NSView Class Reference

Cocoa > Graphics & Imaging

2009-02-04

Contents

NSView Class Reference

Inherits from	NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSClipView.h NSMenuItem.h NSView.h
Companion guides	View Programming Guide for Cocoa Cocoa Drawing Guide Cocoa Event-Handling Guide Drag and Drop Programming Topics for Cocoa Printing Programming Topics for Cocoa
Related sample code	iSpend QTKitMovieShuffler Quartz Composer WWDC 2005 TextEdit Sketch-112 TextEditPlus

Class at a Glance

`NSView` is a class that defines the basic drawing, event-handling, and printing architecture of an application. You typically don't interact with the `NSView` API directly; rather, your custom view classes inherit from `NSView` and override many of its methods, which are invoked automatically by the Application Kit. If you're not creating a custom view class, there are few methods you need to use.

Principal Attributes

- Event handling
- Integrated display to screen and printer
- Flexible coordinate systems
- Icon dragging

Commonly Used Methods

`frame` (page 59)
: Returns the location and size of the `NSView` object.

`bounds` (page 36)
: Returns the internal origin and size of the `NSView` object.

`setNeedsDisplay:` (page 103)
: Marks the `NSView` object as needing to be redrawn

`window` (page 122)
: Returns the `NSWindow` object that contains the `NSView` object.

`drawRect:` (page 55)
: Draws the `NSView` object. (All subclasses must implement this method, but it's rarely invoked explicitly.)

Overview

The `NSView` class that provides a structure for drawing, printing, and handling events.

`NSView` objects (also know, simply, as view objects or views) are arranged within an `NSWindow` object, in a nested hierarchy of subviews. A view object claims a rectangular region of its enclosing superview, is responsible for all drawing within that region, and is eligible to receive mouse events occurring in it as well. In addition to these major responsibilities, `NSView` handles dragging of icons and works with the `NSScrollView` class to support efficient scrolling.

Most of the functionality of `NSView` either is automatically invoked by the Application Kit, or is available in Interface Builder. Unless you're implementing a concrete subclass of `NSView` or working intimately with the content of the view hierarchy at runtime, you don't need to know much about this class's interface. See "Commonly Used Methods" (page 10) for methods you might use regardless.

For more information on how `NSView` instances handle event and action messages, see Handling Events in Views, Handling Mouse Events in Views, and Handling Tracking-Rectangle and Cursor-Update Events in Views. For more information on displaying tooltips and contextual menus, see How Contextual Menus Work and Tooltips.

Subclassing Notes

`NSView` is perhaps the most important class in the Application Kit when it comes to subclassing and inheritance. Most user-interface objects you see in a Cocoa application are objects that inherit from `NSView`. If you want to create an object that draws itself in a special way, or that responds to mouse clicks in a special way, you would create a custom subclass of `NSView` (or of a class that inherits from `NSView`). Subclassing `NSView` is such a common and important procedure that several technical documents describe how to both draw in custom subclasses and respond to events in custom subclasses. See *Cocoa Drawing Guide* and *Cocoa Event-Handling Guide* (especially "Handling Events in Views" and "Handling Mouse Events in Views").

Tasks

Creating Instances

- `initWithFrame:` (page 63)
  Initializes and returns a newly allocated `NSView` object with a specified frame rectangle.

Managing the View Hierarchy

- `superview` (page 111)
  Returns the receiver's superview, or `nil` if it has none.
- `setSubviews:` (page 106)
  Sets the receiver's subviews to the specified subviews.
- `subviews` (page 110)
  Return the receiver's immediate subviews.
- `window` (page 122)
  Returns the receiver's window object, or `nil` if it has none.
- `addSubview:` (page 26)
  Adds a view to the receiver's subviews so it's displayed above its siblings.
- `addSubview:positioned:relativeTo:` (page 27)
  Inserts a view among the receiver's subviews so it's displayed immediately above or below another view.
- `didAddSubview:` (page 47)
  Overridden by subclasses to perform additional actions when subviews are added to the receiver.
- `removeFromSuperview` (page 84)
  Unlinks the receiver from its superview and its window, removes it from the responder chain, and invalidates its cursor rectangles.
- `removeFromSuperviewWithoutNeedingDisplay` (page 84)
  Unlinks the receiver from its superview and its window and removes it from the responder chain, but does not invalidate its cursor rectangles to cause redrawing.
- `replaceSubview:with:` (page 86)
  Replaces one of the receiver's subviews with another view.
- `isDescendantOf:` (page 64)
  Returns `YES` if the receiver is a subview of a given view or if it's identical to that view; otherwise, it returns `NO`.
- `opaqueAncestor` (page 75)
  Returns the receiver's closest opaque ancestor (including the receiver itself).
- `ancestorSharedWithView:` (page 33)
  Returns the closest ancestor shared by the receiver and a given view.
- `sortSubviewsUsingFunction:context:` (page 109)
  Orders the receiver's immediate subviews using the specified comparator function.
- `viewDidMoveToSuperview` (page 116)
  Informs the receiver that its superview has changed (possibly to `nil`).

- `viewDidMoveToWindow` (page 116)
  Informs the receiver that it has been added to a new view hierarchy.
- `viewWillMoveToSuperview:` (page 118)
  Informs the receiver that its superview is about to change to the specified superview (which may be `nil`).
- `viewWillMoveToWindow:` (page 118)
  Informs the receiver that it's being added to the view hierarchy of the specified window object (which may be `nil`).
- `willRemoveSubview:` (page 122)
  Overridden by subclasses to perform additional actions before subviews are removed from the receiver.
- `enclosingMenuItem` (page 57)
  Returns the menu item containing the receiver or any of its superviews in the view hierarchy.

Searching by Tag

- `viewWithTag:` (page 120)
  Returns the receiver's nearest descendant (including itself) with a specific tag, or `nil` if no subview has that tag.
- `tag` (page 111)
  Returns the receiver's tag, an integer that you can use to identify view objects in your application.

Modifying the Frame Rectangle

- `setFrame:` (page 98)
  Sets the receiver's frame rectangle to the specified rectangle.
- `frame` (page 59)
  Returns the receiver's frame rectangle, which defines its position in its superview.
- `setFrameOrigin:` (page 99)
  Sets the origin of the receiver's frame rectangle to the specified point, effectively repositioning it within its superview.
- `setFrameSize:` (page 101)
  Sets the size of the receiver's frame rectangle to the specified dimensions, resizing it within its superview without affecting its coordinate system.
- `setFrameRotation:` (page 100)
  Sets the rotation of the receiver's frame rectangle to a specified degree value, rotating it within its superview without affecting its coordinate system.
- `frameRotation` (page 60)
  Returns the angle, in degrees, of the receiver's frame relative to its superview's coordinate system.

Modifying the Bounds Rectangle

- `setBounds:` (page 94)
  Sets the receiver's bounds rectangle.

- `bounds` (page 36)
  Returns the receiver's bounds rectangle, which expresses its location and size in its own coordinate system.
- `setBoundsOrigin:` (page 94)
  Sets the origin of the receiver's bounds rectangle to a specified point,
- `setBoundsSize:` (page 96)
  Sets the size of the receiver's bounds rectangle to specified dimensions, inversely scaling its coordinate system relative to its frame rectangle.
- `setBoundsRotation:` (page 95)
  Sets the rotation of the receiver's bounds rectangle to a specific degree value.
- `boundsRotation` (page 37)
  Returns the angle, in degrees, of the receiver's bounds rectangle relative to its frame rectangle.

Modifying the Coordinate System

- `translateOriginToPoint:` (page 112)
  Translates the receiver's coordinate system so that its origin moves to a new location.
- `scaleUnitSquareToSize:` (page 89)
  Scales the receiver's coordinate system so that the unit square scales to the specified dimensions.
- `rotateByAngle:` (page 88)
  Rotates the receiver's bounds rectangle by a specified degree value around the origin of the coordinate system, (0.0, 0.0).

Examining Coordinate System Modifications

- `isFlipped` (page 65)
  Returns `YES` if the receiver uses flipped drawing coordinates or `NO` if it uses native coordinates.
- `isRotatedFromBase` (page 67)
  Returns `YES` if the receiver or any of its ancestors has ever received a `setFrameRotation:` (page 100) or `setBoundsRotation:` (page 95) message; otherwise returns `NO`.
- `isRotatedOrScaledFromBase` (page 67)
  Returns `YES` if the receiver or any of its ancestors has ever had a nonzero frame or bounds rotation, or has been scaled from the window's base coordinate system; otherwise returns `NO`.

Base Coordinate Conversion

- `convertPointToBase:` (page 42)
  Converts the point from the receiver's coordinate system to the base coordinate system.
- `convertPointFromBase:` (page 42)
  Converts the point from the base coordinate system to the receiver's coordinate system.
- `convertSizeToBase:` (page 46)
  Converts the size from the receiver's coordinate system to the base coordinate system.
- `convertSizeFromBase:` (page 46)
  Converts the size from the base coordinate system to the receiver's coordinate system.

- `convertRectToBase:` (page 44)
  Converts the rectangle from the receiver's coordinate system to the base coordinate system.
- `convertRectFromBase:` (page 44)
  Converts the rectangle from the base coordinate system to the receiver's coordinate system.

Converting Coordinates

- `convertPoint:fromView:` (page 40)
  Converts a point from the coordinate system of a given view to that of the receiver.
- `convertPoint:toView:` (page 41)
  Converts a point from the receiver's coordinate system to that of a given view.
- `convertSize:fromView:` (page 44)
  Converts a size from another view's coordinate system to that of the receiver.
- `convertSize:toView:` (page 45)
  Converts a size from the receiver's coordinate system to that of another view.
- `convertRect:fromView:` (page 42)
  Converts a rectangle from the coordinate system of another view to that of the receiver.
- `convertRect:toView:` (page 43)
  Converts a rectangle from the receiver's coordinate system to that of another view.
- `centerScanRect:` (page 39)
  Converts the corners of a specified rectangle to lie on the center of device pixels, which is useful in compensating for rendering overscanning when the coordinate system has been scaled.

Controlling Notifications

- `setPostsFrameChangedNotifications:` (page 105)
  Controls whether the receiver informs observers when its frame rectangle changes.
- `postsFrameChangedNotifications` (page 78)
  Returns `YES` if the receiver posts notifications to the default notification center whenever its frame rectangle changes; returns `NO` otherwise.
- `setPostsBoundsChangedNotifications:` (page 104)
  Controls whether the receiver informs observers when its bounds rectangle changes.
- `postsBoundsChangedNotifications` (page 77)
  Returns `YES` if the receiver posts notifications to the default notification center whenever its bounds rectangle changes; returns `NO` otherwise.

Resizing Subviews

- `resizeSubviewsWithOldSize:` (page 87)
  Informs the receiver's subviews that the receiver's bounds rectangle size has changed.
- `resizeWithOldSuperviewSize:` (page 88)
  Informs the receiver that the bounds size of its superview has changed.

