public class
ScrollView
extends FrameLayout

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

java.lang.Object
↳android.view.View
↳android.view.ViewGroup
↳android.widget.FrameLayout
↳android.widget.ScrollView

Class Overview

Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display. A ScrollView is a `FrameLayout` (/reference/android/widget/FrameLayout.html), meaning you should place one child in it containing the entire contents to scroll; this child may itself be a layout manager with a complex hierarchy of objects. A child that is often used is a `LinearLayout` (/reference/android/widget/LinearLayout.html) in a vertical orientation, presenting a vertical array of top-level items that the user can scroll through.

You should never use a ScrollView with a `ListView` (/reference/android/widget/ListView.html), because ListView takes care of its own vertical scrolling. Most importantly, doing this defeats all of the important optimizations in ListView for dealing with large lists, since it effectively forces the ListView to display its entire list of items to fill up the infinite container supplied by ScrollView.

The `TextView` (/reference/android/widget/TextView.html) class also takes care of its own scrolling, so does not require a ScrollView, but using the two together is possible to achieve the effect of a text view within a larger container.

ScrollView only supports vertical scrolling. For horizontal scrolling, use `HorizontalScrollView` (/reference/android/widget/HorizontalScrollView.html).

Summary

XML Attributes

Attribute Name	*Related Method*	*Description*
android:fillViewport	setFillViewport(boolean)	Defines whether the scrollview should stretch its content to fill the viewport.

Inherited XML Attributes [Expand]

▶ From class android.widget.FrameLayout
▶ From class android.view.ViewGroup
▶ From class android.view.View

Inherited Constants [Expand]

▶ From class android.view.ViewGroup
▶ From class android.view.View

Inherited Fields [Expand]

▶ From class android.view.View

Public Constructors

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

Public Methods

void	addView (View child) Adds a child view.
void	addView (View child, int index) Adds a child view.
void	addView (View child, int index, ViewGroup.LayoutParams params) Adds a child view with the specified layout parameters.
void	addView (View child, ViewGroup.LayoutParams params) Adds a child view with the specified layout parameters.
boolean	arrowScroll (int direction) Handle scrolling in response to an up or down arrow click.
void	computeScroll () Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary.

boolean	dispatchKeyEvent (KeyEvent event) Dispatch a key event to the next view on the focus path.
void	draw (Canvas canvas) Manually render this view (and all of its children) to the given Canvas.
boolean	executeKeyEvent (KeyEvent event) You can call this function yourself to have the scroll view perform scrolling from a key event, just as if the event had been dispatched to it by the view hierarchy.
void	fling (int velocityY) Fling the scroll view
boolean	fullScroll (int direction) Handles scrolling in response to a "home/end" shortcut press.
int	getMaxScrollAmount ()
boolean	isFillViewport () Indicates whether this ScrollView's content is stretched to fill the viewport.
boolean	isSmoothScrollingEnabled ()
boolean	onGenericMotionEvent (MotionEvent event) Implement this method to handle generic motion events.
void	onInitializeAccessibilityEvent (AccessibilityEvent event) Initializes an `AccessibilityEvent` with information about this View which is the event source.
void	onInitializeAccessibilityNodeInfo (AccessibilityNodeInfo info) Initializes an `AccessibilityNodeInfo` with information about this view.
boolean	onInterceptTouchEvent (MotionEvent ev) Implement this method to intercept all touch screen motion events.
boolean	onTouchEvent (MotionEvent ev) Implement this method to handle touch screen motion events.
boolean	pageScroll (int direction) Handles scrolling in response to a "page up/down" shortcut press.
boolean	performAccessibilityAction (int action, Bundle arguments) Performs the specified accessibility action on the view.
void	requestChildFocus (View child, View focused) Called when a child of this parent wants focus
boolean	requestChildRectangleOnScreen (View child, Rect rectangle, boolean immediate) Called when a child of this group wants a particular rectangle to be positioned onto the screen.
void	requestDisallowInterceptTouchEvent (boolean disallowIntercept) Called when a child does not want this parent and its ancestors to intercept touch events with `onInterceptTouchEvent(MotionEvent)`.
void	requestLayout () Call this when something has changed which has invalidated the layout of this view.
void	scrollTo (int x, int y) Set the scrolled position of your view. This version also clamps the scrolling to the bounds of our child.
void	setFillViewport (boolean fillViewport) Indicates this ScrollView whether it should stretch its content height to fill the viewport or not.
void	setOverScrollMode (int mode) Set the over-scroll mode for this view.
void	setSmoothScrollingEnabled (boolean smoothScrollingEnabled) Set whether arrow scrolling will animate its transition.
boolean	shouldDelayChildPressedState () Return true if the pressed state should be delayed for children or descendants of this ViewGroup.
final void	smoothScrollBy (int dx, int dy) Like `scrollBy(int, int)`, but scroll smoothly instead of immediately.
final void	smoothScrollTo (int x, int y) Like `scrollTo(int, int)`, but scroll smoothly instead of immediately.

