEdit()` (/reference

/android/widget/TextView.html#beginBatchEdit()).

public boolean **onCheckIsTextEditor** ()

Added in API level 3

Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. Subclasses should override this if they implement onCreateInputConnection(EditorInfo) (/reference/android/view/View.html#onCreateInputConnection(android.view.inputmethod.EditorInfo)) to return true if a call on that method would return a non-null InputConnection, and they are really a first-class editor that the user would normally start typing on when the go into a window containing your view.

The default implementation always returns false. This does *not* mean that its onCreateInputConnection(EditorInfo) (/reference/android/view/View.html#onCreateInputConnection(android.view.inputmethod.EditorInfo)) will not be called or the user can not otherwise perform edits on your view; it is just a hint to the system that this is not the primary purpose of this view.

Returns

Returns true if this view is a text editor, else false.

public void **onCommitCompletion** (CompletionInfo text)

Added in API level 3

Called by the framework in response to a text completion from the current input method, provided by it calling InputConnection.commitCompletion() (/reference/android/view/inputmethod/InputConnection.html#commitCompletion(android.view.inputmethod.CompletionInfo)). The default implementation does nothing; text views that are supporting auto-completion should override this to do their desired behavior.

Parameters

text The auto complete text the user has selected.

public void **onCommitCorrection** (CorrectionInfo info)

Added in API level 11

Called by the framework in response to a text auto-correction (such as fixing a typo using a a dictionnary) from the current input method, provided by it calling commitCorrection(CorrectionInfo) (/reference/android/view/inputmethod

/InputConnection.html#commitCorrection(android.view.inputmethod.CorrectionInfo)) InputConnection.commitCorrection()}. The default implementation flashes the background of the corrected word to provide feedback to the user.

Parameters

info The auto correct info about the text that was corrected.

public InputConnection **onCreateInputConnection** (EditorInfo outAttrs)

Added in API level 3

Create a new InputConnection for an InputMethod to interact with the view. The default implementation returns null, since it doesn't support input methods. You can override this to implement such support. This is only needed for views that take focus and text input.

When implementing this, you probably also want to implement `onCheckIsTextEditor()` (/reference/android/view/View.html#onCheckIsTextEditor()) to indicate you will return a non-null InputConnection.

Parameters

outAttrs Fill in with attribute information about the connection.

public boolean **onDragEvent** (DragEvent event)

Added in API level 11

Handles drag events sent by the system following a call to `startDrag()` (/reference/android/view/View.html#startDrag(android.content.ClipData, android.view.View.DragShadowBuilder, java.lang.Object, int)).

When the system calls this method, it passes a `DragEvent` (/reference/android/view/DragEvent.html) object. A call to `getAction()` (/reference/android/view/DragEvent.html#getAction()) returns one of the action type constants defined in DragEvent. The method uses these to determine what is happening in the drag and drop operation.

Parameters

event The `DragEvent` sent by the system. The `getAction()` method returns an action type constant defined in DragEvent, indicating the type of drag event represented by this object.

Returns

`true` if the method was successful, otherwise `false`.

The method should return `true` in response to an action type of

ACTION_DRAG_STARTED (/reference/android/view/DragEvent.html#ACTION_DRAG_STARTED) to receive drag events for the current operation.

The method should also return `true` in response to an action type of ACTION_DROP (/reference/android/view/DragEvent.html#ACTION_DROP) if it consumed the drop, or `false` if it didn't.

public void **onEditorAction** (int actionCode)

Added in API level 3

Called when an attached input method calls InputConnection.performEditorAction() (/reference/android/view/inputmethod/InputConnection.html#performEditorAction(int)) for this text view. The default implementation will call your action listener supplied to setOnEditorActionListener(TextView.OnEditorActionListener) (/reference/android/widget/TextView.html#setOnEditorActionListener(android.widget.TextView.OnEditorActionListener)), or perform a standard operation for EditorInfo.IME_ACTION_NEXT (/reference/android/view/inputmethod/EditorInfo.html#IME_ACTION_NEXT), EditorInfo.IME_ACTION_PREVIOUS (/reference/android/view/inputmethod/EditorInfo.html#IME_ACTION_PREVIOUS), or EditorInfo.IME_ACTION_DONE (/reference/android/view/inputmethod/EditorInfo.html#IME_ACTION_DONE).

For backwards compatibility, if no IME options have been set and the text view would not normally advance focus on enter, then the NEXT and DONE actions received here will be turned into an enter key down/up pair to go through the normal key handling.

Parameters

actionCode The code of the action being performed.

See Also

setOnEditorActionListener(TextView.OnEditorActionListener)

public void **onEndBatchEdit** ()

Added in API level 3

Called by the framework in response to a request to end a batch of edit operations through a call to link endBatchEdit() (/reference/android/widget/TextView.html#endBatchEdit()).

public void **onFinishTemporaryDetach** ()

Added in API level 3

Called after `onStartTemporaryDetach()` (/reference/android/view/View.html#onStartTemporaryDetach()) when the container is done changing the view.

public boolean **onGenericMotionEvent** (MotionEvent event)

Added in API level 12

Implement this method to handle generic motion events.

Generic motion events describe joystick movements, mouse hovers, track pad touches, scroll wheel movements and other input events. The `source` (/reference/android/view/MotionEvent.html#getSource()) of the motion event specifies the class of input that was received. Implementations of this method must examine the bits in the source before processing the event. The following code example shows how this is done.

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

```
public boolean onGenericMotionEvent(MotionEvent eve
    if (event.isFromSource(InputDevice.SOURCE_CLASS
        if (event.getAction() == MotionEvent.ACTION
            // process the joystick movement...
            return true;
        }
    }
    if (event.isFromSource(InputDevice.SOURCE_CLASS
        switch (event.getAction()) {
            case MotionEvent.ACTION_HOVER_MOVE:
                // process the mouse hover movement
                return true;
            case MotionEvent.ACTION_SCROLL:
                // process the scroll wheel movemen
                return true;
        }
    }
    return super.onGenericMotionEvent(event);
}
```

Parameters

event The generic motion event being processed.

Returns

True if the event was handled, false otherwise.

public void **onInitializeAccessibilityEvent** (AccessibilityEvent event)

Added in API level 14

Initializes an `AccessibilityEvent (/reference/android/view/accessibility/AccessibilityEvent.html)` with information about this View which is the event source. In other words, the source of an accessibility event is the view whose state change triggered firing the event.

Example: Setting the password property of an event in addition to properties set by the super implementation:

```
public void onInitializeAccessibilityEvent(Accessib
    super.onInitializeAccessibilityEvent(event);
    event.setPassword(true);
}
```

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

> **Note:** Always call the super implementation before adding information to the event, in case the default implementation has basic information to add.