- `setAutoresizesSubviews:` (page 92)
  Determines whether the receiver automatically resizes its subviews when its frame size changes.
- `autoresizesSubviews` (page 33)
  Returns `YES` if the receiver automatically resizes its subviews using `resizeSubviewsWithOldSize:` (page 87) whenever its frame size changes, `NO` otherwise.
- `setAutoresizingMask:` (page 93)
  Determines how the receiver's `resizeWithOldSuperviewSize:` (page 88) method changes its frame rectangle.
- `autoresizingMask` (page 34)
  Returns the receiver's autoresizing mask, which determines how it's resized by the `resizeWithOldSuperviewSize:` (page 88) method.

Focusing

- `lockFocus` (page 69)
  Locks the focus on the receiver, so subsequent commands take effect in the receiver's window and coordinate system.
- `lockFocusIfCanDraw` (page 70)
  Locks the focus to the receiver atomically if the `canDraw` method returns `YES` and returns the value of `canDraw`.
- `lockFocusIfCanDrawInContext:` (page 70)
  Locks the focus to the receiver atomically if drawing can occur in the specified graphics context.
- `unlockFocus` (page 114)
  Balances an earlier `lockFocus` (page 69) or `lockFocusIfCanDraw` (page 70) message; restoring the focus to the previously focused view is necessary.

+ `focusView` (page 25)
  Returns the currently focused `NSView` object, or `nil` if there is none.

Displaying

- `setNeedsDisplay:` (page 103)
  Controls whether the receiver's entire bounds is marked as needing display.
- `setNeedsDisplayInRect:` (page 103)
  Marks the region of the receiver within the specified rectangle as needing display, increasing the receiver's existing invalid region to include it.
- `needsDisplay` (page 73)
  Returns `YES` if the receiver needs to be displayed, as indicated using `setNeedsDisplay:` (page 103) and `setNeedsDisplayInRect:` (page 103); returns `NO` otherwise.
- `display` (page 48)
  Displays the receiver and all its subviews if possible, invoking each the `NSView` methods `lockFocus` (page 69), `drawRect:` (page 55), and `unlockFocus` (page 114) as necessary.
- `displayRect:` (page 50)
  Acts as `display` (page 48), but confining drawing to a rectangular region of the receiver.

`- displayRectIgnoringOpacity:` (page 50)
  Displays the receiver but confines drawing to a specified region and does not back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

`- displayRectIgnoringOpacity:inContext:` (page 51)
  Causes the receiver and its descendants to be redrawn to the specified graphics context.

`- displayIfNeeded` (page 49)
  Displays the receiver and all its subviews if any part of the receiver has been marked as needing display with a `setNeedsDisplay:` (page 103) or `setNeedsDisplayInRect:` (page 103) message.

`- displayIfNeededInRect:` (page 49)
  Acts as `displayIfNeeded` (page 49), confining drawing to a specified region of the receiver..

`- displayIfNeededIgnoringOpacity` (page 49)
  Acts as `displayIfNeeded` (page 49), except that this method doesn't back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

`- displayIfNeededInRectIgnoringOpacity:` (page 50)
  Acts as `displayIfNeeded` (page 49), but confining drawing to *`aRect`* and not backing up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

`- translateRectsNeedingDisplayInRect:by:` (page 113)
  Translates the display rectangles by the specified delta.

`- isOpaque` (page 66)
  Overridden by subclasses to return `YES` if the receiver is opaque, `NO` otherwise.

`- viewWillDraw` (page 117)
  Informs the receiver that it will be required to draw content.

Focus Ring Drawing

`- setKeyboardFocusRingNeedsDisplayInRect:` (page 102)
  Invalidates the area around the focus ring.

`+ defaultFocusRingType` (page 24)
  Returns the default focus ring type.

`- setFocusRingType:` (page 98)
  Sets the type of focus ring to be drawn around the receiver.

`- focusRingType` (page 59)
  Returns the type of focus ring drawn around the receiver.

Fullscreen Mode

`- enterFullScreenMode:withOptions:` (page 58)
  Sets the receiver to full screen mode.

`- exitFullScreenModeWithOptions:` (page 58)
  Instructs the receiver to exit full screen mode.

`- isInFullScreenMode` (page 66)
  Returns a Boolean value that indicates whether the receiver is in full screen mode.

Hiding Views

- `setHidden:` (page 101)
  Sets whether the receiver is hidden.
- `isHidden` (page 65)
  Returns whether the receiver is marked as hidden.
- `isHiddenOrHasHiddenAncestor` (page 66)
  Returns `YES` if the receiver is marked as hidden or has an ancestor in the view hierarchy that is marked as hidden; returns `NO` otherwise.
- `viewDidHide` (page 116)
  Invoked when the receiver is hidden, either directly, or in response to an ancestor being hidden.
- `viewDidUnhide` (page 117)
  Invoked when the receiver is unhidden, either directly, or in response to an ancestor being unhidden

Drawing

- `drawRect:` (page 55)
  Overridden by subclasses to draw the receiver's image within the passed-in rectangle.
- `visibleRect` (page 120)
  Returns the portion of the receiver not clipped by its superviews.
- `canDraw` (page 39)
  Returns `YES` if drawing commands will produce any result, `NO` otherwise.
- `shouldDrawColor` (page 109)
  Returns `NO` if the receiver is being drawn in an `NSWindow` object (as opposed, for example, to being printed) and the window object can't store color; otherwise returns `YES`.
- `getRectsBeingDrawn:count:` (page 60)
  Returns by indirection a list of non-overlapping rectangles that define the area the receiver is being asked to draw in `drawRect:` (page 55).
- `needsToDrawRect:` (page 73)
  Returns whether the specified rectangle intersects any part of the area that the receiver is being asked to draw.
- `wantsDefaultClipping` (page 121)
  Returns whether the Application Kit's default clipping provided to `drawRect:` (page 55) implementations is in effect.
- `bitmapImageRepForCachingDisplayInRect:` (page 36)
  Returns a bitmap-representation object suitable for caching the specified portion of the receiver.
- `cacheDisplayInRect:toBitmapImageRep:` (page 38)
  Draws the specified area of the receiver, and its descendants, into a provided bitmap-representation object.

Managing Live Resize

- `inLiveResize` (page 64)
  A convenience method, expected to be called from `drawRect:` (page 55), to assist in decisions about optimized drawing.

– `preservesContentDuringLiveResize` (page 78)
  Returns `YES` if the view supports the optimization of live-resize operations by preserving content that has not moved; otherwise, returns `NO`.

– `getRectsExposedDuringLiveResize:count:` (page 61)
  Returns a list of rectangles indicating the newly exposed areas of the receiver.

– `rectPreservedDuringLiveResize` (page 81)
  Returns the rectangle identifying the portion of your view that did not change during a live resize operation.

– `viewWillStartLiveResize` (page 119)
  Informs the receiver of the start of a live resize.

– `viewDidEndLiveResize` (page 115)
  Informs the receiver of the end of a live resize.

Managing the Graphics State

– `allocateGState` (page 32)
  Causes the receiver to maintain a private graphics state object, which encapsulates all parameters of the graphics environment.

– `gState` (page 62)
  Returns the identifier for the receiver's graphics state object, or 0 if the receiver doesn't have a graphics state object.

– `setUpGState` (page 107)
  Overridden by subclasses to (re)initialize the receiver's graphics state object.

– `renewGState` (page 86)
  Invalidates the receiver's graphics state object, if it has one.

– `releaseGState` (page 83)
  Frees the receiver's graphics state object, if it has one.

Event Handling

– `acceptsFirstMouse:` (page 25)
  Overridden by subclasses to return `YES` if the receiver should be sent a `mouseDown:` message for an initial mouse-down event, `NO` if not.

– `hitTest:` (page 63)
  Returns the farthest descendant of the receiver in the view hierarchy (including itself) that contains a specified point, or `nil` if that point lies completely outside the receiver.

– `mouse:inRect:` (page 71)
  Returns whether a region of the receiver contains a specified point, accounting for whether the receiver is flipped or not.

– `performKeyEquivalent:` (page 76)
  Implemented by subclasses to respond to key equivalents (also known as keyboard shortcuts).

– `performMnemonic:` (page 77)
  Implemented by subclasses to respond to mnemonics.

- `mouseDownCanMoveWindow` (page 72)
  Returns `YES` if the receiver does not need to handle a mouse down and can pass it through to superviews; `NO` if it needs to handle the mouse down.

Dragging Operations

- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 52)
  Initiates a dragging operation from the receiver, allowing the user to drag arbitrary data with a specified icon into any application that has window or view objects that accept dragged data.
- `dragFile:fromRect:slideBack:event:` (page 51)
  Initiates a dragging operation from the receiver, allowing the user to drag a file icon to any application that has window or view objects that accept files.
- `registerForDraggedTypes:` (page 82)
  Registers the pasteboard types that the receiver will accept as the destination of an image-dragging session.
- `registeredDraggedTypes` (page 82)
  Returns the array of pasteboard drag types that the view can accept.
- `unregisterDraggedTypes` (page 114)
  Unregisters the receiver as a possible destination in a dragging session.
- `shouldDelayWindowOrderingForEvent:` (page 108)
  Overridden by subclasses to allow the user to drag images from the receiver without its window moving forward and possibly obscuring the destination and without activating the application.
- `dragPromisedFilesOfTypes:fromRect:source:slideBack:event:` (page 54)
  Initiates a dragging operation from the receiver, allowing the user to drag one or more promised files (or directories) into any application that has window or view objects that accept promised file data.

Tool Tips

- `addToolTipRect:owner:userData:` (page 28)
  Creates a tool tip for a defined area the receiver and returns a tag that identifies the tool tip rectangle.
- `removeAllToolTips` (page 83)
  Removes all tool tips assigned to the receiver.
- `removeToolTip:` (page 85)
  Removes the tool tip identified by specified tag.
- `setToolTip:` (page 107)
  Sets the tool tip text for the view to *`string`*.
- `toolTip` (page 112)
  Returns the text for the view’s tool tip.

Managing Tracking Rectangles

- `addTrackingRect:owner:userData:assumeInside:` (page 29)
  Establishes an area for tracking mouse-entered and mouse-exited events within the receiver and returns a tag that identifies the tracking rectangle.

- `removeTrackingRect:` (page 85)
  Removes the tracking rectangle identified by a tag.

Managing Tracking Areas

- `addTrackingArea:` (page 28)
  Adds a given tracking area to the receiver.
- `removeTrackingArea:` (page 85)
  Removes a given tracking area from the receiver.
- `trackingAreas` (page 112)
  Returns an array of the receiver's tracking areas.
- `updateTrackingAreas` (page 115)
  Invoked automatically when the view's geometry changes such that its tracking areas need to be recalculated.

