public class
ImageView
extends View

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

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

▶ Known Direct Subclasses
ImageButton, QuickContactBadge

▶ Known Indirect Subclasses
ZoomButton

Class Overview

Displays an arbitrary image, such as an icon. The ImageView class can load images from various sources (such as resources or content providers), takes care of computing its measurement from the image so that it can be used in any layout manager, and provides various display options such as scaling and tinting.

Summary

Nested Classes

enum ImageView.ScaleType	Options for scaling the bounds of an image to the bounds of this view.

XML Attributes

Attribute Name	*Related Method*	*Description*
android:adjustViewBounds	setAdjustViewBounds(boolean)	Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
android:baseline	setBaseline(int)	The offset of the baseline within this view.
android:baselineAlignBottom	setBaselineAlignBottom(boolean)	If true, the image view will be baseline

		aligned with based on its bottom edge.
android:cropToPadding	setCropToPadding(boolean)	If true, the image will be cropped to fit within its padding.
android:maxHeight	setMaxHeight(int)	An optional argument to supply a maximum height for this view.
android:maxWidth	setMaxWidth(int)	An optional argument to supply a maximum width for this view.
android:scaleType	setScaleType(ImageView.ScaleType)	Controls how the image should be resized or moved to match the size of this ImageView.
android:src	setImageResource(int)	Sets a drawable as the content of this ImageView.
android:tint	setColorFilter(int,PorterDuff.Mode)	Set a tinting color for the image.

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

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

Public Methods

final void	clearColorFilter ()
boolean	getAdjustViewBounds () True when ImageView is adjusting its bounds to preserve the aspect ratio of its drawable
int	getBaseline () Return the offset of the widget's text baseline from the widget's top boundary.
boolean	getBaselineAlignBottom () Return whether this view's baseline will be considered the bottom of the view.
ColorFilter	getColorFilter () Returns the active color filter for this ImageView.
boolean	getCropToPadding () Return whether this ImageView crops to padding.
Drawable	getDrawable () Return the view's drawable, or null if no drawable has been assigned.
int	getImageAlpha () Returns the alpha that will be applied to the drawable of this ImageView.
Matrix	getImageMatrix () Return the view's optional matrix.
int	getMaxHeight () The maximum height of this view.
int	getMaxWidth () The maximum width of this view.
ImageView.ScaleType	getScaleType () Return the current scale type in use by this ImageView.
boolean	hasOverlappingRendering () Returns whether this View has content which overlaps.
void	invalidateDrawable (Drawable dr) Invalidates the specified Drawable.
void	jumpDrawablesToCurrentState () Call `Drawable.jumpToCurrentState()` on all Drawable objects associated with this view.
int[]	onCreateDrawableState (int extraSpace) Generate the new `Drawable` state for this view.
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.
void	onPopulateAccessibilityEvent (AccessibilityEvent event) Called from `dispatchPopulateAccessibilityEvent(AccessibilityEvent)` giving a chance to this View to populate the accessibility event with its text content.
void	onRtlPropertiesChanged (int layoutDirection) Called when any RTL property (layout direction or text direction or text alignment) has been changed.
void	setAdjustViewBounds (boolean adjustViewBounds) Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
void	setAlpha (int alpha) *This method was deprecated in API level 16. use #setImageAlpha(int) instead*
void	setBaseline (int baseline) Set the offset of the widget's text baseline from the widget's top boundary.
void	setBaselineAlignBottom (boolean aligned) Set whether to set the baseline of this view to the bottom of the view.
final void	setColorFilter (int color) Set a tinting option for the image.
void	setColorFilter (ColorFilter cf) Apply an arbitrary colorfilter to the image.
final void	setColorFilter (int color, PorterDuff.Mode mode) Set a tinting option for the image.
void	setCropToPadding (boolean cropToPadding) Sets whether this ImageView will crop to padding.
void	setImageAlpha (int alpha) Sets the alpha value that should be applied to the image.
void	setImageBitmap (Bitmap bm) Sets a Bitmap as the content of this ImageView.
void	setImageDrawable (Drawable drawable) Sets a drawable as the content of this ImageView.
void	setImageLevel (int level) Sets the image level, when it is constructed from a `LevelListDrawable`.
void	setImageMatrix (Matrix matrix)
void	setImageResource (int resId) Sets a drawable as the content of this ImageView.
void	setImageState (int[] state, boolean merge)