Parameters

event The event to initialize.

public void **onInitializeAccessibilityNodeInfo** (AccessibilityNodeInfo info)

Added in API level 14

Initializes an `AccessibilityNodeInfo (/reference/android/view/accessibility/AccessibilityNodeInfo.html)` with information about this view. The base implementation sets:

- `setParent(View)`,
- `setBoundsInParent(Rect)`,
- `setBoundsInScreen(Rect)`,
- `setPackageName(CharSequence)`,

- setClassName(CharSequence),
- setContentDescription(CharSequence),
- setEnabled(boolean),
- setClickable(boolean),
- setFocusable(boolean),
- setFocused(boolean),
- setLongClickable(boolean),
- setSelected(boolean),

Subclasses should override this method, call the super implementation, and set additional attributes.

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

Parameters

info The instance to initialize.

public boolean **onKeyDown** (int keyCode, KeyEvent event)

Added in API level 1

Default implementation of KeyEvent.Callback.onKeyDown() (/reference/android/view/KeyEvent.Callback.html#onKeyDown(int, android.view.KeyEvent)): perform press of the view when KEYCODE_DPAD_CENTER (/reference/android/view/KeyEvent.html#KEYCODE_DPAD_CENTER) or KEYCODE_ENTER (/reference/android/view/KeyEvent.html#KEYCODE_ENTER) is released, if the view is enabled and clickable.

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

Parameters

keyCode A key code that represents the button pressed, from KeyEvent.

event The KeyEvent object that defines the button action.

Returns

If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyMultiple** (int keyCode, int repeatCount, KeyEvent event)

Added in API level 1

Default implementation of `KeyEvent.Callback.onKeyMultiple()` (/reference/android/view/KeyEvent.Callback.html#onKeyMultiple(int, int, android.view.KeyEvent)): always returns false (doesn't handle the event).

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

Parameters

keyCode	A key code that represents the button pressed, from `KeyEvent`.
repeatCount	The number of times the action was made.
event	The KeyEvent object that defines the button action.

Returns

If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyPreIme** (int keyCode, KeyEvent event)

Added in API level 3

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

Parameters

keyCode	The value in event.getKeyCode().
event	Description of the key event.

Returns

If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyShortcut** (int keyCode, KeyEvent event)

Called on the focused view when a key shortcut event is not handled. Override this method to implement local key shortcuts for the View. Key shortcuts can also be implemented by setting the `shortcut` (/reference/android/view/MenuItem.html#setShortcut(char, char)) property of menu items.

Parameters

keyCode	The value in event.getKeyCode().
event	Description of the key event.

Returns

If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public boolean **onKeyUp** (int keyCode, KeyEvent event)

Added in API level 1

Default implementation of `KeyEvent.Callback.onKeyUp()` (/reference/android/view/KeyEvent.Callback.html#onKeyUp(int, android.view.KeyEvent)): perform clicking of the view when `KEYCODE_DPAD_CENTER` (/reference/android/view/KeyEvent.html#KEYCODE_DPAD_CENTER) or `KEYCODE_ENTER` (/reference/android/view/KeyEvent.html#KEYCODE_ENTER) is released.

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

Parameters

keyCode	A key code that represents the button pressed, from `KeyEvent`.
event	The KeyEvent object that defines the button action.

Returns

If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

public void **onPopulateAccessibilityEvent** (AccessibilityEvent event)

Added in API level 14

Called from `dispatchPopulateAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) giving a chance to this View to populate the accessibility event with its text content. While this method is free to modify event attributes other than text content, doing so should

normally be performed in `onInitializeAccessibilityEvent(AccessibilityEvent)` (/reference/android/view/View.html#onInitializeAccessibilityEvent(android.view.accessibility.AccessibilityEvent)).

Example: Adding formatted date string to an accessibility event in addition to the text added by the super implementation:

```
public void onPopulateAccessibilityEvent(Accessibil
    super.onPopulateAccessibilityEvent(event);
    final int flags = DateUtils.FORMAT_SHOW_DATE | D
    String selectedDateUtterance = DateUtils.formatD
        mCurrentDate.getTimeInMillis(), flags);
    event.getText().add(selectedDateUtterance);
}
```

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

> **Note:** Always call the super implementation before adding information to the event, in case the default implementation has basic information to add.

Parameters

event The accessibility event which to populate.

public boolean **onPreDraw** ()

Added in API level 1

Callback method to be invoked when the view tree is about to be drawn. At this point, all views in the tree have been measured and given a frame. Clients can use this to adjust their scroll bounds or even to request a new layout before drawing occurs.

Returns

Return true to proceed with the current drawing pass, or false to cancel.

public boolean **onPrivateIMECommand** (String action, Bundle data)

Added in API level 3

Called by the framework in response to a private command from the current method, provided by it calling `InputConnection.performPrivateCommand()` (/reference/android/view/inputmethod/InputConnection.html#performPrivateCommand(java.lang.String, android.os.Bundle)).

Parameters

action	The action name of the command.
data	Any additional data for the command. This may be null.

Returns

Return true if you handled the command, else false.

public void **onRestoreInstanceState** (Parcelable state)

Added in API level 1

Hook allowing a view to re-apply a representation of its internal state that had previously been generated by `onSaveInstanceState()` (/reference/android/view/View.html#onSaveInstanceState()). This function will never be called with a null state.

Parameters

state	The frozen state that had previously been returned by `onSaveInstanceState()`.

public void **onRtlPropertiesChanged** (int layoutDirection)

Added in API level 17

Called when any RTL property (layout direction or text direction or text alignment) has been changed. Subclasses need to override this method to take care of cached information that depends on the resolved layout direction, or to inform child views that inherit their layout direction. The default implementation does nothing.

Parameters

layoutDirection	the direction of the layout

public Parcelable **onSaveInstanceState** ()

Added in API level 1

Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state. This state should only contain information that is not persistent or can not be reconstructed later. For example, you will never store your

current position on screen because that will be computed again when a new instance of the view is placed in its view hierarchy.

Some examples of things you may store here: the current cursor position in a text view (but usually not the text itself since that is stored in a content provider or other persistent storage), the currently selected item in a list view.

Returns

Returns a Parcelable object containing the view's current dynamic state, or null if there is nothing interesting to save. The default implementation returns null.

public void **onScreenStateChanged** (int screenState) Added in API level 16

This method is called whenever the state of the screen this view is attached to changes. A state change will usually occurs when the screen turns on or off (whether it happens automatically or the user does it manually.)