Managing Cursor Tracking

- `addCursorRect:cursor:` (page 26)
  Establishes the cursor to be used when the mouse pointer lies within a specified region.
- `removeCursorRect:cursor:` (page 83)
  Completely removes a cursor rectangle from the receiver.
- `discardCursorRects` (page 48)
  Invalidates all cursor rectangles set up using `addCursorRect:cursor:` (page 26).
- `resetCursorRects` (page 87)
  Overridden by subclasses to define their default cursor rectangles.

Scrolling

- `scrollPoint:` (page 90)
  Scrolls the receiver's closest ancestor `NSClipView` object so a point in the receiver lies at the origin of the clip view's bounds rectangle.
- `scrollRectToVisible:` (page 91)
  Scrolls the receiver's closest ancestor `NSClipView` object the minimum distance needed so a specified region of the receiver becomes visible in the clip view.
- `autoscroll:` (page 34)
  Scrolls the receiver's closest ancestor `NSClipView` object proportionally to the distance of an event that occurs outside of it.
- `adjustScroll:` (page 31)
  Overridden by subclasses to modify a given rectangle, returning the altered rectangle.
- `scrollRect:by:` (page 90)
  Copies the visible portion of the receiver's rendered image within a region and lays that portion down again at a specified offset .

- `enclosingScrollView` (page 57)
  Returns the nearest ancestor `NSScrollView` object containing the receiver (not including the receiver itself); otherwise returns `nil`.
- `scrollClipView:toPoint:` (page 89)
  Notifies the superview of a clip view that the clip view needs to reset the origin of its bounds rectangle.
- `reflectScrolledClipView:` (page 81)
  Notifies a clip view's superview that either the clip view's bounds rectangle or the document view's frame rectangle has changed, and that any indicators of the scroll position need to be adjusted.

Contextual Menus

- `menuForEvent:` (page 71)
  Overridden by subclasses to return a context-sensitive pop-up menu for a given mouse-down event.

+ `defaultMenu` (page 24)
  Overridden by subclasses to return the default pop-up menu for instances of the receiving class.

Key-view Loop Management

- `canBecomeKeyView` (page 38)
  Returns `YES` if the receiver can become key view, `NO` otherwise.
- `needsPanelToBecomeKey` (page 73)
  Overridden by subclasses to return `YES` if the receiver requires its panel, which might otherwise avoid becoming key, to become the key window so that it can handle keyboard input.
- `setNextKeyView:` (page 104)
  Inserts a specified view object after the receiver in the key view loop of the receiver's window.
- `nextKeyView` (page 74)
  Returns the view object following the receiver in the key view loop, or `nil` if there is none.
- `nextValidKeyView` (page 74)
  Returns the closest view object in the key view loop that follows the receiver and actually accepts first responder status, or `nil` if there is none.
- `previousKeyView` (page 78)
  Returns the view object preceding the receiver in the key view loop, or `nil` if there is none.
- `previousValidKeyView` (page 79)
  Returns the closest view object in the key view loop that precedes the receiver and actually accepts first responder status, or `nil` if there is none.

Printing

- `print:` (page 79)
  This action method opens the Print panel, and if the user chooses an option other than canceling, prints the receiver and all its subviews to the device specified in the Print panel.
- `beginPageInRect:atPlacement:` (page 35)
  Called at the beginning of each page, this method sets up the coordinate system so that a region inside the receiver's bounds is translated to a specified location..

- `dataWithEPSInsideRect:` (page 46)
  Returns EPS data that draws the region of the receiver within a specified rectangle.
- `dataWithPDFInsideRect:` (page 47)
  Returns PDF data that draws the region of the receiver within a specified rectangle.
- `printJobTitle` (page 80)
  Returns the receiver's print job title.
- `pageFooter` (page 75)
  Returns a default footer string that includes the current page number and page count.
- `pageHeader` (page 76)
  Returns a default header string that includes the print job title and date.
- `writeEPSInsideRect:toPasteboard:` (page 123)
  Writes EPS data that draws the region of the receiver within a specified rectangle onto a pasteboard.
- `writePDFInsideRect:toPasteboard:` (page 123)
  Writes PDF data that draws the region of the receiver within a specified rectangle onto a pasteboard.
- `drawPageBorderWithSize:` (page 55)
  Allows applications that use the Application Kit pagination facility to draw additional marks on each logical page.
- `drawSheetBorderWithSize:` (page 56)
  Allows applications that use the Application Kit pagination facility to draw additional marks on each printed sheet.

Pagination

- `heightAdjustLimit` (page 62)
  Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as lines of text from being divided across pages.
- `widthAdjustLimit` (page 121)
  Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as small images or text columns from being divided across pages.
- `adjustPageWidthNew:left:right:limit:` (page 31)
  Overridden by subclasses to adjust page width during automatic pagination.
- `adjustPageHeightNew:top:bottom:limit:` (page 30)
  Overridden by subclasses to adjust page height during automatic pagination.
- `knowsPageRange:` (page 68)
  Returns `YES` if the receiver handles page boundaries, `NO` otherwise.
- `rectForPage:` (page 80)
  Implemented by subclasses to determine the portion of the receiver to be printed for the page number page.
- `locationOfPrintRect:` (page 69)
  Invoked by `print:` (page 79) to determine the location of the region of the receiver being printed on the physical page.

Writing Conforming Rendering Instructions

- `beginDocument` (page 35)
  Invoked at the beginning of the printing session, this method sets up the current graphics context.
- `endDocument` (page 57)
  This method is invoked at the end of the printing session.
- `endPage` (page 58)
  Writes the end of a conforming page.

Core Animation Layer-Backing

- `setLayer:` (page 102)
  Sets the Core Animation layer that the receiver uses for layer-backing to the specified layer.
- `layer` (page 68)
  Returns the Core Animation layer that the receiver as its backing store.
- `setWantsLayer:` (page 108)
  Specifes whether the receiver and its subviews use a Core Animation layer as a backing store.
- `wantsLayer` (page 121)
  Returns a Boolean value that indicates whether the receiver is using a layer as it's backing store.

Core Animation Layer Properties

- `setFrameCenterRotation:` (page 99)
  Rotates the frame of the receiver about the layer's position.
- `frameCenterRotation` (page 60)
  Returns the receiver's rotation about the layer's position.
- `setAlphaValue:` (page 92)
  Sets the opacity of the receiver.
- `alphaValue` (page 32)
  Returns the opacity of the receiver
- `setBackgroundFilters:` (page 93)
  An array of CoreImage filters that are applied to the receiver's background.
- `backgroundFilters` (page 35)
  Returns the array of CoreImage filters that are applied to the receiver's background
- `setCompositingFilter:` (page 97)
  Sets a CoreImage filter that is used to composite the receiver's contents with the background.
- `compositingFilter` (page 40)
  Returns the CoreImage filter that is used to composite the receiver's contents with the background
- `setContentFilters:` (page 97)
  Sets the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.
- `contentFilters` (page 40)
  Returns the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.

- `setShadow:` (page 106)
  Sets the shadow drawn by the receiver.
- `shadow` (page 108)
  Returns the shadow drawn by the receiver

Class Methods

defaultFocusRingType

Returns the default focus ring type.

```
+ (NSFocusRingType)defaultFocusRingType
```

Return Value
The default type of focus ring for objects of the receiver's class. Possible return values are listed in `NSFocusRingType`.

Discussion
If `NSFocusRingTypeDefault` is returned from the instance method `focusRingType` (page 59), the receiver can invoke this class method to find out what type of focus ring is the default. The receiver is free to ignore the default setting.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSView.h`

defaultMenu

Overridden by subclasses to return the default pop-up menu for instances of the receiving class.

```
+ (NSMenu *)defaultMenu
```

Discussion
The default implementation returns `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `menuForEvent:` (page 71)
- `menu` (`NSResponder`)

Declared In
`NSView.h`

focusView

Returns the currently focused `NSView` object, or `nil` if there is none.

```
+ (NSView *)focusView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`– lockFocus` (page 69)
`– unlockFocus` (page 114)

Declared In
`NSView.h`

Instance Methods

acceptsFirstMouse:

Overridden by subclasses to return `YES` if the receiver should be sent a `mouseDown:` message for an initial mouse-down event, `NO` if not.

```
- (BOOL)acceptsFirstMouse:(NSEvent *)theEvent
```

Parameters

`theEvent`
    The initial mouse-down event, which must be over the receiver in its window.

Discussion
The receiver can either return a value unconditionally or use the location of *`theEvent`* to determine whether or not it wants the event. The default implementation ignores *`theEvent`* and returns `NO`.

Override this method in a subclass to allow instances to respond to click-through. This allows the user to click on a view in an inactive window, activating the view with one click, instead of clicking first to make the window active and then clicking the view. Most view objects refuse a click-through attempt, so the event simply activates the window. Many control objects, however, such as instances of `NSButton` and `NSSlider`, do accept them, so the user can immediately manipulate the control without having to release the mouse button.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– hitTest:` (page 63)

Declared In
`NSView.h`

addCursorRect:cursor:

Establishes the cursor to be used when the mouse pointer lies within a specified region.

```
- (void)addCursorRect:(NSRect)aRect cursor:(NSCursor *)aCursor
```

Parameters

aRect

A rectangle defining a region of the receiver.

aCursor

An object representing a cursor.

Discussion

Cursor rectangles aren't subject to clipping by superviews, nor are they intended for use with rotated views. You should explicitly confine a cursor rectangle to the view's visible rectangle to prevent improper behavior.

This method is intended to be invoked only by the `resetCursorRects` (page 87) method. If invoked in any other way, the resulting cursor rectangle will be discarded the next time the view's cursor rectangles are rebuilt.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `removeCursorRect:cursor:` (page 83)
- `discardCursorRects` (page 48)
- `resetCursorRects` (page 87)
- `visibleRect` (page 120)

Declared In

`NSView.h`

addSubview:

Adds a view to the receiver's subviews so it's displayed above its siblings.

```
- (void)addSubview:(NSView *)aView
```

Parameters

aView

The view to add to the receiver as a subview.

Discussion

This method also sets the receiver as the next responder of *aView*.

The receiver retains *aView*. If you use `removeFromSuperview` (page 84) to remove *aView* from the view hierarchy, *aView* is released. If you want to keep using *aView* after removing it from the view hierarchy (if, for example, you are swapping through a number of views), you must retain it before invoking `removeFromSuperview` (page 84).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addSubview:positioned:relativeTo:` (page 27)
- `subviews` (page 110)
- `removeFromSuperview` (page 84)
- `setNextResponder:` (NSResponder)
- `viewWillMoveToSuperview:` (page 118)
- `viewWillMoveToWindow:` (page 118)

Related Sample Code

iSpend

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

WhackedTV

Declared In

`NSView.h`

addSubview:positioned:relativeTo:

Inserts a view among the receiver's subviews so it's displayed immediately above or below another view.