Protected Methods

int	computeScrollDeltaToGetChildRectOnScreen (Rect rect) Compute the amount to scroll in the Y direction in order to get a rectangle completely on the screen (or, if taller than the screen, at least the first screen size chunk of it).

int	computeVerticalScrollOffset () Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range.
int	computeVerticalScrollRange () The scroll range of a scroll view is the overall height of all of its children.
float	getBottomFadingEdgeStrength () Returns the strength, or intensity, of the bottom faded edge.
float	getTopFadingEdgeStrength () Returns the strength, or intensity, of the top faded edge.
void	measureChild (View child, int parentWidthMeasureSpec, int parentHeightMeasureSpec) Ask one of the children of this view to measure itself, taking into account both the MeasureSpec requirements for this view and its padding.
void	measureChildWithMargins (View child, int parentWidthMeasureSpec, int widthUsed, int parentHeightMeasureSpec, int heightUsed) Ask one of the children of this view to measure itself, taking into account both the MeasureSpec requirements for this view and its padding and margins.
void	onDetachedFromWindow () This is called when the view is detached from a window.
void	onLayout (boolean changed, int l, int t, int r, int b) Called from layout when this view should assign a size and position to each of its children.
void	onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height.
void	onOverScrolled (int scrollX, int scrollY, boolean clampedX, boolean clampedY) Called by `overScrollBy(int, int, int, int, int, int, int, int, boolean)` to respond to the results of an over-scroll operation.
boolean	onRequestFocusInDescendants (int direction, Rect previouslyFocusedRect) When looking for focus in children of a scroll view, need to be a little more careful not to give focus to something that is scrolled off screen.
void	onRestoreInstanceState (Parcelable state) Hook allowing a view to re-apply a representation of its internal state that had previously been generated by `onSaveInstanceState()`.
Parcelable	onSaveInstanceState () Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state.
void	onSizeChanged (int w, int h, int oldw, int oldh) This is called during layout when the size of this view has changed.

Inherited Methods [Expand]

- From class android.widget.FrameLayout
- From class android.view.ViewGroup
- 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.ViewManager
- From interface android.view.ViewParent
- From interface android.view.accessibility.AccessibilityEventSource

XML Attributes

android:fillViewport

Defines whether the scrollview should stretch its content to fill the viewport.

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 `fillViewport` (/reference/android/R.attr.html#fillViewport).

Related Methods

setFillViewport(boolean)

Public Constructors

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

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

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

Public Methods

public void **addView** (View child) Added in API level 1

Adds a child view. If no layout parameters are already set on the child, the default parameters for this ViewGroup are set on the child.