Parameters

screenState	The new state of the screen. Can be either SCREEN_STATE_ON or SCREEN_STATE_OFF

public void **onStartTemporaryDetach** () Added in API level 3

This is called when a container is going to temporarily detach a child, with ViewGroup.detachViewFromParent (/reference/android/view/ViewGroup.html#detachViewFromParent(android.view.View)). It will either be followed by onFinishTemporaryDetach() (/reference/android/view/View.html#onFinishTemporaryDetach()) or onDetachedFromWindow() (/reference/android/view/View.html#onDetachedFromWindow()) when the container is done.

public boolean **onTextContextMenuItem** (int id) Added in API level 3

Called when a context menu option for the text view is selected. Currently this will be one of selectAll (/reference/android/R.id.html#selectAll), cut (/reference/android/R.id.html#cut), copy (/reference/android/R.id.html#copy) or paste (/reference/android/R.id.html#paste).

Returns

true if the context menu item action was performed.

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

Implement this method to handle touch screen motion events.

If this method is used to detect click actions, it is recommended that the actions be performed by implementing and calling `performClick()` (/reference/android/view/View.html#performClick()). This will ensure consistent system behavior, including:

- obeying click sound preferences
- dispatching OnClickListener calls
- handling `ACTION_CLICK` when accessibility features are enabled

Parameters

event The motion event.

Returns

True if the event was handled, false otherwise.

public boolean **onTrackballEvent** (MotionEvent event)

Added in API level 1

Implement this method to handle trackball motion events. The *relative* movement of the trackball since the last event can be retrieve with `MotionEvent.getX()` (/reference/android/view/MotionEvent.html#getX()) and `MotionEvent.getY()` (/reference/android/view/MotionEvent.html#getY()). These are normalized so that a movement of 1 corresponds to the user pressing one DPAD key (so they will often be fractional values, representing the more fine-grained movement information available from a trackball).

Parameters

event The motion event.

Returns

True if the event was handled, false otherwise.

public void **onWindowFocusChanged** (boolean hasWindowFocus)

Added in API level 1

Called when the window containing this view gains or loses focus. Note that this is separate from view focus: to receive key events, both your view and its window must have focus. If a window is displayed on top of yours that takes input focus, then your own window will lose focus but the view focus will remain unchanged.

Parameters

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

public boolean **performAccessibilityAction** (int action, Bundle arguments)

Added in API level 16

Performs the specified accessibility action on the view. For possible accessibility actions look at `AccessibilityNodeInfo (/reference/android/view/accessibility/AccessibilityNodeInfo.html)`.

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

Parameters

action	The action to perform.
arguments	Optional action arguments.

Returns

Whether the action was performed.

public boolean **performLongClick** ()

Added in API level 1

Call this view's OnLongClickListener, if it is defined. Invokes the context menu if the OnLongClickListener did not consume the event.

Returns

True if one of the above receivers consumed the event, false otherwise.

public void **removeTextChangedListener** (TextWatcher watcher)

Added in API level 1

Removes the specified TextWatcher from the list of those whose methods are called whenever this TextView's text changes.

public void **sendAccessibilityEvent** (int eventType)

Added in API level 4

Sends an accessibility event of the given type. If accessibility is not enabled this method has no effect. The default implementation calls `onInitializeAccessibilityEvent(AccessibilityEvent) (/reference/android/view/View.html#onInitializeAccessibilityEvent(android.view.accessibilit`

y.AccessibilityEvent)) first to populate information about the event source (this View), then calls dispatchPopulateAccessibilityEvent(AccessibilityEvent) (/reference/android/view/View.html#dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) to populate the text content of the event source including its descendants, and last calls requestSendAccessibilityEvent(View, AccessibilityEvent) (/reference/android/view/ViewParent.html#requestSendAccessibilityEvent(android.view.View, android.view.accessibility.AccessibilityEvent)) on its parent to resuest sending of the event to interested parties.

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

Parameters

eventType	The type of the event to send, as defined by several types from AccessibilityEvent, such as TYPE_VIEW_CLICKED or TYPE_VIEW_HOVER_ENTER.

public void **setAllCaps** (boolean allCaps) Added in API level 14

Sets the properties of this field to transform input to ALL CAPS display. This may use a "small caps" formatting if available. This setting will be ignored if this field is editable or selectable. This call replaces the current transformation method. Disabling this will not necessarily restore the previous behavior from before this was enabled.

Related XML Attributes

android:textAllCaps

See Also

setTransformationMethod(TransformationMethod)

public final void **setAutoLinkMask** (int mask) Added in API level 1

Sets the autolink mask of the text. See `Linkify.ALL` (/reference/android/text/util/Linkify.html#ALL) and peers for possible values.

Related XML Attributes

android:autoLink

public void **setCompoundDrawablePadding** (int pad) Added in API level 1

Sets the size of the padding between the compound drawables and the text.

Related XML Attributes

android:drawablePadding

public void **setCompoundDrawables** (Drawable left, Drawable top, Drawable right, Drawable bottom) Added in API level 1

Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. Use null if you do not want a Drawable there. The Drawables must already have had `setBounds(Rect)` (/reference/android/graphics/drawable/Drawable.html#setBounds(android.graphics.Rect)) called.

Related XML Attributes

android:drawableLeft
android:drawableTop
android:drawableRight
android:drawableBottom

public void **setCompoundDrawablesRelative** (Drawable start, Drawable top, Drawable end, Drawable bottom) Added in API level 17

Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. Use null if you do not want a Drawable there. The Drawables must already have had `setBounds(Rect)` (/reference/android/graphics/drawable/Drawable.html#setBounds(android.graphics.Rect)) called.

Related XML Attributes

android:drawableStart
android:drawableTop
android:drawableEnd
android:drawableBottom

public void **setCompoundDrawablesRelativeWithIntrinsicBounds**

(Drawable start, Drawable top, Drawable end, Drawable bottom) Added in API level 17

Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. Use null if you do not want a Drawable there. The Drawables' bounds will be set to their intrinsic bounds.

Related XML Attributes

android:drawableStart
android:drawableTop
android:drawableEnd
android:drawableBottom

public void
setCompoundDrawablesRelativeWithIntrinsicBounds
(int start, int top, int end, int bottom) Added in API level 17

Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. Use 0 if you do not want a Drawable there. The Drawables' bounds will be set to their intrinsic bounds.

Related XML Attributes

android:drawableStart
android:drawableTop
android:drawableEnd
android:drawableBottom

Parameters

start	Resource identifier of the start Drawable.
top	Resource identifier of the top Drawable.
end	Resource identifier of the end Drawable.
bottom	Resource identifier of the bottom Drawable.

public void
setCompoundDrawablesWithIntrinsicBounds
(Drawable left, Drawable top, Drawable right, Drawable bottom) Added in API level 1

Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. Use null if you do not want a Drawable there. The Drawables' bounds will be set to their intrinsic bounds.

Related XML Attributes

android:drawableLeft
android:drawableTop
android:drawableRight
android:drawableBottom

public void
setCompoundDrawablesWithIntrinsicBounds (int left, int top, int right, int bottom) Added in API level 3

Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. Use 0 if you do not want a Drawable there. The Drawables' bounds will be set to their intrinsic bounds.

