

public class

ScrollView

extends [FrameLayout](#)

```

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](#), 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](#) in a vertical orientation, presenting a vertical array of top-level items that the user can scroll through.

The [TextView](#) 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.

Summary

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	
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	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.
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	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	<code>setSmoothScrollingEnabled</code> (boolean <code>smoothScrollingEnabled</code>) Set whether arrow scrolling will animate its transition.
final void	<code>smoothScrollBy</code> (int dx, int dy) Like <code>scrollBy(int, int)</code> , but scroll smoothly instead of immediately.
final void	<code>smoothScrollTo</code> (int x, int y) Like <code>scrollTo(int, int)</code> , but scroll smoothly instead of immediately.

Protected Methods	
int	<code>computeScrollDeltaToGetChildRectOnScreen</code> (<code>Rect</code> 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	<code>computeVerticalScrollOffset</code> () Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range.
int	<code>computeVerticalScrollRange</code> () The scroll range of a scroll view is the overall height of all of its children.
float	<code>getBottomFadingEdgeStrength</code> () Returns the strength, or intensity, of the bottom faded edge.
float	<code>getTopFadingEdgeStrength</code> () Returns the strength, or intensity, of the top faded edge.
void	<code>measureChild</code> (<code>View</code> child, int <code>parentWidthMeasureSpec</code> , int <code>parentHeightMeasureSpec</code>) Ask one of the children of this view to measure itself, taking into account both the <code>MeasureSpec</code> requirements for this view and its padding.
void	<code>measureChildWithMargins</code> (<code>View</code> child, int <code>parentWidthMeasureSpec</code> , int <code>widthUsed</code> , int <code>parentHeightMeasureSpec</code> , int <code>heightUsed</code>) Ask one of the children of this view to measure itself, taking into account both the <code>MeasureSpec</code> requirements for this view and its padding and margins.
void	<code>onLayout</code> (boolean <code>changed</code> , 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	<code>onMeasure</code> (int <code>widthMeasureSpec</code> , int <code>heightMeasureSpec</code>) Measure the view and its content to determine the measured width and the measured height.
void	<code>onOverScrolled</code> (int <code>scrollX</code> , int <code>scrollY</code> , boolean <code>clampedX</code> , boolean <code>clampedY</code>) Called by <code>overScrollBy(int, int, int, int, int, int, int, int, int, boolean)</code> to respond to the results of an over-scroll operation.
boolean	<code>onRequestFocusInDescendants</code> (int <code>direction</code> , <code>Rect</code> <code>previouslyFocusedRect</code>) 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	<code>onSizeChanged</code> (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

Public Constructors

public ScrollView (Context context)	Since: API Level 1
public ScrollView (Context context, AttributeSet attrs)	Since: API Level 1
public ScrollView (Context context, AttributeSet attrs, int defStyle)	Since: API Level 1

Public Methods

public void addView (View child)	Since: 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.	
Parameters	
<i>child</i>	the child view to add
public void addView (View child, int index)	Since: 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.	
Parameters	
<i>child</i>	the child view to add
<i>index</i>	the position at which to add the child
public void addView (View child, int index, ViewGroup.LayoutParams params)	Since: API Level 1
Adds a child view with the specified layout parameters.	
Parameters	
<i>child</i>	the child view to add
<i>index</i>	the position at which to add the child
<i>params</i>	the layout parameters to set on the child
public void addView (View child, ViewGroup.LayoutParams params)	Since: API Level 1
Adds a child view with the specified layout parameters.	

Parameters

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

public boolean **arrowScroll** (int direction)

Since: 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** ()

Since: 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](#) object.

public boolean **dispatchKeyEvent** ([KeyEvent](#) event)

Since: 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)

Since: 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, do not override this method; instead, you should implement [onDraw\(Canvas\)](#).

Parameters

canvas The Canvas to which the View is rendered.

public boolean **executeKeyEvent** ([KeyEvent](#) event)

Since: 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)

Since: 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)

Since: 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** ()

Since: API Level 1

Returns

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

public boolean **isFillViewport** ()

Since: API Level 1

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

Returns

True if the content fills the viewport, false otherwise.