void	setImageURI (Uri uri) Sets the content of this ImageView to the specified Uri.
void	setMaxHeight (int maxHeight) An optional argument to supply a maximum height for this view.
void	setMaxWidth (int maxWidth) An optional argument to supply a maximum width for this view.
void	setScaleType (ImageView.ScaleType scaleType) Controls how the image should be resized or moved to match the size of this ImageView.
void	setSelected (boolean selected) Changes the selection state of this view.
void	setVisibility (int visibility) Set the enabled state of this view.

Protected Methods

void	drawableStateChanged () This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown.
void	onAttachedToWindow () This is called when the view is attached to a window.
void	onDetachedFromWindow () This is called when the view is detached from a window.
void	onDraw (Canvas canvas) Implement this to do your drawing.
void	onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height.
boolean	setFrame (int l, int t, int r, int b) Assign a size and position to this view.
boolean	verifyDrawable (Drawable dr) 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.accessibility.AccessibilityEventSource

XML Attributes

android:adjustViewBounds

Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.

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

Related Methods

setAdjustViewBounds(boolean)

android:baseline

The offset of the baseline within this view. See {see android.view.View#getBaseline} for details

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

Related Methods

setBaseline(int)

android:baselineAlignBottom

If true, the image view will be baseline aligned with based on its bottom edge.

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

Related Methods

setBaselineAlignBottom(boolean)

android:cropToPadding

If true, the image will be cropped to fit within its padding.

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

Related Methods
setCropToPadding(boolean)

android:maxHeight

An optional argument to supply a maximum height for this view. See {see android.widget.ImageView#setMaxHeight} for details.

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:maxWidth

An optional argument to supply a maximum width for this view. See {see android.widget.ImageView#setMaxWidth} for details.

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:scaleType

Controls how the image should be resized or moved to match the size of this ImageView.

Must be one of the following constant values.

Constant	Value Description
`matrix`	0
`fitXY`	1
`fitStart`	2
`fitCenter`	3
`fitEnd`	4
`center`	5
`centerCrop`	6
`centerInside`	7

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

Related Methods
setScaleType(ImageView.ScaleType)

android:src

Sets a drawable as the content of this ImageView.

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

Related Methods
setImageResource(int)

android:tint

Set a tinting color for the image.

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

Related Methods

setColorFilter(int,PorterDuff.Mode)

Public Constructors

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

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

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

Public Methods

public final void **clearColorFilter** () Added in API level 1

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

True when ImageView is adjusting its bounds to preserve the aspect ratio of its drawable

Related XML Attributes

android:adjustViewBounds

Returns

whether to adjust the bounds of this view to presrve the original aspect ratio of the drawable

See Also

`setAdjustViewBounds(boolean)`

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

Return the offset of the widget's text baseline from the widget's top boundary.

Returns

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

public boolean **getBaselineAlignBottom** () Added in API level 11

Return whether this view's baseline will be considered the bottom of the view.

See Also

`setBaselineAlignBottom(boolean)`

public ColorFilter **getColorFilter** ()

Added in API level 16

Returns the active color filter for this ImageView.

Returns

the active color filter for this ImageView

See Also

`setColorFilter(android.graphics.ColorFilter)`

public boolean **getCropToPadding** ()

Added in API level 16

Return whether this ImageView crops to padding.

Related XML Attributes

android:cropToPadding

Returns

whether this ImageView crops to padding

See Also

`setCropToPadding(boolean)`

public Drawable **getDrawable** ()

Added in API level 1

Return the view's drawable, or null if no drawable has been assigned.

public int **getImageAlpha** ()

Added in API level 16

Returns the alpha that will be applied to the drawable of this ImageView.

Returns

the alpha that will be applied to the drawable of this ImageView

See Also

`setImageAlpha(int)`

public Matrix **getImageMatrix** ()

Added in API level 1

Return the view's optional matrix. This is applied to the view's drawable when it is drawn. If there is not matrix, this method will return an identity matrix. Do not change this matrix in place but make a copy. If you want a different matrix applied to the drawable, be sure to call setImageMatrix().

public int **getMaxHeight** ()

Added in API level 16

The maximum height of this view.