Related XML Attributes

android:drawableLeft
android:drawableTop
android:drawableRight
android:drawableBottom

Parameters

left	Resource identifier of the left Drawable.
top	Resource identifier of the top Drawable.
right	Resource identifier of the right Drawable.
bottom	Resource identifier of the bottom Drawable.

public void **setCursorVisible** (boolean visible) Added in API level 1

Set whether the cursor is visible. The default is true. Note that this property only makes sense for editable TextView.

Related XML Attributes

android:cursorVisible

See Also

`isCursorVisible()`

public void **setCustomSelectionActionModeCallback** (ActionMode.Callback actionModeCallback) Added in API level 11

If provided, this ActionMode.Callback will be used to create the ActionMode when text selection is initiated in this View. The standard implementation populates the menu with a subset of Select All, Cut, Copy and Paste actions, depending on what this View supports. A custom implementation can add new entries in the default menu in its `onPrepareActionMode(ActionMode, Menu)` (/reference/android/view/ActionMode.Callback.html#onPrepareActionMode(android.view.ActionMode, android.view.Menu)) method. The default actions can also be removed from the menu using `removeItem(int)` (/reference/android/view/Menu.html#removeItem(int)) and passing `selectAll` (/reference/android/R.id.html#selectAll), `cut` (/reference/android/R.id.html#cut), `copy` (/reference/android/R.id.html#copy) or `paste` (/reference/android/R.id.html#paste) ids as parameters. Returning false from

onCreateActionMode(ActionMode, Menu) (/reference/android/view/ActionMode.Callback.html#onCreateActionMode(android.view.ActionMode, android.view.Menu)) will prevent the action mode from being started. Action click events should be handled by the custom implementation of onActionItemClicked(ActionMode, MenuItem) (/reference/android/view/ActionMode.Callback.html#onActionItemClicked(android.view.ActionMode, android.view.MenuItem)). Note that text selection mode is not started when a TextView receives focus and the selectAllOnFocus (/reference/android/R.attr.html#selectAllOnFocus) flag has been set. The content is highlighted in that case, to allow for quick replacement.

public final void **setEditableFactory** (Editable.Factory factory)

Added in API level 1

Sets the Factory used to create new Editables.

public void **setEllipsize** (TextUtils.TruncateAt where)

Added in API level 1

Causes words in the text that are longer than the view is wide to be ellipsized instead of broken in the middle. You may also want to setSingleLine() (/reference/android/widget/TextView.html#setSingleLine()) or setHorizontallyScrolling(boolean) (/reference/android/widget/TextView.html#setHorizontallyScrolling(boolean)) to constrain the text to a single line. Use null to turn off ellipsizing. If setMaxLines(int) (/reference/android/widget/TextView.html#setMaxLines(int)) has been used to set two or more lines, END (/reference/android/text/TextUtils.TruncateAt.html#END) and MARQUEE (/reference/android/text/TextUtils.TruncateAt.html#MARQUEE)* are only supported (other ellipsizing types will not do anything).

Related XML Attributes

android:ellipsize

public void **setEms** (int ems)

Added in API level 1

Makes the TextView exactly this many ems wide

Related XML Attributes

android:ems

See Also

setMaxEms(int)
setMinEms(int)
getMinEms()

getMaxEms()

public void **setEnabled** (boolean enabled) Added in API level 1

Set the enabled state of this view. The interpretation of the enabled state varies by subclass.

Parameters

enabled True if this view is enabled, false otherwise.

public void **setError** (CharSequence error) Added in API level 1

Sets the right-hand compound drawable of the TextView to the "error" icon and sets an error message that will be displayed in a popup when the TextView has focus. The icon and error message will be reset to null when any key events cause changes to the TextView's text. If the `error` is `null`, the error message and icon will be cleared.

public void **setError** (CharSequence error, Drawable icon) Added in API level 1

Sets the right-hand compound drawable of the TextView to the specified icon and sets an error message that will be displayed in a popup when the TextView has focus. The icon and error message will be reset to null when any key events cause changes to the TextView's text. The drawable must already have had `setBounds(Rect)` (/reference/android/graphics/drawable/Drawable.html#setBounds(android.graphics.Rect)) set on it. If the `error` is `null`, the error message will be cleared (and you should provide a `null` icon as well).

public void **setExtractedText** (ExtractedText text) Added in API level 3

Apply to this text view the given extracted text, as previously returned by `extractText(ExtractedTextRequest, ExtractedText)` (/reference/android/widget/TextView.html#extractText(android.view.inputmethod.ExtractedTextRequest, android.view.inputmethod.ExtractedText)).

public void **setFilters** (InputFilter[] filters) Added in API level 1

Sets the list of input filters that will be used if the buffer is Editable. Has no effect otherwise.

Related XML Attributes

android:maxLength

public void **setFreezesText** (boolean freezesText) Added in API level 1

Control whether this text view saves its entire text contents when freezing to an icicle, in addition to dynamic state such as cursor position. By default this is false, not saving the text. Set to true if the text in the text view is not being saved somewhere else in persistent storage (such as in a content provider) so that if the view is later thawed the user will not lose their data.

Related XML Attributes

android:freezesText

Parameters

freezesText	Controls whether a frozen icicle should include the entire text data: true to include it, false to not.

public void **setGravity** (int gravity) Added in API level 1

Sets the horizontal alignment of the text and the vertical gravity that will be used when there is extra space in the TextView beyond what is required for the text itself.

Related XML Attributes

android:gravity

See Also

`Gravity`

public void **setHeight** (int pixels) Added in API level 1

Makes the TextView exactly this many pixels tall. You could do the same thing by specifying this number in the LayoutParams. Note that setting this value overrides any other (minimum / maximum) number of lines or height setting.

Related XML Attributes

android:height

public void **setHighlightColor** (int color) Added in API level 1

Sets the color used to display the selection highlight.

Related XML Attributes

android:textColorHighlight

public final void **setHint** (CharSequence hint) Added in API level 1

Sets the text to be displayed when the text of the TextView is empty. Null means to use the normal empty text. The hint does not currently

participate in determining the size of the view.

Related XML Attributes

android:hint

public final void **setHint** (int resid) Added in API level 1

Sets the text to be displayed when the text of the TextView is empty, from a resource.

Related XML Attributes

android:hint

public final void **setHintTextColor** (ColorStateList colors) Added in API level 1

Sets the color of the hint text.

Related XML Attributes

android:textColorHint

See Also

getHintTextColors()
setHintTextColor(int)
setTextColor(ColorStateList)
setLinkTextColor(ColorStateList)

public final void **setHintTextColor** (int color) Added in API level 1

Sets the color of the hint text for all the states (disabled, focussed, selected...) of this TextView.

Related XML Attributes

android:textColorHint

See Also

setHintTextColor(ColorStateList)
getHintTextColors()
setTextColor(int)

public void **setHorizontallyScrolling** (boolean whether) Added in API level 1

Sets whether the text should be allowed to be wider than the View is. If false, it will be wrapped to the width of the View.