```
- (void)addSubview:(NSView *)aView positioned:(NSWindowOrderingMode)place
    relativeTo:(NSView *)otherView
```

Parameters

aView

The view object to add to the receiver as a subview.

place

An `enum` constant specifying the position of the `aView` relative to `otherView`. Valid values are `NSWindowAbove` or `NSWindowBelow`.

otherView

The other view `aView` is to be positioned relative to. If *otherView* is `nil` (or isn't a subview of the receiver), *aView* is added above or below all of its new siblings.

Discussion

This method also sets the receiver as the next responder of *aView*.

The receiver retains *aView*. If you use `removeFromSuperview` (page 84) to remove *aView* from the view hierarchy, *aView* is released. If you want to keep using *aView* after removing it from the view hierarchy (if, for example, you are swapping through a number of views), you must retain it before invoking `removeFromSuperview` (page 84).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addSubview:` (page 26)
- `subviews` (page 110)
- `removeFromSuperview` (page 84)
- `setNextResponder:` (NSResponder)

Declared In
`NSView.h`

addToolTipRect:owner:userData:

Creates a tool tip for a defined area the receiver and returns a tag that identifies the tool tip rectangle.

```
- (NSToolTipTag)addToolTipRect:(NSRect)aRect owner:(id)anObject userData:(void
    *)userData
```

Parameters

`aRect`
A rectangle defining the region of the receiver to associate the tool tip with.

`anObject`
An object from which to obtain the tool tip string. The object should either implement `view:stringForToolTip:point:userData:`, or return a suitable string from its `description` method. (It can therefore simply be an `NSString` object.)

Important: The receiver maintains a weak reference to *`anObject`*. You are responsible for ensuring that *`anObject`* remains valid for as long as it may be needed.

`userData`
Any additional information you want to pass to `view:stringForToolTip:point:userData:`; it is not used if *`anObject`* does not implement this method.

Return Value
An integer tag identifying the tool tip; you can use this tag to remove the tool tip.

Discussion
The tool tip string is obtained dynamically from *`anObject`* by invoking either the `NSToolTipOwner` informal protocol method `view:stringForToolTip:point:userData:`, if implemented, or the `NSObject` protocol method `description`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– removeToolTip:` (page 85)
`– removeAllToolTips` (page 83)

Declared In
`NSView.h`

addTrackingArea:

Adds a given tracking area to the receiver.

```
- (void)addTrackingArea:(NSTrackingArea *)trackingArea
```

Parameters

trackingArea

The tracking area to add to the receiver.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
BasicCocoaAnimations

Declared In
NSView.h

addTrackingRect:owner:userData:assumeInside:

Establishes an area for tracking mouse-entered and mouse-exited events within the receiver and returns a tag that identifies the tracking rectangle.

```
- (NSTrackingRectTag)addTrackingRect:(NSRect)aRect owner:(id)userObject
    userData:(void *)userData assumeInside:(BOOL)flag
```

Parameters

aRect

A rectangle that defines a region of the receiver for tracking mouse-entered and mouse-exited events.

userObject

The object that gets sent the event messages. It can be the receiver itself or some other object (such as an NSCursor or a custom drawing tool object), as long as it responds to both `mouseEntered:` and `mouseExited:`.

userData

Data stored in the `NSEvent` object for each tracking event.

flag

If `YES`, the first event will be generated when the cursor leaves *aRect*, regardless if the cursor is inside *aRect* when the tracking rectangle is added. If `NO` the first event will be generated when the cursor leaves *aRect* if the cursor is initially inside *aRect*, or when the cursor enters *aRect* if the cursor is initially outside *aRect*. You usually want to set this flag to `NO`.

Return Value
A tag that identifies the tracking rectangle. It is stored in the associated `NSEvent` objects and can be used to remove the tracking rectangle.

Discussion
Tracking rectangles provide a general mechanism that can be used to trigger actions based on the cursor location (for example, a status bar or hint field that provides information on the item the cursor lies over). To simply change the cursor over a particular area, use `addCursorRect:cursor:` (page 26). If you must use tracking rectangles to change the cursor, the `NSCursor` class specification describes the additional methods that must be invoked to change cursors by using tracking rectangles.

On Mac OS X v10.5 and later, tracking areas provide a greater range of functionality (see `addTrackingArea:` (page 28)).

Availability
Available in Mac OS X v10.0 and later.

See Also

- `removeTrackingRect:` (page 85)
- `addTrackingArea:` (page 28)
- `userData` (`NSEvent`)

Related Sample Code

FunkyOverlayWindow

Declared In

`NSView.h`

adjustPageHeightNew:top:bottom:limit:

Overridden by subclasses to adjust page height during automatic pagination.

```
- (void)adjustPageHeightNew:(CGFloat *)newBottom top:(CGFloat)top
    bottom:(CGFloat)proposedBottom limit:(CGFloat)bottomLimit
```

Parameters

`newBottom`
: Returns by indirection a new `float` value for the bottom edge of the pending page rectangle in the receiver's coordinate system.

`top`
: A `float` value that sets the top edge of the pending page rectangle in the receiver's coordinate system.

`proposedBottom`
: A `float` value that sets the bottom edge of the pending page rectangle in the receiver's coordinate system.

`bottomLimit`
: The topmost float value *`newBottom`* can be set to, as calculated using the return value of `heightAdjustLimit` (page 62).

Discussion

This method is invoked by `print:` (page 79). The receiver can raise the bottom edge and return the new value in *`newBottom`*, allowing it to prevent items such as lines of text from being divided across pages. If *`bottomLimit`* is exceeded, the pagination mechanism simply uses *`bottomLimit`* for the bottom edge.

The default implementation of this method propagates the message to its subviews, allowing nested views to adjust page height for their drawing as well. An `NSButton` object or other small view, for example, will nudge the bottom edge up if necessary to prevent itself from being cut in two (thereby pushing it onto an adjacent page). Subclasses should invoke `super`'s implementation, if desired, after first making their own adjustments.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `adjustPageWidthNew:left:right:limit:` (page 31)

Declared In

`NSView.h`

adjustPageWidthNew:left:right:limit:

Overridden by subclasses to adjust page width during automatic pagination.

```
- (void)adjustPageWidthNew:(CGFloat *)newRight left:(CGFloat)left
    right:(CGFloat)proposedRight limit:(CGFloat)rightLimit
```

Parameters

newRight

Returns by indirection a new `float` value for the right edge of the pending page rectangle in the receiver's coordinate system.

left

A `float` value that sets the left edge of the pending page rectangle in the receiver's coordinate system.

proposedRight

A `float` value that sets the right edge of the pending page rectangle in the receiver's coordinate system.

rightLimit

The leftmost `float` value *newRight* can be set to, as calculated using the return value of `widthAdjustLimit` (page 121).

Discussion

This method is invoked by `print:` (page 79). The receiver can pull in the right edge and return the new value in *newRight*, allowing it to prevent items such as small images or text columns from being divided across pages. If *rightLimit* is exceeded, the pagination mechanism simply uses *rightLimit* for the right edge.

The default implementation of this method propagates the message to its subviews, allowing nested views to adjust page width for their drawing as well. An `NSButton` object or other small view, for example, will nudge the right edge out if necessary to prevent itself from being cut in two (thereby pushing it onto an adjacent page). Subclasses should invoke `super`'s implementation, if desired, after first making their own adjustments.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- adjustPageHeightNew:top:bottom:limit:` (page 30)

Declared In

`NSView.h`

adjustScroll:

Overridden by subclasses to modify a given rectangle, returning the altered rectangle.

```
- (NSRect)adjustScroll:(NSRect)proposedVisibleRect
```

Parameters

proposedVisibleRect

A rectangle defining a region of the receiver.

Discussion

`NSClipView` invokes this method to allow its document view to adjust its position during scrolling. For example, a custom view object that displays a table of data can adjust the origin of *`proposedVisibleRect`* so rows or columns aren't cut off by the edge of the enclosing `NSClipView`. `NSView`'s implementation simply returns *`proposedVisibleRect`*.

`NSClipView` only invokes this method during automatic or user controlled scrolling. Its `scrollToPoint:` method doesn't invoke this method, so you can still force a scroll to an arbitrary point.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

allocateGState

Causes the receiver to maintain a private graphics state object, which encapsulates all parameters of the graphics environment.

```
- (void)allocateGState
```

Discussion

If you do not invoke `allocateGState`, a graphics state object is constructed from scratch each time the NSView is focused.

The receiver builds the graphics state parameters using `setUpGState` (page 107), then automatically establishes this graphics state each time the focus is locked on it. A graphics state may improve performance for view objects that are focused often and need to set many parameters, but use of standard rendering operators is normally efficient enough.

Because graphics states occupy a fair amount of memory, they can actually degrade performance. Be sure to test application performance with and without the private graphics state before committing to its use.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setUpGState` (page 107)
- `gState` (page 62)
- `renewGState` (page 86)
- `releaseGState` (page 83)
- `lockFocus` (page 69)

Declared In

`NSView.h`

alphaValue

Returns the opacity of the receiver

```
- (CGFloat)alphaValue
```

Return Value
The current opacity of the receiver

Discussion
This method returns the value of the `opacity` property of the receiver’s layer. Possible values are between 0.0 (transparent) and 1.0 (opaque). The default is 1.0.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
RoundTransparentWindow

Declared In
`NSView.h`

ancestorSharedWithView:

Returns the closest ancestor shared by the receiver and a given view.

```
- (NSView *)ancestorSharedWithView:(NSView *)aView
```

Parameters

`aView`
The view to test (along with the receiver) for closest shared ancestor.

Return Value
The closest ancestor or `nil` if there’s no such object. Returns `self` if *`aView`* is identical to the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isDescendantOf:` (page 64)

Declared In
`NSView.h`

autoresizesSubviews

Returns `YES` if the receiver automatically resizes its subviews using `resizeSubviewsWithOldSize:` (page 87) whenever its frame size changes, `NO` otherwise.

```
- (BOOL)autoresizesSubviews
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAutoresizesSubviews:` (page 92)

Declared In
`NSView.h`

autoresizingMask

Returns the receiver's autoresizing mask, which determines how it's resized by the `resizeWithOldSuperviewSize:` (page 88) method.

```
- (NSUInteger)autoresizingMask
```

Return Value
An integer bit mask specified by combining using the C bitwise OR operator any of the options described in "`Resizing masks`" (page 124).

Discussion
If the autoresizing mask is equal to `NSViewNotSizable` (that is, if none of the options are set), then the receiver doesn't resize at all in `resizeWithOldSuperviewSize:` (page 88).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
PredicateEditorSample

Declared In
`NSView.h`

autoscroll:

Scrolls the receiver's closest ancestor `NSClipView` object proportionally to the distance of an event that occurs outside of it.

```
- (BOOL)autoscroll:(NSEvent *)theEvent
```

Parameters

`theEvent`
: An event object whose location should be expressed in the window's base coordinate system (which it normally is), not the receiving view's.