Related XML Attributes

android:maxHeight

Returns

The maximum height of this view

See Also

`setMaxHeight(int)`

public int **getMaxWidth** ()

Added in API level 16

The maximum width of this view.

Related XML Attributes

android:maxWidth

Returns

The maximum width of this view

See Also

`setMaxWidth(int)`

public ImageView.ScaleType **getScaleType** ()

Added in API level 1

Return the current scale type in use by this ImageView.

Related XML Attributes

android:scaleType

See Also

`ImageView.ScaleType`

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 void **invalidateDrawable** (Drawable dr)

Added in API level 1

Invalidates the specified Drawable.

Parameters

dr	the drawable to invalidate

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[] **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.

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(Accessibility
    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 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(AccessibilityEv
    super.onPopulateAccessibilityEvent(event);
    final int flags = DateUtils.FORMAT_SHOW_DATE | DateU
    String selectedDateUtterance = DateUtils.formatDateT
        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 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 void **setAdjustViewBounds** (boolean adjustViewBounds) Added in API level 1

Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.

Note: If the application targets API level 17 or lower, adjustViewBounds will allow the drawable to shrink the view bounds, but not grow to fill available measured space in all cases. This is for compatibility with legacy `MeasureSpec` (/reference/android/view/View.MeasureSpec.html) and `RelativeLayout` (/reference/android/widget/RelativeLayout.html) behavior.

Related XML Attributes

android:adjustViewBounds

Parameters

adjustViewBounds	Whether to adjust the bounds of this view to preserve the original aspect ratio of the drawable.

See Also

`getAdjustViewBounds()`

public void **setAlpha** (int alpha) Added in API level 1

This method was deprecated in API level 16.
use #setImageAlpha(int) instead

Sets the alpha value that should be applied to the image.

Parameters

alpha	the alpha value that should be applied to the image

public void **setBaseline** (int baseline) Added in API level 11

Set the offset of the widget's text baseline from the widget's top boundary. This value is overridden by the `setBaselineAlignBottom(boolean)` (/reference/android/widget/ImageView.html#setBaselineAlignBottom(boolean)) property.

Related XML Attributes

android:baseline

Parameters

baseline	The baseline to use, or -1 if none is to be provided.

See Also

`setBaseline(int)`

public void **setBaselineAlignBottom** (boolean aligned) Added in API level 11

Set whether to set the baseline of this view to the bottom of the view. Setting this value overrides any calls to setBaseline.

Related XML Attributes

android:baselineAlignBottom

Parameters

aligned	If true, the image view will be baseline aligned with based on its bottom edge.

public final void **setColorFilter** (int color) Added in API level 8

Set a tinting option for the image. Assumes `SRC_ATOP (/reference/android/graphics/PorterDuff.Mode.html#SRC_ATOP)` blending mode.

Related XML Attributes

android:tint

Parameters

color	Color tint to apply.

public void **setColorFilter** (ColorFilter cf) Added in API level 1

Apply an arbitrary colorfilter to the image.

Parameters

cf	the colorfilter to apply (may be null)

See Also

`getColorFilter()`

public final void **setColorFilter** (int color, PorterDuff.Mode mode) Added in API level 1

Set a tinting option for the image.

Related XML Attributes

android:tint

Parameters

color	Color tint to apply.
mode	How to apply the color. The standard mode is `SRC_ATOP`

public void **setCropToPadding** (boolean cropToPadding) Added in API level 16

Sets whether this ImageView will crop to padding.

Related XML Attributes
android:cropToPadding

Parameters

cropToPadding	whether this ImageView will crop to padding

See Also
`getCropToPadding()`

public void **setImageAlpha** (int alpha) Added in API level 16

Sets the alpha value that should be applied to the image.

Parameters

alpha	the alpha value that should be applied to the image

See Also
`getImageAlpha()`

public void **setImageBitmap** (Bitmap bm) Added in API level 1

Sets a Bitmap as the content of this ImageView.

Parameters

bm	The bitmap to set

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

Sets a drawable as the content of this ImageView.