Related XML Attributes

android:scrollHorizontally

public void **setImeActionLabel** (CharSequence label, int actionId) Added in API level 3

Change the custom IME action associated with the text view, which will be reported to an IME with `actionLabel (/reference/android/view/inputmethod/EditorInfo.html#actionLabel)` and `actionId (/reference/android/view/inputmethod/EditorInfo.html#actionId)` when it has focus.

Related XML Attributes

android:imeActionLabel
android:imeActionId

See Also

`getImeActionLabel()`
`getImeActionId()`
`EditorInfo`

public void **setImeOptions** (int imeOptions) Added in API level 3

Change the editor type integer associated with the text view, which will be reported to an IME with `imeOptions (/reference/android/view/inputmethod/EditorInfo.html#imeOptions)` when it has focus.

Related XML Attributes

android:imeOptions

See Also

`getImeOptions()`
`EditorInfo`

public void **setIncludeFontPadding** (boolean includepad) Added in API level 1

Set whether the TextView includes extra top and bottom padding to make room for accents that go above the normal ascent and descent. The default is true.

Related XML Attributes

android:includeFontPadding

See Also

`getIncludeFontPadding()`

public void **setInputExtras** (int xmlResId) Added in API level 3

Set the extra input data of the text, which is the `TextBoxAttribute.extras (/reference/android/view/inputmethod/EditorInfo.html#extras)` Bundle that will be filled in when creating an

input connection. The given integer is the resource ID of an XML resource holding an `<input-extras>` (/reference/android/R.styleable.html#InputExtras) XML tree.

Related XML Attributes

android:editorExtras

Throws

XmlPullParserException

IOException

See Also

`getInputExtras(boolean)`
`extras`

public void **setInputType** (int type)

Added in API level 3

Set the type of the content with a constant as defined for `inputType` (/reference/android/view/inputmethod/EditorInfo.html#inputType). This will take care of changing the key listener, by calling `setKeyListener(KeyListener)` (/reference/android/widget/TextView.html#setKeyListener(android.text.method.KeyListener)), to match the given content type. If the given content type is `TYPE_NULL` (/reference/android/text/InputType.html#TYPE_NULL) then a soft keyboard will not be displayed for this text view. Note that the maximum number of displayed lines (see `setMaxLines(int)` (/reference/android/widget/TextView.html#setMaxLines(int))) will be modified if you change the `TYPE_TEXT_FLAG_MULTI_LINE` (/reference/android/text/InputType.html#TYPE_TEXT_FLAG_MULTI_LINE) flag of the input type.

Related XML Attributes

android:inputType

See Also

`getInputType()`
`setRawInputType(int)`
`InputType`

public void **setKeyListener** (KeyListener input)

Added in API level 1

Sets the key listener to be used with this TextView. This can be null to disallow user input. Note that this method has significant and subtle interactions with soft keyboards and other input method: see `KeyListener.getContentType()` (/reference/android/text/method/KeyListener.html#getInputType()) for important details. Calling this method will replace the current content type of the text view with the content type returned by the key listener.

Be warned that if you want a TextView with a key listener or movement method not to be focusable, or if you want a TextView without a key listener or movement method to be focusable, you must call `setFocusable(boolean)` (/reference/android/view/View.html#setFocusable(boolean)) again after calling this to get the focusability back the way you want it.

Related XML Attributes

android:numeric
android:digits
android:phoneNumber
android:inputMethod
android:capitalize
android:autoText

public void **setLineSpacing** (float add, float mult)

Added in API level 1

Sets line spacing for this TextView. Each line will have its height multiplied by `mult` and have `add` added to it.

Related XML Attributes

android:lineSpacingExtra
android:lineSpacingMultiplier

public void **setLines** (int lines)

Added in API level 1

Makes the TextView exactly this many lines tall. Note that setting this value overrides any other (minimum / maximum) number of lines or height setting. A single line TextView will set this value to 1.

Related XML Attributes

android:lines

public final void **setLinkTextColor** (ColorStateList colors)

Added in API level 1

Sets the color of links in the text.

Related XML Attributes

android:textColorLink

See Also

`setLinkTextColor(int)`
`getLinkTextColors()`
`setTextColor(ColorStateList)`
`setHintTextColor(ColorStateList)`

public final void **setLinkTextColor** (int color)

Added in API level 1

Sets the color of links in the text.

Related XML Attributes
android:textColorLink

See Also
`setLinkTextColor(ColorStateList)`
`getLinkTextColors()`

public final void **setLinksClickable** (boolean whether) Added in API level 1

Sets whether the movement method will automatically be set to `LinkMovementMethod` (/reference/android/text/method/LinkMovementMethod.html) if `setAutoLinkMask(int)` (/reference/android/widget/TextView.html#setAutoLinkMask(int)) has been set to nonzero and links are detected in `setText(char[], int, int)` (/reference/android/widget/TextView.html#setText(char[], int, int)). The default is true.

Related XML Attributes
android:linksClickable

public void **setMarqueeRepeatLimit** (int marqueeLimit) Added in API level 2

Sets how many times to repeat the marquee animation. Only applied if the TextView has marquee enabled. Set to -1 to repeat indefinitely.

Related XML Attributes
android:marqueeRepeatLimit

See Also
`getMarqueeRepeatLimit()`

public void **setMaxEms** (int maxems) Added in API level 1

Makes the TextView at most this many ems wide

Related XML Attributes
android:maxEms

public void **setMaxHeight** (int maxHeight) Added in API level 1

Makes the TextView at most this many pixels tall. This option is mutually exclusive with the `setMaxLines(int)` (/reference/android/widget/TextView.html#setMaxLines(int)) method. Setting this value overrides any other (maximum) number of lines setting.

Related XML Attributes

android:maxHeight

public void **setMaxLines** (int maxlines) Added in API level 1

Makes the TextView at most this many lines tall. Setting this value overrides any other (maximum) height setting.

Related XML Attributes

android:maxLines

public void **setMaxWidth** (int maxpixels) Added in API level 1

Makes the TextView at most this many pixels wide

Related XML Attributes

android:maxWidth

public void **setMinEms** (int minems) Added in API level 1

Makes the TextView at least this many ems wide

Related XML Attributes

android:minEms

public void **setMinHeight** (int minHeight) Added in API level 1

Makes the TextView at least this many pixels tall. Setting this value overrides any other (minimum) number of lines setting.

Related XML Attributes

android:minHeight

public void **setMinLines** (int minlines) Added in API level 1

Makes the TextView at least this many lines tall. Setting this value overrides any other (minimum) height setting. A single line TextView will set this value to 1.