public boolean **isSmoothScrollingEnabled** ()

Since: API Level 1

Returns

Whether arrow scrolling will animate its transition.

public boolean **onInterceptTouchEvent** ([MotionEvent](#) ev)

Since: 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\(MotionEvent\)](#), 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)

Since: API Level 1

Implement this method to handle touch screen motion events.

Parameters

ev The motion event.

Returns

True if the event was handled, false otherwise.

public boolean **pageScroll** (int direction)

Since: 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 void **requestChildFocus** ([View](#) child, [View](#) focused)

Since: 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)

Since: API Level 1

Called when a child of this group wants a particular rectangle to be positioned onto the screen. [ViewGroups](#) overriding this can trust that:

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

[ViewGroups](#) 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 **requestLayout** ()

Since: 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.

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

Since: API Level 1

Set the scrolled position of your view. This will cause a call to [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)

Since: API Level 1

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

Parameters

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

public void **setOverScrollMode** (int mode)

Since: API Level 9

Set the over-scroll mode for this view. Valid over-scroll modes are [OVER_SCROLL_ALWAYS](#) (default), [OVER_SCROLL_IF_CONTENT_SCROLLS](#) (allow over-scrolling only if the view content is larger than the container), or [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)

Since: API Level 1

Set whether arrow scrolling will animate its transition.

Parameters

smoothScrollingEnabled whether arrow scrolling will animate its transition

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

Since: API Level 1

Like [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)

Since: API Level 1

Like [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)

Since: 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** ()

Since: 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\(\)](#) and [computeVerticalScrollExtent\(\)](#).

The default offset is the scroll offset of this view.

Returns

the vertical offset of the scrollbar's thumb

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.

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

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

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

<i>child</i>	The child to measure
<i>parentWidthMeasureSpec</i>	The width requirements for this view
<i>parentHeightMeasureSpec</i>	The height requirements for this view

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

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

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

<i>changed</i>	This is a new size or position for this view
<i>l</i>	Left position, relative to parent
<i>t</i>	Top position, relative to parent
<i>r</i>	Right position, relative to parent
<i>b</i>	Bottom position, relative to parent

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

Since: API Level 1

Measure the view and its content to determine the measured width and the measured height. This method is invoked by [measure\(int, int\)](#) and should be overridden by subclasses to provide accurate and efficient measurement of their contents.

CONTRACT: When overriding this method, you *must* call [setMeasuredDimension\(int, int\)](#) to store the measured width and height of this view. Failure to do so will trigger an [IllegalStateException](#), thrown by [measure\(int, int\)](#). Calling the superclass' [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\(int, int\)](#) 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\(\)](#) and [getSuggestedMinimumWidth\(\)](#)).

Parameters

<i>widthMeasureSpec</i>	horizontal space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec .
<i>heightMeasureSpec</i>	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)

Since: API Level 9

Called by [overScrollBy\(int, int, int, int, int, int, int, int, boolean\)](#) to respond to the results of an over-scroll operation.

Parameters

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

protected boolean **onRequestFocusInDescendants** (int direction, [Rect](#) previouslyFocusedRect)

Since: 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](#) implementation, otherwise this behavior might have been made the default.

Parameters

<i>direction</i>	One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT
<i>previouslyFocusedRect</i>	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 **onSizeChanged** (int w, int h, int oldw, int oldh)

Since: 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

<i>w</i>	Current width of this view.
<i>h</i>	Current height of this view.
<i>oldw</i>	Old width of this view.
<i>oldh</i>	Old height of this view.

Except as noted, this content is licensed under [Apache 2.0](#). For details and restrictions, see the [Content License](#).
Android 2.3 r1 - 14 Feb 2011 9:37

[Site Terms of Service](#) - [Privacy Policy](#) - [Brand Guidelines](#)