Return Value
Returns `YES` if any scrolling is performed; otherwise returns `NO`.

Discussion
View objects that track mouse-dragged events can use this method to scroll automatically when the cursor is dragged outside of the `NSClipView` object. Repeated invocations of this method (with an appropriate delay) result in continual scrolling, even when the mouse doesn't move.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- autoscroll:` (NSClipView)
`- scrollPoint:` (page 90)

`- isDescendantOf:` (page 64)

Declared In
`NSView.h`

backgroundFilters

Returns the array of CoreImage filters that are applied to the receiver's background

```
- (NSArray *)backgroundFilters
```

Return Value
An array of CoreImage filters.

Discussion
This method returns the value of the `backgroundFilters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

beginDocument

Invoked at the beginning of the printing session, this method sets up the current graphics context.

```
- (void)beginDocument
```

Discussion
Note that this method may be invoked in a subthread.

Override it to configure printing related settings. You should store your settings in the object returned by `NSPrintInfo`'s `sharedPrintInfo` class method, which is guaranteed to return an instance specific to the thread in which you invoke this method. If you override this method, call the superclass implementation.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

beginPageInRect:atPlacement:

Called at the beginning of each page, this method sets up the coordinate system so that a region inside the receiver's bounds is translated to a specified location..

```
- (void)beginPageInRect:(NSRect)aRect atPlacement:(NSPoint)location
```

Parameters

aRect
: A rectangle defining the region to be translated.

location
: A point that is the end-point of translation.

Discussion

If you override this method, be sure to call the superclass implementation.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

bitmapImageRepForCachingDisplayInRect:

Returns a bitmap-representation object suitable for caching the specified portion of the receiver.

```
- (NSBitmapImageRep *)bitmapImageRepForCachingDisplayInRect:(NSRect)aRect
```

Parameters

aRect
: A rectangle defining the area of the receiver to be cached.

Return Value

An autoreleased `NSBitmapImageRep` object or `nil` if the object could not be created.

Discussion

Passing the visible rectangle of the receiver (`[self visibleRect]`) returns a bitmap suitable for caching the current contents of the view, including all of its descendants.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- cacheDisplayInRect:toBitmapImageRep:` (page 38)

Related Sample Code

CarbonCocoaCoreImageTab

Reducer

Declared In

`NSView.h`

bounds

Returns the receiver's bounds rectangle, which expresses its location and size in its own coordinate system.

```
- (NSRect)bounds
```

Discussion
By default, the origin of the returned rectangle is (0, 0) and its size matches the size of the receiver's frame rectangle (measured in points). In Mac OS X v10.5 and later, if the receiver is being rendered into an OpenGL graphics context (using an `NSOpenGLContext` object), the default bounds origin is still (0, 0) but the default bounds size is measured in pixels instead of points. Thus, for user space scale factors other than 1.0, the default size of the bounds rectangle may be bigger or smaller than the default size of the frame rectangle when drawing with OpenGL.

Important: Developers of OpenGL applications should not rely on this method converting coordinates to pixels automatically in future releases. Instead, you should convert coordinates to device space explicitly using the `convertPointToBase:` (page 42), `convertSizeToBase:` (page 46), or `convertRectToBase:` (page 44) methods or their earlier counterparts `convertPoint:toView:` (page 41), `convertSize:toView:` (page 45), or `convertRect:toView:` (page 43).

If you explicitly change the origin or size of the bounds rectangle, this method does not return the default rectangle and instead returns the rectangle you set. If you add a rotation factor to the view, however, that factor is also reflected in the returned bounds rectangle. You can determine if a rotation factor is in effect by calling the `boundsRotation` (page 37) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
– `frame` (page 59)
– `setBounds:` (page 94)

Related Sample Code
Cocoa - SGDataProc
GLChildWindowDemo
LiveVideoMixer2
Sketch-112
WhackedTV

Declared In
`NSView.h`

boundsRotation

Returns the angle, in degrees, of the receiver's bounds rectangle relative to its frame rectangle.

```
- (CGFloat)boundsRotation
```

Discussion
See the `setBoundsRotation:` (page 95) method description for more information on bounds rotation.

Availability
Available in Mac OS X v10.0 and later.

See Also
– `rotateByAngle:` (page 88)
– `setBoundsRotation:` (page 95)

Declared In
`NSView.h`

cacheDisplayInRect:toBitmapImageRep:

Draws the specified area of the receiver, and its descendants, into a provided bitmap-representation object.

```
- (void)cacheDisplayInRect:(NSRect)rect toBitmapImageRep:(NSBitmapImageRep
    *)bitmapImageRep
```

Parameters

`rect`

A rectangle defining the region to be drawn into *`bimapImageRep`*.

`bitmapImageRep`

An `NSBitmapImageRep` object. For pixel-format compatibility, *`bitmapImageRep`* should have been obtained from `bitmapImageRepForCachingDisplayInRect:` (page 36).

Discussion
You are responsible for initializing the bitmap to the desired configuration before calling this method. However, once initialized, you can reuse the same bitmap multiple times to refresh the cached copy of your view's contents.

The bitmap produced by this method is transparent (that is, has an alpha value of 0) wherever the receiver and its descendants do not draw any content.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- bitmapImageRepForCachingDisplayInRect:` (page 36)

Related Sample Code
CarbonCocoaCoreImageTab

Reducer

Declared In
`NSView.h`

canBecomeKeyView

Returns `YES` if the receiver can become key view, `NO` otherwise.

```
- (BOOL)canBecomeKeyView
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSView.h`

canDraw

Returns `YES` if drawing commands will produce any result, `NO` otherwise.

```
- (BOOL)canDraw
```

Discussion
Use this method when invoking a draw method directly along with `lockFocus` (page 69) and `unlockFocus` (page 114), bypassing the `display...` methods (which test drawing ability and perform locking for you). If this method returns `NO`, you shouldn't invoke `lockFocus` (page 69) or perform any drawing.

A view object can draw on-screen if it is not hidden, it is attached to a view hierarchy in a window (`NSWindow`), and the window has a corresponding window device. A view object can draw during printing if it is a descendant of the view being printed.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setHidden:` (page 101)

Declared In
`NSView.h`

centerScanRect:

Converts the corners of a specified rectangle to lie on the center of device pixels, which is useful in compensating for rendering overscanning when the coordinate system has been scaled.

```
- (NSRect)centerScanRect:(NSRect)aRect
```

Parameters

`aRect`
The rectangle whose corners are to be converted.

Return Value
The adjusted rectangle.

Discussion
This method converts the given rectangle to device coordinates, adjusts the rectangle to lie in the center of the pixels, and converts the resulting rectangle back to the receiver's coordinate system. Note that this method does not take into account any transformations performed using the `NSAffineTransform` class or Quartz 2D routines.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isRotatedOrScaledFromBase` (page 67)

Related Sample Code
Sketch-112

Declared In
`NSView.h`

compositingFilter

Returns the CoreImage filter that is used to composite the receiver's contents with the background

```
- (CIFilter *)compositingFilter
```

Return Value
The CoreImage filter.

Discussion
This method returns the value of the `filters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

contentFilters

Returns the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.

```
- (NSArray *)contentFilters
```

Return Value
An array of CoreImage filters

Discussion
This method returns the value of the `filters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

convertPoint:fromView:

Converts a point from the coordinate system of a given view to that of the receiver.

```
- (NSPoint)convertPoint:(NSPoint)aPoint fromView:(NSView *)aView
```

Parameters
`aPoint`
A point specifying a location in the coordinate system of `aView`.

aView
: The view with `aPoint` in its coordinate system. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value
The point converted to the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertRect:fromView:` (page 42)
- `convertSize:fromView:` (page 44)
- `ancestorSharedWithView:` (page 33)
- `contentView` (NSWindow)

Related Sample Code
CIAnnotation
Clock Control
LiveVideoMixer2
Sketch-112
TextLinks

Declared In
`NSView.h`

convertPoint:toView:

Converts a point from the receiver's coordinate system to that of a given view.

```
- (NSPoint)convertPoint:(NSPoint)aPoint toView:(NSView *)aView
```

Parameters

aPoint
: A point specifying a location in the coordinate system of the receiver.

aView
: The view into whose coordinate system *aPoint* is to be converted. If *aView* is `nil`, this method instead converts to window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value
The point converted to the coordinate system of *aView*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertRect:toView:` (page 43)
- `convertSize:toView:` (page 45)
- `ancestorSharedWithView:` (page 33)
- `contentView` (`NSWindow`)

Declared In
`NSView.h`

convertPointFromBase:

Converts the point from the base coordinate system to the receiver's coordinate system.

```
- (NSPoint)convertPointFromBase:(NSPoint)aPoint
```

Parameters

aPoint
A point specifying a location in the base coordinate system.

Return Value
The point converted to the receiver's base coordinate system.

Discussion
See View Coordinate Conversion in *View Programming Guide for Cocoa* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

convertPointToBase:

Converts the point from the receiver's coordinate system to the base coordinate system.

```
- (NSPoint)convertPointToBase:(NSPoint)aPoint
```

Parameters

aPoint
A point specifying a location in the coordinate system of the receiver.

Return Value
The point converted to the base coordinate system.

Discussion
See View Coordinate Conversion in *View Programming Guide for Cocoa* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

convertRect:fromView:

Converts a rectangle from the coordinate system of another view to that of the receiver.

```
- (NSRect)convertRect:(NSRect)aRect fromView:(NSView *)aView
```

Parameters

aRect

The rectangle in `aView`'s coordinate system.

aView

The view with `aRect` in its coordinate system. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value

The converted rectangle.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `convertPoint:fromView:` (page 40)
- `convertSize:fromView:` (page 44)
- `ancestorSharedWithView:` (page 33)
- `contentView` (NSWindow)

Declared In

`NSView.h`

convertRect:toView:

Converts a rectangle from the receiver's coordinate system to that of another view.

```
- (NSRect)convertRect:(NSRect)aRect toView:(NSView *)aView
```

Parameters

aRect

A rectangle in the receiver's coordinate system.

aView

The view that is the target of the conversion operation. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value

The converted rectangle.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `convertPoint:toView:` (page 41)
- `convertSize:toView:` (page 45)
- `ancestorSharedWithView:` (page 33)
- `contentView` (NSWindow)

Declared In
`NSView.h`

convertRectFromBase:

Converts the rectangle from the base coordinate system to the receiver's coordinate system.

```
- (NSRect)convertRectFromBase:(NSRect)aRect
```

Parameters

aRect
  A rectangle in the base coordinate system

Return Value
A rectangle in the receiver's coordinate system

Discussion
See View Coordinate Conversion in *View Programming Guide for Cocoa* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

convertRectToBase:

Converts the rectangle from the receiver's coordinate system to the base coordinate system.