Related XML Attributes

android:minLines

See Also

`getMinLines()`

public void **setMinWidth** (int minpixels) Added in API level 1

Makes the TextView at least this many pixels wide

Related XML Attributes

android:minWidth

public final void **setMovementMethod** (MovementMethod movement)

Added in API level 1

Sets the movement method (arrow key handler) to be used for this TextView. This can be null to disallow using the arrow keys to move the cursor or scroll the view.

Be warned that if you want a TextView with a key listener or movement method not to be focusable, or if you want a TextView without a key listener or movement method to be focusable, you must call `setFocusable(boolean)` (/reference/android/view/View.html#setFocusable(boolean)) again after calling this to get the focusability back the way you want it.

public void **setOnEditorActionListener** (TextView.OnEditorActionListener l)

Added in API level 3

Set a special listener to be called when an action is performed on the text view. This will be called when the enter key is pressed, or when an action supplied to the IME is selected by the user. Setting this means that the normal hard key event will not insert a newline into the text view, even if it is multi-line; holding down the ALT modifier will, however, allow the user to insert a newline character.

public void **setPadding** (int left, int top, int right, int bottom)

Added in API level 1

Sets the padding. The view may add on the space required to display the scrollbars, depending on the style and visibility of the scrollbars. So the values returned from `getPaddingLeft()` (/reference/android/view/View.html#getPaddingLeft()), `getPaddingTop()` (/reference/android/view/View.html#getPaddingTop()), `getPaddingRight()` (/reference/android/view/View.html#getPaddingRight()) and `getPaddingBottom()` (/reference/android/view/View.html#getPaddingBottom()) may be different from the values set in this call.

Parameters

left	the left padding in pixels
top	the top padding in pixels
right	the right padding in pixels
bottom	the bottom padding in pixels

public void **setPaddingRelative** (int start, int top, int

end, int bottom) Added in API level 16

Sets the relative padding. The view may add on the space required to display the scrollbars, depending on the style and visibility of the scrollbars. So the values returned from `getPaddingStart()` (/reference/android/view/View.html#getPaddingStart()), `getPaddingTop()` (/reference/android/view/View.html#getPaddingTop()), `getPaddingEnd()` (/reference/android/view/View.html#getPaddingEnd()) and `getPaddingBottom()` (/reference/android/view/View.html#getPaddingBottom()) may be different from the values set in this call.

Parameters

start	the start padding in pixels
top	the top padding in pixels
end	the end padding in pixels
bottom	the bottom padding in pixels

public void **setPaintFlags** (int flags) Added in API level 1

Sets flags on the Paint being used to display the text and reflows the text if they are different from the old flags.

See Also

`setFlags(int)`

public void **setPrivateImeOptions** (String type) Added in API level 3

Set the private content type of the text, which is the `EditorInfo.privateImeOptions` (/reference/android/view/inputmethod/EditorInfo.html#privateImeOptions) field that will be filled in when creating an input connection.

Related XML Attributes

android:privateImeOptions

See Also

`getPrivateImeOptions()`
`privateImeOptions`

public void **setRawInputType** (int type) Added in API level 3

Directly change the content type integer of the text view, without modifying any other state.

Related XML Attributes

android:inputType

See Also

setInputType(int)
InputType

public void **setScroller** (Scroller s) Added in API level 1

public void **setSelectAllOnFocus** (boolean selectAllOnFocus) Added in API level 1

Set the TextView so that when it takes focus, all the text is selected.

Related XML Attributes

android:selectAllOnFocus

public void **setSelected** (boolean selected) Added in API level 1

Changes the selection state of this view. A view can be selected or not. Note that selection is not the same as focus. Views are typically selected in the context of an AdapterView like ListView or GridView; the selected view is the view that is highlighted.

Parameters

selected true if the view must be selected, false otherwise

public void **setShadowLayer** (float radius, float dx, float dy, int color) Added in API level 1

Gives the text a shadow of the specified radius and color, the specified distance from its normal position.

Related XML Attributes

android:shadowColor
android:shadowDx
android:shadowDy
android:shadowRadius

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

Sets the properties of this field (lines, horizontally scrolling, transformation method) to be for a single-line input.

Related XML Attributes

android:singleLine

public void **setSingleLine** (boolean singleLine) Added in API level 1

If true, sets the properties of this field (number of lines, horizontally

scrolling, transformation method) to be for a single-line input; if false, restores these to the default conditions. Note that the default conditions are not necessarily those that were in effect prior this method, and you may want to reset these properties to your custom values.

Related XML Attributes

android:singleLine

public final void **setSpannableFactory** (Spannable.Factory factory) Added in API level 1

Sets the Factory used to create new Spannables.

public final void **setText** (int resid) Added in API level 1

public final void **setText** (char[] text, int start, int len) Added in API level 1

Sets the TextView to display the specified slice of the specified char array. You must promise that you will not change the contents of the array except for right before another call to setText(), since the TextView has no way to know that the text has changed and that it needs to invalidate and re-layout.

public final void **setText** (int resid, TextView.BufferType type) Added in API level 1

public final void **setText** (CharSequence text) Added in API level 1

Sets the string value of the TextView. TextView *does not* accept HTML-like formatting, which you can do with text strings in XML resource files. To style your strings, attach android.text.style.* objects to a SpannableString (/reference/android/text/SpannableString.html), or see the Available Resource Types (/guide/topics/resources/available-resources.html#stringresources) documentation for an example of setting formatted text in the XML resource file.

Related XML Attributes

android:text

public void **setText** (CharSequence text, TextView.BufferType type) Added in API level 1

Sets the text that this TextView is to display (see setText(CharSequence) (/reference/android/widget/TextView.html#setText(java.lang.CharSequence)) and also sets whether it is stored in a styleable/spannable buffer and whether it is editable.

Related XML Attributes

android:text
android:bufferType

public void **setTextAppearance** (Context context, int resid) Added in API level 1

Sets the text color, size, style, hint color, and highlight color from the specified TextAppearance resource.

public void **setTextColor** (ColorStateList colors) Added in API level 1

Sets the text color.

Related XML Attributes

android:textColor

See Also

`setTextColor(int)`
`getTextColors()`
`setHintTextColor(ColorStateList)`
`setLinkTextColor(ColorStateList)`

public void **setTextColor** (int color) Added in API level 1

Sets the text color for all the states (normal, selected, focused) to be this color.

Related XML Attributes

android:textColor

See Also

`setTextColor(ColorStateList)`
`getTextColors()`

public void **setTextIsSelectable** (boolean selectable) Added in API level 11

Sets whether the content of this view is selectable by the user. The default is `false`, meaning that the content is not selectable.

When you use a TextView to display a useful piece of information to the user (such as a contact's address), make it selectable, so that the user can select and copy its content. You can also use set the XML attribute `TextView_textIsSelectable (/reference/android/R.styleable.html#TextView_textIsSelectable)` to "true".