Note: do not invoke this method from `draw(android.graphics.Canvas)` (/reference/android/view/View.html#draw(android.graphics.Canvas)), `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)), `dispatchDraw(android.graphics.Canvas)` (/reference/android/view/ViewGroup.html#dispatchDraw(android.graphics.Canvas)) or any related method.

Parameters

child	the child view to add

public void **addView** (View child, int index) Added in API level 1

Adds a child view. If no layout parameters are already set on the child, the default parameters for this ViewGroup are set on the child.

Note: do not invoke this method from `draw(android.graphics.Canvas)` (/reference/android/view/View.html#draw(android.graphics.Canvas)), `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)), `dispatchDraw(android.graphics.Canvas)` (/reference/android/view/ViewGroup.html#dispatchDraw(android.graphics.Canvas)) or any related method.

Parameters

child	the child view to add
index	the position at which to add the child

public void **addView** (View child, int index, ViewGroup.LayoutParams params) Added in API level 1

Adds a child view with the specified layout parameters.

Note: do not invoke this method from `draw(android.graphics.Canvas)` (/reference/android/view/View.html#draw(android.graphics.Canvas)), `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)), `dispatchDraw(android.graphics.Canvas)` (/reference/android/view/ViewGroup.html#dispatchDraw(android.graphics.Canvas)) or any related method.

Parameters

child	the child view to add
index	the position at which to add the child
params	the layout parameters to set on the child

public void **addView** (View child, ViewGroup.LayoutParams params) Added in API level 1

Adds a child view with the specified layout parameters.

Note: do not invoke this method from `draw(android.graphics.Canvas)` (/reference/android/view/View.html#draw(android.graphics.Canvas)), `onDraw(android.graphics.Canvas)` (/reference/android/view/View.html#onDraw(android.graphics.Canvas)), `dispatchDraw(android.graphics.Canvas)` (/reference/android/view/ViewGroup.html#dispatchDraw(android.graphics.Canvas)) or any related method.

Parameters

child	the child view to add
params	the layout parameters to set on the child

public boolean **arrowScroll** (int direction) Added in API level 1

Handle scrolling in response to an up or down arrow click.

Parameters

direction	The direction corresponding to the arrow key that was pressed

Returns

True if we consumed the event, false otherwise

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 boolean **dispatchKeyEvent** (KeyEvent event)

Added in API level 1

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

Parameters

event	The key event to be dispatched.

Returns

True if the event was handled, false otherwise.

public void **draw** (Canvas canvas)

Added in API level 1

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

Parameters

canvas	The Canvas to which the View is rendered.

public boolean **executeKeyEvent** (KeyEvent event)

Added in API level 1

You can call this function yourself to have the scroll view perform scrolling from a key event, just as if the event had been dispatched to it by the view hierarchy.

Parameters

event	The key event to execute.

Returns

Return true if the event was handled, else false.

public void **fling** (int velocityY)

Added in API level 1

Fling the scroll view

Parameters

velocityY	The initial velocity in the Y direction. Positive numbers mean that the finger/cursor is moving down the screen, which means we want to scroll towards the top.

public boolean **fullScroll** (int direction)

Added in API level 1

Handles scrolling in response to a "home/end" shortcut press. This method will scroll the view to the top or bottom and give the focus to the topmost/bottommost component in the new visible area. If no component is a good candidate for focus, this scrollview reclaims the focus.

Parameters

direction	the scroll direction: `FOCUS_UP` to go the top of the view or `FOCUS_DOWN` to go the bottom

Returns

true if the key event is consumed by this method, false otherwise

public int **getMaxScrollAmount** ()

Added in API level 1

Returns

The maximum amount this scroll view will scroll in response to an arrow event.

public boolean **isFillViewport** ()

Added in API level 1

Indicates whether this ScrollView's content is stretched to fill the viewport.

Related XML Attributes

android:fillViewport

Returns

True if the content fills the viewport, false otherwise.

public boolean **isSmoothScrollingEnabled** ()

Added in API level 1

Returns

Whether arrow scrolling will animate its transition.