```
- (NSRect)convertRectToBase:(NSRect)aRect
```

Parameters

aRect
  A rectangle in the receiver's coordinate system

Return Value
A rectangle in the base coordinate system

Discussion
See View Coordinate Conversion in *View Programming Guide for Cocoa* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

convertSize:fromView:

Converts a size from another view's coordinate system to that of the receiver.

```
- (NSSize)convertSize:(NSSize)aSize fromView:(NSView *)aView
```

Parameters

aSize
: The size (width and height) in *aView*'s coordinate system.

aView
: The view with *aSize* in its coordinate system. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value

The converted size, as an `NSSize` structure.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `convertPoint:fromView:` (page 40)
- `convertRect:fromView:` (page 42)
- `ancestorSharedWithView:` (page 33)
- `contentView` (NSWindow)

Declared In

`NSView.h`

convertSize:toView:

Converts a size from the receiver's coordinate system to that of another view.

```
- (NSSize)convertSize:(NSSize)aSize toView:(NSView *)aView
```

Parameters

aSize
: The size (width and height) in the receiver's coordinate system.

aView
: The view that is the target of the conversion operation. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value

The converted size, as an `NSSize` structure.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `convertPoint:toView:` (page 41)
- `convertRect:toView:` (page 43)
- `ancestorSharedWithView:` (page 33)
- `contentView` (NSWindow)

Declared In
`NSView.h`

convertSizeFromBase:

Converts the size from the base coordinate system to the receiver's coordinate system.

```
- (NSSize)convertSizeFromBase:(NSSize)aSize
```

Parameters

aSize
A size in the base coordinate system

Return Value
The size converted to the receiver's coordinate system.

Discussion
See View Coordinate Conversion in *View Programming Guide for Cocoa* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

convertSizeToBase:

Converts the size from the receiver's coordinate system to the base coordinate system.

```
- (NSSize)convertSizeToBase:(NSSize)aSize
```

Parameters

aSize
A size in the receiver's coordinate system

Return Value
The size converted to the base coordinate system.

Discussion
See View Coordinate Conversion in *View Programming Guide for Cocoa* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

dataWithEPSInsideRect:

Returns EPS data that draws the region of the receiver within a specified rectangle.

```
- (NSData *)dataWithEPSInsideRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region.

Discussion

This data can be placed on an `NSPasteboard` object, written to a file, or used to create an `NSImage` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- writeEPSInsideRect:toPasteboard:` (page 123)

Declared In

`NSView.h`

dataWithPDFInsideRect:

Returns PDF data that draws the region of the receiver within a specified rectangle.

```
- (NSData *)dataWithPDFInsideRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region.

Discussion

This data can be placed on an `NSPasteboard` object, written to a file, or used to create an `NSImage` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- writePDFInsideRect:toPasteboard:` (page 123)

Related Sample Code

Sketch-112

Declared In

`NSView.h`

didAddSubview:

Overridden by subclasses to perform additional actions when subviews are added to the receiver.

```
- (void)didAddSubview:(NSView *)subview
```

Parameters

subview

The view that was added as a subview.

Discussion
This method is invoked by `addSubview:` (page 26).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

discardCursorRects

Invalidates all cursor rectangles set up using `addCursorRect:cursor:` (page 26).

```
- (void)discardCursorRects
```

Discussion
You need never invoke this method directly; neither is it typically invoked during the invalidation of cursor rectangles. `NSWindow` automatically invalidates cursor rectangles in response to `invalidateCursorRectsForView:` and before the receiver's cursor rectangles are reestablished using `resetCursorRects` (page 87). This method is invoked just before the receiver is removed from a window and when the receiver is deallocated.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- discardCursorRects` (NSWindow)

Declared In
`NSView.h`

display

Displays the receiver and all its subviews if possible, invoking each the `NSView` methods `lockFocus` (page 69), `drawRect:` (page 55), and `unlockFocus` (page 114) as necessary.

```
- (void)display
```

Discussion
If the receiver isn't opaque, this method backs up the view hierarchy to the first opaque ancestor, calculates the portion of the opaque ancestor covered by the receiver, and begins displaying from there.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- canDraw` (page 39)
`- opaqueAncestor` (page 75)
`- visibleRect` (page 120)
`- displayIfNeededIgnoringOpacity` (page 49)

Related Sample Code
CIVideoDemoGL

Cocoa - SGDataProc
LiveVideoMixer2
StickiesExample

Declared In
`NSView.h`

displayIfNeeded

Displays the receiver and all its subviews if any part of the receiver has been marked as needing display with a `setNeedsDisplay:` (page 103) or `setNeedsDisplayInRect:` (page 103) message.

```
- (void)displayIfNeeded
```

Discussion
This method invokes the `NSView` methods `lockFocus` (page 69), `drawRect:` (page 55), and `unlockFocus` (page 114) as necessary. If the receiver isn't opaque, this method backs up the view hierarchy to the first opaque ancestor, calculates the portion of the opaque ancestor covered by the receiver, and begins displaying from there.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– display` (page 48)
`– needsDisplay` (page 73)
`– displayIfNeededIgnoringOpacity` (page 49)

Declared In
`NSView.h`

displayIfNeededIgnoringOpacity

Acts as `displayIfNeeded` (page 49), except that this method doesn't back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

```
- (void)displayIfNeededIgnoringOpacity
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

displayIfNeededInRect:

Acts as `displayIfNeeded` (page 49), confining drawing to a specified region of the receiver..

```
- (void)displayIfNeededInRect:(NSRect)aRect
```

Parameters

aRect
: A rectangle defining the region to be redrawn. It should be specified in the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

displayIfNeededInRectIgnoringOpacity:

Acts as `displayIfNeeded` (page 49), but confining drawing to *aRect* and not backing up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

```
- (void)displayIfNeededInRectIgnoringOpacity:(NSRect)aRect
```

Parameters

aRect
: A rectangle defining the region to be redrawn. It should be specified in the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

displayRect:

Acts as `display` (page 48), but confining drawing to a rectangular region of the receiver.

```
- (void)displayRect:(NSRect)aRect
```

Parameters

aRect
: A rectangle defining the region of the receiver to be redrawn; should be specified in the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

displayRectIgnoringOpacity:

Displays the receiver but confines drawing to a specified region and does not back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

```
- (void)displayRectIgnoringOpacity:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region of the receiver to be redrawn; should be specified in the coordinate system of the receiver.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

displayRectIgnoringOpacity:inContext:

Causes the receiver and its descendants to be redrawn to the specified graphics context.

```
- (void)displayRectIgnoringOpacity:(NSRect)aRect inContext:(NSGraphicsContext
    *)context
```

Parameters

aRect

A rectangle defining the region of the receiver to be redrawn. It should be specified in the coordinate system of the receiver.

context

The graphics context in which drawing will occur. See the discussion below for more about this parameter.

Discussion

Acts as `display` (page 48), but confines drawing to *aRect*. This method initiates drawing with the receiver, even if the receiver is not opaque. Appropriate scaling factors for the view are obtained from *context*.

If the *context* parameter represents the context for the window containing the view, then all of the necessary transformations are applied. This includes the application of the receiver's bounds and frame transforms along with any transforms it inherited from its ancestors. In this situation, the view is also marked as no longer needing an update for the specified rectangle.

If *context* specifies any other graphics context, then only the receiver's bounds transform is applied. This means that drawing is not constrained to the view's visible rectangle. It also means that any dirty rectangles are not cleared, since they are not being redrawn to the window.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSView.h`

dragFile:fromRect:slideBack:event:

Initiates a dragging operation from the receiver, allowing the user to drag a file icon to any application that has window or view objects that accept files.

```
- (BOOL)dragFile:(NSString *)fullPath fromRect:(NSRect)aRect
    slideBack:(BOOL)slideBack event:(NSEvent *)theEvent
```

Parameters

fullPath

A string that specifies the absolute path for the file that is dragged.

aRect

A rectangle that describes the position of the icon in the receiver's coordinate system.

slideBack

A Boolean that indicates whether the icon being dragged should slide back to its position in the receiver if the file isn't accepted. The icon slides back to *aRect* if *slideBack* is `YES`, the file is not accepted by the dragging destination, and the user has not disabled icon animation; otherwise it simply disappears.

theEvent

The mouse-down event object from which to initiate the drag operation. In particular, its mouse location is used for the offset of the icon being dragged.

Return Value

`YES` if the receiver successfully initiates the dragging operation (which doesn't necessarily mean the dragging operation concluded successfully). Otherwise returns `NO`.

Discussion

This method must be invoked only within an implementation of the `mouseDown:` method.

See the `NSDraggingSource`, `NSDraggingInfo`, and `NSDraggingDestination` protocol specifications for more information on dragging operations.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 52)
- `shouldDelayWindowOrderingForEvent:` (page 108)

Declared In

`NSView.h`

dragImage:at:offset:event:pasteboard:source:slideBack:

Initiates a dragging operation from the receiver, allowing the user to drag arbitrary data with a specified icon into any application that has window or view objects that accept dragged data.

```
- (void)dragImage:(NSImage *)anImage at:(NSPoint)imageLoc offset:(NSSize)mouseOffset
     event:(NSEvent *)theEvent pasteboard:(NSPasteboard *)pboard
    source:(id)sourceObject slideBack:(BOOL)slideBack
```

Parameters

anImage

The `NSImage` object to be dragged.

imageLoc

The location of the image's lower-left corner, in the receiver's coordinate system. It determines the placement of the dragged image under the cursor. When determining the image location you should use the mouse down coordinate, provided in *theEvent*, rather than the current mouse location.

mouseOffset

The mouse's current location relative to the mouse-down location. In Mac OS X v10.4 and later, this parameter is ignored. In earlier versions of Mac OS X, this parameter is ignored when positioning the dragged image on screen but not ignored when passing the drag location to the source in `draggedImage:endedAt:operation:`.

Historically, this argument represented the difference between the mouse position when the drag operation commenced and the mouse position of the initial mouse down. If you initiated a dragging operation immediately on a mouse-down event, this offset was (0.0, 0.0). If you tested for a mouse-dragged event first, this offset was the difference between the mouse-dragged event's location and that of the mouse-down event. Since the underlying drag machinery now takes the current mouse position into account when positioning the drag image, this argument is no longer necessary

theEvent

The left mouse-down event that triggered the dragging operation (see discussion below).

pboard

The pasteboard that holds the data to be transferred to the destination (see discussion below).

sourceObject

An object that serves as the controller of the dragging operation. It must conform to the `NSDraggingSource` informal protocol and is typically the receiver itself or its `NSWindow` object.

slideBack

A Boolean that determines whether the drag image should slide back if it's rejected. The image slides back to *imageLoc* if *slideBack* is `YES` and the image isn't accepted by the dragging destination. If `NO` the image doesn't slide back.

Discussion
This method must be invoked only within an implementation of the `mouseDown:` or `mouseDragged:` methods.

Before invoking this method, you must place the data to be transferred on *pboard*. To do this, get the drag pasteboard object (`NSDragPboard`), declare the types of the data, and then put the data on the pasteboard. This code fragment initiates a dragging operation on an image itself (that is, the image is the data to be transferred):