When you call this method to set the value of `textIsSelectable`, it sets the flags `focusable`, `focusableInTouchMode`, `clickable`, and `longClickable` to the same value. These flags correspond to

the attributes android:focusable (/reference/android/R.styleable.html#View_focusable), android:focusableInTouchMode (/reference/android/R.styleable.html#View_focusableInTouchMode), android:clickable (/reference/android/R.styleable.html#View_clickable), and android:longClickable (/reference/android/R.styleable.html#View_longClickable). To restore any of these flags to a state you had set previously, call one or more of the following methods: setFocusable() (/reference/android/view/View.html#setFocusable(boolean)), setFocusableInTouchMode() (/reference/android/view/View.html#setFocusableInTouchMode(boolean)), setClickable() (/reference/android/view/View.html#setClickable(boolean)) or setLongClickable() (/reference/android/view/View.html#setLongClickable(boolean)).

Parameters

selectable	Whether the content of this TextView should be selectable.

public final void **setTextKeepState** (CharSequence text) Added in API level 1

Like setText(CharSequence) (/reference/android/widget/TextView.html#setText(java.lang.CharSequence)), except that the cursor position (if any) is retained in the new text.

Parameters

text	The new text to place in the text view.

See Also

setText(CharSequence)

public final void **setTextKeepState** (CharSequence text, TextView.BufferType type) Added in API level 1

Like setText(CharSequence, android.widget.TextView.BufferType) (/reference/android/widget/TextView.html#setText(java.lang.CharSequence, android.widget.TextView.BufferType)), except that the cursor position (if any) is retained in the new text.

See Also

setText(CharSequence, android.widget.TextView.BufferType)

public void **setTextLocale** (Locale locale) Added in API level 17

Set the default `Locale (/reference/java/util/Locale.html)` of the text in this TextView to the given value. This value is used to choose appropriate typefaces for ambiguous characters. Typically used for CJK locales to disambiguate Hanzi/Kanji/Hanja characters.

Parameters

locale the `Locale` for drawing text, must not be null.

See Also

`setTextLocale(Locale)`

public void **setTextScaleX** (float size) Added in API level 1

Sets the extent by which text should be stretched horizontally.

Related XML Attributes

android:textScaleX

public void **setTextSize** (float size) Added in API level 1

Set the default text size to the given value, interpreted as "scaled pixel" units. This size is adjusted based on the current density and user font size preference.

Related XML Attributes

android:textSize

Parameters

size The scaled pixel size.

public void **setTextSize** (int unit, float size) Added in API level 1

Set the default text size to a given unit and value. See `TypedValue (/reference/android/util/TypedValue.html)` for the possible dimension units.

Related XML Attributes

android:textSize

Parameters

unit The desired dimension unit.

size The desired size in the given units.

public final void **setTransformationMethod** (TransformationMethod method) Added in API level 1

Sets the transformation that is applied to the text that this TextView is displaying.

Related XML Attributes

android:password
android:singleLine

public void **setTypeface** (Typeface tf, int style)

Added in API level 1

Sets the typeface and style in which the text should be displayed, and turns on the fake bold and italic bits in the Paint if the Typeface that you provided does not have all the bits in the style that you specified.

Related XML Attributes

android:typeface
android:textStyle

public void **setTypeface** (Typeface tf)

Added in API level 1

Sets the typeface and style in which the text should be displayed. Note that not all Typeface families actually have bold and italic variants, so you may need to use `setTypeface(Typeface, int)` `(/reference/android/widget/TextView.html#setTypeface(android.graphics.Typeface, int))` to get the appearance that you actually want.

Related XML Attributes

android:fontFamily
android:typeface
android:textStyle

See Also

`getTypeface()`

public void **setWidth** (int pixels)

Added in API level 1

Makes the TextView exactly this many pixels wide. You could do the same thing by specifying this number in the LayoutParams.

Related XML Attributes

android:width

See Also

`setMaxWidth(int)`
`setMinWidth(int)`
`getMinWidth()`
`getMaxWidth()`

Protected Methods

protected int **computeHorizontalScrollRange** () Added in API level 1

Compute the horizontal range that the horizontal scrollbar represents.

The range is expressed in arbitrary units that must be the same as the units used by `computeHorizontalScrollExtent()` (/reference/android/view/View.html#computeHorizontalScrollExtent()) and `computeHorizontalScrollOffset()` (/reference/android/view/View.html#computeHorizontalScrollOffset()).

The default range is the drawing width of this view.

Returns

the total horizontal range represented by the horizontal scrollbar

protected int **computeVerticalScrollExtent** () Added in API level 1

Compute the vertical extent of the horizontal scrollbar's thumb within the vertical range. This value is used to compute the length of the thumb within the scrollbar's track.

The range is expressed in arbitrary units that must be the same as the units used by `computeVerticalScrollRange()` (/reference/android/view/View.html#computeVerticalScrollRange()) and `computeVerticalScrollOffset()` (/reference/android/view/View.html#computeVerticalScrollOffset()).

The default extent is the drawing height of this view.

Returns

the vertical extent of the scrollbar's thumb

protected int **computeVerticalScrollRange** () Added in API level 1

Compute the vertical range that the vertical scrollbar represents.

The range is expressed in arbitrary units that must be the same as the units used by `computeVerticalScrollExtent()` (/reference/android/view/View.html#computeVerticalScrollExtent()) and `computeVerticalScrollOffset()` (/reference/android/view/View.html#computeVerticalScrollOffset()).

Returns

the total vertical range represented by the vertical scrollbar
The default range is the drawing height of this view.

protected void **drawableStateChanged** ()

Added in API level 1

This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown.

Be sure to call through to the superclass when overriding this function.

protected int **getBottomPaddingOffset** ()

Added in API level 2

Amount by which to extend the bottom fading region. Called only when `isPaddingOffsetRequired() (/reference/android/view/View.html#isPaddingOffsetRequired())` returns true.

Returns

The bottom padding offset in pixels.

protected boolean **getDefaultEditable** ()

Added in API level 1

Subclasses override this to specify that they have a KeyListener by default even if not specifically called for in the XML options.

protected MovementMethod **getDefaultMovementMethod** ()

Added in API level 1

Subclasses override this to specify a default movement method.

protected float **getLeftFadingEdgeStrength** ()

Added in API level 1

Returns the strength, or intensity, of the left faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

Returns

the intensity of the left fade as a float between 0.0f and 1.0f

protected int **getLeftPaddingOffset** ()

Added in API level 2

Amount by which to extend the left fading region. Called only when `isPaddingOffsetRequired() (/reference/android/view/View.html#isPaddingOffsetRequired())` returns true.