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 event) {
    if (event.isFromSource(InputDevice.SOURCE_CLASS_JOYSTICK)) {
        if (event.getAction() == MotionEvent.ACTION_MOVE) {
            // process the joystick movement...
            return true;
        }
    }
    if (event.isFromSource(InputDevice.SOURCE_CLASS_POINTER)) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_HOVER_MOVE:
                // process the mouse hover movement...
                return true;
            case MotionEvent.ACTION_SCROLL:
                // process the scroll wheel movement...
                return true;
        }
    }
    return super.onGenericMotionEvent(event);
}
```

Parameters

event	The generic motion event being processed.

Returns

True if the event was handled, false otherwise.

public void **onInitializeAccessibilityEvent** (AccessibilityEvent event)

Added in API level 14

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

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

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

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

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

Parameters

event	The event to initialize.

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 **onInterceptTouchEvent** (MotionEvent ev) Added in API level 1

Implement this method to intercept all touch screen motion events. This allows you to watch events as they are dispatched to your children, and take ownership of the current gesture at any point.

Using this function takes some care, as it has a fairly complicated interaction with `View.onTouchEvent(MotionEvent)` (/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)), and using it requires implementing that method as well as this one in the correct way. Events will be received in the following order:

1. You will receive the down event here.
2. The down event will be handled either by a child of this view group, or given to your own onTouchEvent() method to handle; this means you should implement onTouchEvent() to return true, so you will continue to see the rest of the gesture (instead of looking for a parent view to handle it). Also, by returning true from onTouchEvent(), you will not receive any following events in onInterceptTouchEvent() and all touch processing must happen in onTouchEvent() like normal.
3. For as long as you return false from this function, each following event (up to and including the final up) will be delivered first here and then to the target's onTouchEvent().
4. If you return true from here, you will not receive any following events: the target view will receive the same event but with the action `ACTION_CANCEL`, and all further events will be delivered to your onTouchEvent() method and no longer appear here.

Parameters

ev	The motion event being dispatched down the hierarchy.

Returns

Return true to steal motion events from the children and have them dispatched to this ViewGroup through onTouchEvent(). The current target will receive an ACTION_CANCEL event, and no further messages will be delivered here.

public boolean **onTouchEvent** (MotionEvent ev)

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

ev	The motion event.

Returns

True if the event was handled, false otherwise.

public boolean **pageScroll** (int direction)

Added in API level 1

Handles scrolling in response to a "page up/down" shortcut press. This method will scroll the view by one page up or down and give the focus to the topmost/bottommost component in the new visible area. If no component is a good candidate for focus, this scrollview reclaims the focus.

Parameters

direction	the scroll direction: `FOCUS_UP` to go one page up or `FOCUS_DOWN` to go one page down

Returns

true if the key event is consumed by this method, 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 void **requestChildFocus** (View child, View focused)

Added in API level 1

Called when a child of this parent wants focus

Parameters

child	The child of this ViewParent that wants focus. This view will contain the focused view. It is not necessarily the view that actually has focus.
focused	The view that is a descendant of child that actually has focus

public boolean **requestChildRectangleOnScreen** (View child, Rect rectangle, boolean immediate)

Added in API level 1

Called when a child of this group wants a particular rectangle to be positioned onto the screen. `ViewGroup` (/reference/android/view/ViewGroup.html)s overriding this can trust that:

- child will be a direct child of this group
- rectangle will be in the child's coordinates

`ViewGroup` (/reference/android/view/ViewGroup.html)s overriding this should uphold the contract:

- nothing will change if the rectangle is already visible
- the view port will be scrolled only just enough to make the rectangle visible