Parameters

drawable	The drawable to set

public void **setImageLevel** (int level) Added in API level 1

Sets the image level, when it is constructed from a `LevelListDrawable` `(/reference/android/graphics/drawable/LevelListDrawable.html)`.

Parameters

level	The new level for the image.

public void **setImageMatrix** (Matrix matrix) Added in API level 1

public void **setImageResource** (int resId) Added in API level 1

Sets a drawable as the content of this ImageView.

> This does Bitmap reading and decoding on the UI thread, which can cause a latency hiccup. If that's a concern, consider using `setImageDrawable(android.graphics.drawable.Drawable)` `(/reference/android/widget/ImageView.html#setImageDrawable(android.graphics.drawable.Drawable))` or `setImageBitmap(android.graphics.Bitmap)` `(/reference/android/widget/ImageView.html#setImageBitmap(android.graphics.Bitmap))` and `BitmapFactory` `(/reference/android/graphics/BitmapFactory.html)` instead.

Related XML Attributes

android:src

Parameters

resId	the resource identifier of the drawable

public void **setImageState** (int[] state, boolean merge)

Added in API level 1

public void **setImageURI** (Uri uri)

Added in API level 1

Sets the content of this ImageView to the specified Uri.

> This does Bitmap reading and decoding on the UI thread, which can cause a latency hiccup. If that's a concern, consider using `setImageDrawable(android.graphics.drawable.Drawable)` `(/reference/android/widget/ImageView.html#setImageDrawable(android.graphics.drawable.Drawable))` or `setImageBitmap(android.graphics.Bitmap)` `(/reference/android/widget/ImageView.html#setImageBitmap(android.graphics.Bitmap))` and `BitmapFactory` `(/reference/android/graphics/BitmapFactory.html)` instead.

Parameters

uri	The Uri of an image

public void **setMaxHeight** (int maxHeight)

Added in API level 1

An optional argument to supply a maximum height for this view. Only valid if `setAdjustViewBounds(boolean)` `(/reference/android/widget/ImageView.html#setAdjustViewBounds(boolean))` has been set to true. To set an image to be a maximum of 100 x 100 while preserving the original aspect ratio, do the following: 1) set adjustViewBounds to true 2) set maxWidth and maxHeight to 100 3) set the height and width layout params to WRAP_CONTENT.

Note that this view could be still smaller than 100 x 100 using this approach if the original image is small. To set an image to a fixed size, specify that size in the layout params and then use `setScaleType(android.widget.ImageView.ScaleType)` `(/reference/android/widget/ImageView.html#setScaleType(android.widget.ImageView.ScaleType))` to determine how to fit the image within the bounds.

Related XML Attributes

android:maxHeight

Parameters

maxHeight	maximum height for this view

See Also

`getMaxHeight()`

public void **setMaxWidth** (int maxWidth)

Added in API level 1

An optional argument to supply a maximum width for this view. Only valid if `setAdjustViewBounds(boolean) (/reference/android/widget/ImageView.html#setAdjustViewBounds(boolean))` has been set to true. To set an image to be a maximum of 100 x 100 while preserving the original aspect ratio, do the following: 1) set adjustViewBounds to true 2) set maxWidth and maxHeight to 100 3) set the height and width layout params to WRAP_CONTENT.

Note that this view could be still smaller than 100 x 100 using this approach if the original image is small. To set an image to a fixed size, specify that size in the layout params and then use `setScaleType(android.widget.ImageView.ScaleType) (/reference/android/widget/ImageView.html#setScaleType(android.widget.ImageView.ScaleType))` to determine how to fit the image within the bounds.

Related XML Attributes

android:maxWidth

Parameters

maxWidth	maximum width for this view

See Also

`getMaxWidth()`

public void **setScaleType** (ImageView.ScaleType scaleType)

Added in API level 1

Controls how the image should be resized or moved to match the size of this ImageView.

Related XML Attributes

android:scaleType

Parameters

scaleType	The desired scaling mode.

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 **setVisibility** (int visibility) Added in API level 1

Set the enabled state of this view.

Parameters

visibility	One of `VISIBLE`, `INVISIBLE`, or `GONE`.

Protected Methods

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 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 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 **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 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 dr)

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

dr	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.