Returns

The left padding offset in pixels.

protected float **getRightFadingEdgeStrength** ()

Returns the strength, or intensity, of the right faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

Returns

the intensity of the right fade as a float between 0.0f and 1.0f

protected int **getRightPaddingOffset** ()

Added in API level 2

Amount by which to extend the right fading region. Called only when `isPaddingOffsetRequired()` (/reference/android/view/View.html#isPaddingOffsetRequired()) returns true.

Returns

The right padding offset in pixels.

protected int **getTopPaddingOffset** ()

Added in API level 2

Amount by which to extend the top fading region. Called only when `isPaddingOffsetRequired()` (/reference/android/view/View.html#isPaddingOffsetRequired()) returns true.

Returns

The top padding offset in pixels.

protected boolean **isPaddingOffsetRequired** ()

Added in API level 2

If the View draws content inside its padding and enables fading edges, it needs to support padding offsets. Padding offsets are added to the fading edges to extend the length of the fade so that it covers pixels drawn inside the padding. Subclasses of this class should override this method if they need to draw content inside the padding.

Returns

True if padding offset must be applied, false otherwise.

protected void **onAttachedToWindow** ()

Added in API level 1

This is called when the view is attached to a window. At this point it has a Surface and will start drawing. Note that this function is guaranteed to be called before `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)), however it may be called any time before the first onDraw -- including before or after `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)).

protected int[] **onCreateDrawableState** (int extraSpace)

Added in API level 1

Generate the new `Drawable` (/reference/android/graphics/drawable/Drawable.html) state for this view. This is called by the view system when the cached Drawable state is determined to be invalid. To retrieve the current state, you should use `getDrawableState()` (/reference/android/view/View.html#getDrawableState()).

Parameters

extraSpace	if non-zero, this is the number of extra entries you would like in the returned array in which you can place your own states.

Returns

Returns an array holding the current `Drawable` state of the view.

protected void **onDetachedFromWindow** ()

Added in API level 1

This is called when the view is detached from a window. At this point it no longer has a surface for drawing.

protected void **onDraw** (Canvas canvas)

Added in API level 1

Implement this to do your drawing.

Parameters

canvas	the canvas on which the background will be drawn

protected void **onFocusChanged** (boolean focused, int direction, Rect previouslyFocusedRect)

Added in API level 1

Called by the view system when the focus state of this view changes. When the focus change event is caused by directional navigation, direction and previouslyFocusedRect provide insight into where the focus is coming from. When overriding, be sure to call up through to the super class so that the standard focus handling will occur.

Parameters

focused	True if the View has focus; false otherwise.
direction	The direction focus has moved when requestFocus() is called to give this view focus. Values are `FOCUS_UP`, `FOCUS_DOWN`, `FOCUS_LEFT`, `FOCUS_RIGHT`, `FOCUS_FORWARD`, or `FOCUS_BACKWARD`. It may not always

	apply, in which case use the default.
previouslyFocusedRect	The rectangle, in this view's coordinate system, of the previously focused view. If applicable, this will be passed in as finer grained information about where the focus is coming from (in addition to direction). Will be `null` otherwise.

protected void **onLayout** (boolean changed, int left, int top, int right, int bottom)

Added in API level 1

Called from layout when this view should assign a size and position to each of its children. Derived classes with children should override this method and call layout on each of their children.

Parameters

changed	This is a new size or position for this view
left	Left position, relative to parent
top	Top position, relative to parent
right	Right position, relative to parent
bottom	Bottom position, relative to parent

protected void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec)

Added in API level 1

Measure the view and its content to determine the measured width and the measured height. This method is invoked by `measure(int, int)` (/reference/android/view/View.html#measure(int, int)) and should be overriden by subclasses to provide accurate and efficient measurement of their contents.

CONTRACT: When overriding this method, you *must* call `setMeasuredDimension(int, int)` (/reference/android/view/View.html#setMeasuredDimension(int, int)) to store the measured width and height of this view. Failure to do so will trigger an `IllegalStateException`, thrown by `measure(int, int)` (/reference/android/view/View.html#measure(int, int)). Calling the superclass' `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)) is a valid use.

The base class implementation of measure defaults to the background size, unless a larger size is allowed by the MeasureSpec. Subclasses should override `onMeasure(int, int)` (/reference/android/view/View.html#onMeasure(int, int)) to provide better

measurements of their content.

If this method is overridden, it is the subclass's responsibility to make sure the measured height and width are at least the view's minimum height and width (getSuggestedMinimumHeight() (/reference/android/view/View.html#getSuggestedMinimumHeight()) and getSuggestedMinimumWidth() (/reference/android/view/View.html#getSuggestedMinimumWidth())).

Parameters

widthMeasureSpec	horizontal space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec.
heightMeasureSpec	vertical space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec.

protected void **onScrollChanged** (int horiz, int vert, int oldHoriz, int oldVert) Added in API level 1

This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). This is typically as a result of scrollBy(int, int) (/reference/android/view/View.html#scrollBy(int, int)) or scrollTo(int, int) (/reference/android/view/View.html#scrollTo(int, int)) having been called.

Parameters

horiz	Current horizontal scroll origin.
vert	Current vertical scroll origin.
oldHoriz	Previous horizontal scroll origin.
oldVert	Previous vertical scroll origin.

protected void **onSelectionChanged** (int selStart, int selEnd) Added in API level 3

This method is called when the selection has changed, in case any subclasses would like to know.

Parameters

selStart	The new selection start location.
selEnd	The new selection end location.

protected void **onTextChanged** (CharSequence text, int start, int lengthBefore, int lengthAfter)

This method is called when the text is changed, in case any subclasses would like to know. Within `text`, the `lengthAfter` characters beginning at `start` have just replaced old text that had length `lengthBefore`. It is an error to attempt to make changes to `text` from this callback.

Parameters

text	The text the TextView is displaying
start	The offset of the start of the range of the text that was modified
lengthBefore	The length of the former text that has been replaced
lengthAfter	The length of the replacement modified text

protected void **onVisibilityChanged** (View changedView, int visibility)

Added in API level 8

Called when the visibility of the view or an ancestor of the view is changed.

Parameters

changedView	The view whose visibility changed. Could be 'this' or an ancestor view.
visibility	The new visibility of changedView: `VISIBLE`, `INVISIBLE` or `GONE`.

protected boolean **setFrame** (int l, int t, int r, int b)

Added in API level 1

Assign a size and position to this view. This is called from layout.

Parameters

l	Left position, relative to parent
t	Top position, relative to parent
r	Right position, relative to parent
b	Bottom position, relative to parent

Returns

true if the new size and position are different than the previous ones

protected boolean **verifyDrawable** (Drawable who)

Added in API level 1

If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying. This allows animations for those drawables to be scheduled.

Be sure to call through to the super class when overriding this

function.

Parameters

who The Drawable to verify. Return true if it is one you are displaying, else return the result of calling through to the super class.

Returns

boolean If true than the Drawable is being displayed in the view; else false and it is not allowed to animate.