Parameters

child	The direct child making the request.
rectangle	The rectangle in the child's coordinates the child wishes to be on the screen.
immediate	True to forbid animated or delayed scrolling, false otherwise

Returns

Whether the group scrolled to handle the operation

public void **requestDisallowInterceptTouchEvent** (boolean disallowIntercept)

Added in API level 1

Called when a child does not want this parent and its ancestors to intercept touch events with `onInterceptTouchEvent(MotionEvent)` (/reference/android/view/ViewGroup.html#onInterceptTouchEvent(android.view.MotionEvent)).

This parent should pass this call onto its parents. This parent must obey this request for the duration of the touch (that is, only clear the flag after this parent has received an up or a cancel.

Parameters

disallowIntercept	True if the child does not want the parent to intercept touch events.

public void **requestLayout** ()

Added in API level 1

Call this when something has changed which has invalidated the layout of this view. This will schedule a layout pass of the view tree. This should not be called while the view hierarchy is currently in a layout pass (`isInLayout()` (/reference/android/view/View.html#isInLayout()). If layout is happening, the request may be honored at the end of the current layout pass (and then layout will run again) or after the current frame is drawn and the next layout occurs.

Subclasses which override this method should call the superclass method to handle possible request-during-layout errors correctly.

public void **scrollTo** (int x, int y)

Added in API level 1

Set the scrolled position of your view. This will cause a call to `onScrollChanged(int, int, int, int)` (/reference/android/view/View.html#onScrollChanged(int, int, int, int)) and the view will be invalidated.

This version also clamps the scrolling to the bounds of our child.

Parameters

x	the x position to scroll to
y	the y position to scroll to

public void **setFillViewport** (boolean fillViewport)

Added in API level 1

Indicates this ScrollView whether it should stretch its content height to fill the viewport or not.

Related XML Attributes

android:fillViewport

Parameters

fillViewport	True to stretch the content's height to the viewport's boundaries, false otherwise.

public void **setOverScrollMode** (int mode)

Added in API level 9

Set the over-scroll mode for this view. Valid over-scroll modes are `OVER_SCROLL_ALWAYS` (/reference/android/view/View.html#OVER_SCROLL_ALWAYS) (default), `OVER_SCROLL_IF_CONTENT_SCROLLS` (/reference/android/view/View.html#OVER_SCROLL_IF_CONTENT_SCROLLS) (allow over-scrolling only if the view content is larger than the container), or `OVER_SCROLL_NEVER` (/reference/android/view/View.html#OVER_SCROLL_NEVER). Setting the over-scroll mode of a view will have an effect only if the view is capable of scrolling.

Parameters

mode	The new over-scroll mode for this view.

public void **setSmoothScrollingEnabled** (boolean smoothScrollingEnabled)

Added in API level 1

Set whether arrow scrolling will animate its transition.

Parameters

smoothScrollingEnabled	whether arrow scrolling will animate its transition

public boolean **shouldDelayChildPressedState** ()

Added in API level 14

Return true if the pressed state should be delayed for children or descendants of this ViewGroup. Generally, this should be done for containers that can scroll, such as a List. This prevents the pressed state from appearing when the user is actually trying to scroll the content. The default implementation returns true for compatibility reasons. Subclasses that do not scroll should generally override this method and return false.

public final void **smoothScrollBy** (int dx, int dy)

Added in API level 1

Like `scrollBy(int, int)` (/reference/android/view/View.html#scrollBy(int, int)), but scroll smoothly instead of immediately.

Parameters

dx	the number of pixels to scroll by on the X axis
dy	the number of pixels to scroll by on the Y axis

public final void **smoothScrollTo** (int x, int y)

Added in API level 1

Like `scrollTo(int, int)` (/reference/android/widget/ScrollView.html#scrollTo(int, int)), but scroll smoothly instead of immediately.

Parameters

x	the position where to scroll on the X axis
y	the position where to scroll on the Y axis

Protected Methods

protected int **computeScrollDeltaToGetChildRectOnScreen** (Rect rect)

Added in API level 1

Compute the amount to scroll in the Y direction in order to get a rectangle completely on the screen (or, if taller than the screen, at least the first screen size chunk of it).

Parameters

rect	The rect.

Returns

The scroll delta.

protected int **computeVerticalScrollOffset** ()

Added in API level 1

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