```
- (void)mouseDown:(NSEvent *)theEvent
{
    NSSize dragOffset = NSMakeSize(0.0, 0.0);
    NSPasteboard *pboard;

    pboard = [NSPasteboard pasteboardWithName:NSDragPboard];
   [pboard declareTypes:[NSArray arrayWithObject:NSTIFFPboardType]  owner:self];
   [pboard setData:[[self image] TIFFRepresentation] forType:NSTIFFPboardType];

    [self dragImage:[self image] at:[self imageLocation] offset:dragOffset
        event:theEvent pasteboard:pboard source:self slideBack:YES];

    return;
}
```

See the `NSDraggingSource`, `NSDraggingInfo`, and `NSDraggingDestination` protocol specifications for more information on dragging operations.

Availability
Available in Mac OS X v10.0 and later.

See Also

- `dragFile:fromRect:slideBack:event:` (page 51)
- `shouldDelayWindowOrderingForEvent:` (page 108)

Declared In

`NSView.h`

dragPromisedFilesOfTypes:fromRect:source:slideBack:event:

Initiates a dragging operation from the receiver, allowing the user to drag one or more promised files (or directories) into any application that has window or view objects that accept promised file data.

```
- (BOOL)dragPromisedFilesOfTypes:(NSArray *)typeArray fromRect:(NSRect)aRect
    source:(id)sourceObject slideBack:(BOOL)slideBack event:(NSEvent *)theEvent
```

Parameters

typeArray

An array of file types being promised. The array elements can consist of file extensions and HFS types encoded with the `NSFileTypeForHFSTypeCode` function. If promising a directory of files, only include the top directory in the array.

aRect

A rectangle that describes the position of the icon in the receiver's coordinate system.

sourceObject

An object that serves as the controller of the dragging operation. It must conform to the `NSDraggingSource` informal protocol, and is typically the receiver itself or its `NSWindow` object.

slideBack

A Boolean that indicates whether the icon being dragged should slide back to its position in the receiver if the file isn't accepted. The icon slides back to *aRect* if *slideBack* is `YES`, the promised files are not accepted by the dragging destination, and the user has not disabled icon animation; otherwise it simply disappears.

theEvent

The mouse-down event object from which to initiate the drag operation. In particular, its mouse location is used for the offset of the icon being dragged.

Return Value

`YES` if the drag operation is initiated successfully, `NO` otherwise.

Discussion

This method must be invoked only within an implementation of the `mouseDown:` method. As part of its implementation, this method invokes `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 52).

Promised files are files that do not exist, yet, but that the drag source, *sourceObject*, promises to create at a file system location specified by the drag destination when the drag is successfully dropped.

See *Drag and Drop Programming Topics for Cocoa* for more information on dragging operations.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 52)

`- shouldDelayWindowOrderingForEvent:` (page 108)

Declared In
`NSView.h`

drawPageBorderWithSize:

Allows applications that use the Application Kit pagination facility to draw additional marks on each logical page.

```
- (void)drawPageBorderWithSize:(NSSize)borderSize
```

Parameters

borderSize
An `NSSize` structure that defines a logical page.

Discussion
The marks can be such things as alignment marks or a virtual sheet border of size *borderSize*. The default implementation doesn't draw anything.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawSheetBorderWithSize:` (page 56)

Declared In
`NSView.h`

drawRect:

Overridden by subclasses to draw the receiver's image within the passed-in rectangle.

```
- (void)drawRect:(NSRect)dirtyRect
```

Parameters

aRect
A rectangle defining the dirty area of the view that requires redrawing.

Discussion
The receiver can assume the focus has been locked and the coordinate transformations of its frame and bounds rectangles have been applied; all it needs to do is invoke rendering client functions.

You can avoid unnecessary drawing by paying attention to the *dirtyRect* parameter which defines the area that must be redrawn. The bounds of the entire view, as opposed to the dirty region, is given by [self bounds].

On Mac OS X version 10.2 and earlier, the Application Kit automatically clips any drawing you perform in this method to this rectangle. On Mac OS X version 10.3 and later, the Application Kit automatically clips drawing to a list of non-overlapping rectangles that more rigorously specify the area needing drawing. You can invoke the `getRectsBeingDrawn:count:` (page 60) method to retrieve this list of rectangles and use them to constrain your drawing more tightly, if you wish. Moreover, the `needsToDrawRect:` (page 73) method gives you a convenient way to test individual objects for intersection with the rectangles in the list. See *Cocoa Drawing Guide* for information and references on drawing.

The default implementation does nothing. If your custom view is a direct `NSView` subclass you do not need to call `super`'s implementation.

Note: If a subclass returns `YES` upon receiving an `isOpaque` (page 66) message, it must completely fill *`aRect`* with opaque content.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `display` (page 48)
- `getRectsBeingDrawn:count:` (page 60)
- `isFlipped` (page 65)
- `needsToDrawRect:` (page 73)
- `setNeedsDisplayInRect:` (page 103)
- `shouldDrawColor` (page 109)

Related Sample Code
QTCoreVideo102
QTCoreVideo103
QTCoreVideo201
QTCoreVideo301
WebKitDOMElementPlugIn

Declared In
`NSView.h`

drawSheetBorderWithSize:

Allows applications that use the Application Kit pagination facility to draw additional marks on each printed sheet.

```
- (void)drawSheetBorderWithSize:(NSSize)borderSize
```

Parameters

`borderSize`
An `NSSize` structure that defines a printed sheet.

Discussion
The marks can be such things as crop marks or fold lines of size *`borderSize`*. This method has been deprecated.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawPageBorderWithSize:` (page 55)

Declared In
`NSView.h`

enclosingMenuItem

Returns the menu item containing the receiver or any of its superviews in the view hierarchy.

```
- (NSMenuItem *)enclosingMenuItem
```

Return Value
Returns the menu item containing the receiver or any of its superviews in the view hierarchy, or `nil` if the receiver's view hierarchy is not in a menu item

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSMenuItem.h`

enclosingScrollView

Returns the nearest ancestor `NSScrollView` object containing the receiver (not including the receiver itself); otherwise returns `nil`.

```
- (NSScrollView *)enclosingScrollView
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CIAnnotation

CITransitionSelectorSample2

Sketch-112

Declared In
`NSView.h`

endDocument

This method is invoked at the end of the printing session.

```
- (void)endDocument
```

Discussion
If you override this method, call the superclass implementation.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

endPage

Writes the end of a conforming page.

```
- (void)endPage
```

Discussion
This method is invoked after each page is printed. It invokes `unlockFocus` (page 114). This method also generates comments for the bounding box and page fonts, if they were specified as being at the end of the page.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

enterFullScreenMode:withOptions:

Sets the receiver to full screen mode.

```
- (BOOL)enterFullScreenMode:(NSScreen *)screen withOptions:(NSDictionary *)options
```

Parameters

`screen`
The screen the receiver should cover.

`options`
A dictionary of options for the mode. For possible keys, see "`Full screen mode`" (page 126).

Return Value
`YES` if the receiver was able to enter full screen mode, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

exitFullScreenModeWithOptions:

Instructs the receiver to exit full screen mode.

```
- (void)exitFullScreenModeWithOptions:(NSDictionary *)options
```

Parameters

`options`
A dictionary of options for the mode. For possible keys, see "`Full screen mode`" (page 126).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

focusRingType

Returns the type of focus ring drawn around the receiver.

```
- (NSFocusRingType)focusRingType
```

Return Value
An `enum` constant identifying a type of focus ring. Possible values are listed in `NSFocusRingType`.

Discussion
You can disable a view's drawing of its focus ring by overriding this method to return `NSFocusRingTypeNone`, or by invoking `setFocusRingType:` (page 98) with and argument of `NSFocusRingTypeNone`. You should only disable the default drawing of a view's focus ring if you want it to draw its own focus ring (for example, setting the background color of the view), or if the view does not have sufficient space to display a focus ring.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setFocusRingType:` (page 98)

Declared In
`NSView.h`

frame

Returns the receiver's frame rectangle, which defines its position in its superview.

```
- (NSRect)frame
```

Discussion
The frame rectangle may be rotated; use the `frameRotation` (page 60) method to check this.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- bounds` (page 36)
`- setFrame:` (page 98)

Related Sample Code
MyPhoto
Quartz Composer QCTV
Quartz Composer WWDC 2005 TextEdit
Reducer
TextEditPlus

Declared In
`NSView.h`

frameCenterRotation

Returns the receiver's rotation about the layer's position.

```
- (CGFloat)frameCenterRotation
```

Return Value
The angle of rotatation of the frame around the center of the receiver.

Discussion
If the application has altered the layer's `anchorPoint` property, the behavior is undefined. Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

frameRotation

Returns the angle, in degrees, of the receiver's frame relative to its superview's coordinate system.

```
- (CGFloat)frameRotation
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFrameRotation:` (page 100)
- `boundsRotation` (page 37)

Declared In
`NSView.h`

getRectsBeingDrawn:count:

Returns by indirection a list of non-overlapping rectangles that define the area the receiver is being asked to draw in `drawRect:` (page 55).

```
- (void)getRectsBeingDrawn:(const NSRect **)rects count:(NSInteger *)count
```

Parameters

`rects`
: On return, contains a list of non-overlapping rectangles defining areas to be drawn in. The rectangles returned in *`rects`* are in the coordinate space of the receiver.

`count`
: On return, the number of rectangles in the `rects` list.

Discussion
An implementation of `drawRect:` can use this information to test whether objects or regions within the view intersect with the rectangles in the list, and thereby avoid unnecessary drawing that would be completely clipped away.

The `needsToDrawRect:` (page 73) method gives you a convenient way to test individual objects for intersection with the area being drawn in `drawRect:` (page 55). However, you may want to retrieve and directly inspect the rectangle list if this is a more efficient way to perform intersection testing.

You should send this message only from within a `drawRect:` (page 55) implementation. The *`aRect`* parameter of `drawRect:` is the rectangle enclosing the returned list of rectangles; you can use it in an initial pass to reject objects that are clearly outside the area to be drawn.

Availability
Available in Mac OS X v10.3 and later.

See Also
`– wantsDefaultClipping` (page 121)

Declared In
`NSView.h`

getRectsExposedDuringLiveResize:count:

Returns a list of rectangles indicating the newly exposed areas of the receiver.