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

The default offset is the scroll offset of this view.

Returns

the vertical offset of the scrollbar's thumb

protected int **computeVerticalScrollRange** ()

Added in API level 1

The scroll range of a scroll view is the overall height of all of its children.

Returns

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

protected float **getBottomFadingEdgeStrength** ()

Added in API level 1

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

Returns

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

protected float **getTopFadingEdgeStrength** ()

Added in API level 1

Returns the strength, or intensity, of the top faded edge. The strength is a value between 0.0 (no fade) and 1.0

(full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

Returns

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

protected void **measureChild** (View child, int parentWidthMeasureSpec, int parentHeightMeasureSpec)

Added in API level 1

Ask one of the children of this view to measure itself, taking into account both the MeasureSpec requirements for this view and its padding. The heavy lifting is done in getChildMeasureSpec.

Parameters

child	The child to measure
parentWidthMeasureSpec	The width requirements for this view
parentHeightMeasureSpec	The height requirements for this view

protected void **measureChildWithMargins** (View child, int parentWidthMeasureSpec, int widthUsed, int parentHeightMeasureSpec, int heightUsed)

Added in API level 1

Ask one of the children of this view to measure itself, taking into account both the MeasureSpec requirements for this view and its padding and margins. The child must have MarginLayoutParams The heavy lifting is done in getChildMeasureSpec.

Parameters

child	The child to measure
parentWidthMeasureSpec	The width requirements for this view
widthUsed	Extra space that has been used up by the parent horizontally (possibly by other children of the parent)
parentHeightMeasureSpec	The height requirements for this view
heightUsed	Extra space that has been used up by the parent vertically (possibly by other children of the parent)

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 **onLayout** (boolean changed, int l, int t, int r, int b)

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
l	Left position, relative to parent
t	Top position, relative to parent
r	Right position, relative to parent
b	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 **onOverScrolled** (int scrollX, int scrollY, boolean clampedX, boolean clampedY)

Added in API level 9

Called by `overScrollBy(int, int, int, int, int, int, int, int, boolean)` (/reference/android/view/View.html#overScrollBy(int, int, int, int, int, int, int, int, boolean)) to respond to the results of an over-scroll operation.

Parameters

scrollX	New X scroll value in pixels
scrollY	New Y scroll value in pixels
clampedX	True if scrollX was clamped to an over-scroll boundary
clampedY	True if scrollY was clamped to an over-scroll boundary

protected boolean **onRequestFocusInDescendants** (int direction, Rect previouslyFocusedRect)

Added in API level 1

When looking for focus in children of a scroll view, need to be a little more careful not to give focus to something that is scrolled off screen. This is more expensive than the default `ViewGroup` (/reference/android/view/ViewGroup.html) implementation, otherwise this behavior might have been made the default.

Parameters

direction	One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT
previouslyFocusedRect	The rectangle (in this View's coordinate system) to give a finer grained hint about where focus is coming from. May be null if there is no hint.

Returns

Whether focus was taken.

protected void **onRestoreInstanceState** (Parcelable state)

Added in API level 1

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

Parameters

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

protected Parcelable **onSaveInstanceState** ()

Added in API level 1

Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state. This state should only contain information that is not persistent or can not be reconstructed later. For example, you will never store your current position on screen because that will be computed again when a new instance of the view is placed in its view hierarchy.

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

Returns

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

protected void **onSizeChanged** (int w, int h, int oldw, int oldh)

Added in API level 1

This is called during layout when the size of this view has changed. If you were just added to the view hierarchy, you're called with the old values of 0.

Parameters

w	Current width of this view.
h	Current height of this view.
oldw	Old width of this view.

oldh	Old height of this view.