```
- (void)getRectsExposedDuringLiveResize:(NSRect)exposedRects count:(NSInteger
    *)count
```

Parameters

`exposedRects`
On return, contains the list of rectangles. The returned rectangles are in the coordinate space of the receiver.

`count`
Contains the number of rectangles in *`exposedRects`*; this value may be 0 and is guaranteed to be no more than 4.

Discussion
If your view does not support content preservation during live resizing, the entire area of your view is returned in the *`exposedRects`* parameter. To support content preservation, override `preservesContentDuringLiveResize` (page 78) in your view and have your implementation return `YES`.

Note: The window containing your view must also support content preservation. To enable support for this feature in your window, use the `setPreservesContentDuringLiveResize:` method of `NSWindow`.

If the view decreased in both height and width, the list of returned rectangles will be empty. If the view increased in both height and width and its upper-left corner stayed anchored in the same position, the list of returned rectangles will contain a vertical and horizontal component indicating the exposed area.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– preservesContentDuringLiveResize` (page 78)
`– rectPreservedDuringLiveResize` (page 81)

Declared In
`NSView.h`

gState

Returns the identifier for the receiver's graphics state object, or 0 if the receiver doesn't have a graphics state object.

```
- (NSInteger)gState
```

Discussion
A view object's graphics state object is recreated from scratch whenever the view is focused, unless the `allocateGState` (page 32) method has been invoked. So if the receiver hasn't been focused or hasn't received the `allocateGState` (page 32) message, this method returns 0.

Although applications rarely need to use the value returned by `gState` (page 62), it can be passed to the few methods that take an object identifier as a parameter.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- allocateGState` (page 32)
`- setUpGState` (page 107)
`- renewGState` (page 86)
`- releaseGState` (page 83)
`- lockFocus` (page 69)

Declared In
`NSView.h`

heightAdjustLimit

Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as lines of text from being divided across pages.

```
- (CGFloat)heightAdjustLimit
```

Discussion
This fraction is used to calculate the bottom edge limit for an `adjustPageHeightNew:top:bottom:limit:` (page 30) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- widthAdjustLimit` (page 121)

Declared In
`NSView.h`

hitTest:

Returns the farthest descendant of the receiver in the view hierarchy (including itself) that contains a specified point, or `nil` if that point lies completely outside the receiver.

```
- (NSView *)hitTest:(NSPoint)aPoint
```

Parameters

aPoint

A point that is in the coordinate system of the receiver's superview, not of the receiver itself.

Return Value

A view object that is the farthest descendent of *aPoint*.

Discussion

This method is used primarily by an `NSWindow` object to determine which view should receive a mouse-down event. You'd rarely need to invoke this method, but you might want to override it to have a view object hide mouse-down events from its subviews. This method ignores hidden views.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `mouse:inRect:` (page 71)
- `convertPoint:toView:` (page 41)
- `setHidden:` (page 101)

Declared In

`NSView.h`

initWithFrame:

Initializes and returns a newly allocated `NSView` object with a specified frame rectangle.

```
- (id)initWithFrame:(NSRect)frameRect
```

Parameters

frameRect

The frame rectangle for the created view object.

Return Value

An initialized `NSView` object or nil if the object couldn't be created.

Discussion

The new view object must be inserted into the view hierarchy of a window before it can be used. This method is the designated initializer for the `NSView` class. Returns an initialized object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addSubview:` (page 26)
- `addSubview:positioned:relativeTo:` (page 27)
- `setFrame:` (page 98)

Declared In
`NSView.h`

inLiveResize

A convenience method, expected to be called from `drawRect:` (page 55), to assist in decisions about optimized drawing.

```
- (BOOL)inLiveResize
```

Return Value
`YES` if the receiver is in a live-resize operation, `NO` otherwise.

Availability
Available in Mac OS X v10.1 and later.

See Also
`– viewDidEndLiveResize` (page 115)
`– viewWillStartLiveResize` (page 119)

Related Sample Code
GLChildWindowDemo
NSGLImage
OpenGL Screensaver
WhackedTV

Declared In
`NSView.h`

isDescendantOf:

Returns `YES` if the receiver is a subview of a given view or if it's identical to that view; otherwise, it returns `NO`.

```
- (BOOL)isDescendantOf:(NSView *)aView
```

Parameters

aView
The view to test for subview relationship within the view hierarchy.

Discussion
The method returns `YES` if the receiver is either an immediate or distant subview of `aView`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– superview` (page 111)
`– subviews` (page 110)
`– ancestorSharedWithView:` (page 33)

Declared In
`NSView.h`

isFlipped

Returns `YES` if the receiver uses flipped drawing coordinates or `NO` if it uses native coordinates.

```
- (BOOL)isFlipped
```

Discussion
The default implementation returns `NO`; subclasses that use flipped coordinates should override this method to return `YES`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ImageMapExample
Quartz Composer QCTV
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TextEditPlus

Declared In
`NSView.h`

isHidden

Returns whether the receiver is marked as hidden.

```
- (BOOL)isHidden
```

Discussion
The return value reflects the state of the receiver only, as set in Interface Builder or through the most recent `setHidden:` (page 101) message, and does not account for the state of the receiver's ancestors in the view hierarchy, Thus this method returns `NO` when the receiver is effectively hidden because it has a hidden ancestor. See `setHidden:` for a discussion of the mechanics and implications of hidden views.

If you want to determine whether a view is effectively hidden, for whatever reason, send the `isHiddenOrHasHiddenAncestor` (page 66) to the view instead.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
GLSLShowpiece

Declared In
`NSView.h`

isHiddenOrHasHiddenAncestor

Returns `YES` if the receiver is marked as hidden or has an ancestor in the view hierarchy that is marked as hidden; returns `NO` otherwise.

```
- (BOOL)isHiddenOrHasHiddenAncestor
```

Discussion
The return value reflects state set through the `setHidden:` (page 101) method in the receiver of one of its ancestors in the view hierarchy. It does not account for other reasons why a view might be considered hidden, such as being positioned outside its superview's bounds, not having a window, or residing in a window that is offscreen or overlapped by another window.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- isHidden` (page 65)

Declared In
`NSView.h`

isInFullScreenMode

Returns a Boolean value that indicates whether the receiver is in full screen mode.

```
- (BOOL)isInFullScreenMode
```

Return Value
`YES` if the receiver is in full screen mode, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

isOpaque

Overridden by subclasses to return `YES` if the receiver is opaque, `NO` otherwise.

```
- (BOOL)isOpaque
```

Discussion
A view object is opaque if it completely covers its frame rectangle when drawing itself. The default implementation performs no drawing at all and so returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- opaqueAncestor` (page 75)
`- displayRectIgnoringOpacity:` (page 50)

- `displayIfNeededIgnoringOpacity` (page 49)
- `displayIfNeededInRectIgnoringOpacity:` (page 50)

Related Sample Code
GLChildWindowDemo
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TextEditPlus
WhackedTV

Declared In
`NSView.h`

isRotatedFromBase

Returns `YES` if the receiver or any of its ancestors has ever received a `setFrameRotation:` (page 100) or `setBoundsRotation:` (page 95) message; otherwise returns `NO`.

```
- (BOOL)isRotatedFromBase
```

Discussion
The intent of this information is to optimize drawing and coordinate calculation, not necessarily to reflect the exact state of the receiver's coordinate system, so it may not reflect the actual rotation. For example, if an `NSView` object is rotated to 45 degrees and later back to 0, this method still returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frameRotation` (page 60)
- `boundsRotation` (page 37)

Declared In
`NSView.h`

isRotatedOrScaledFromBase

Returns `YES` if the receiver or any of its ancestors has ever had a nonzero frame or bounds rotation, or has been scaled from the window's base coordinate system; otherwise returns `NO`.

```
- (BOOL)isRotatedOrScaledFromBase
```

Discussion
The intent of this information is to optimize drawing and coordinate calculation, not necessarily to reflect the exact state of the receiver's coordinate system, so it may not reflect the actual rotation or scaling. For example, if an `NSView` object is rotated to 45 degrees and later back to 0, this method still returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frameRotation` (page 60)

- `boundsRotation` (page 37)
- `centerScanRect:` (page 39)
- `setBounds:` (page 94)
- `setBoundsSize:` (page 96)
- `scaleUnitSquareToSize:` (page 89)

Declared In
`NSView.h`

knowsPageRange:

Returns `YES` if the receiver handles page boundaries, `NO` otherwise.

```
- (BOOL)knowsPageRange:(NSRangePointer)aRange
```

Parameters

`aRange`
  On return, holds the page range if `YES` is returned directly. Page numbers are one-based—that is pages run from one to *N*.

Discussion
Returns `NO` if the receiver uses the default auto-pagination mechanism. The default implementation returns `NO`. Override this method if your class handles page boundaries.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

layer

Returns the Core Animation layer that the receiver as its backing store.

```
- (CALayer *)layer
```

Return Value
The Core Animation layer the receiver is using as its backing store.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
CALayerEssentials
OpenGLCompositorLab

Declared In
`NSView.h`

locationOfPrintRect:

Invoked by `print:` (page 79) to determine the location of the region of the receiver being printed on the physical page.

```
- (NSPoint)locationOfPrintRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining a region of the receiver; it is expressed in the default coordinate system of the page.

Return Value

A point to be used for setting the origin for *aRect*, whose size the receiver can examine in order to properly place it. It is expressed in the default coordinate system of the page.

Discussion

The default implementation places *aRect* according to the status of the `NSPrintInfo` object for the print job. By default it places the image in the upper-left corner of the page, but if the `NSPrintInfo` methods `isHorizontallyCentered` or `isVerticallyCentered` return `YES`, it centers a single-page image along the appropriate axis. A multiple-page document, however, is always placed so the divided pieces can be assembled at their edges.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

lockFocus

Locks the focus on the receiver, so subsequent commands take effect in the receiver's window and coordinate system.

```
- (void)lockFocus
```

Discussion

If you don't use a `display...` method to draw an `NSView` object, you must invoke `lockFocus` before invoking methods that send commands to the window server, and must balance it with an `unlockFocus` (page 114) message when finished.

Hiding or miniaturizing a one-shot window causes the backing store for that window to be released. If you don't use the standard display mechanism to draw, you should use `lockFocusIfCanDraw` (page 70) rather than `lockFocus` if there is a chance of drawing while the window is either miniaturized or hidden.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ focusView` (page 25)
`- display` (page 48)
`- drawRect:` (page 55)
`- lockFocusIfCanDraw` (page 70)

Related Sample Code
MungSaver

QTKitFrameStepper

VideoViewer

Declared In
`NSView.h`

lockFocusIfCanDraw

Locks the focus to the receiver atomically if the `canDraw` method returns `YES` and returns the value of `canDraw`.

```
- (BOOL)lockFocusIfCanDraw
```

Discussion
Your thread will not be preempted by other threads between the `canDraw` method and the lock. This method fails to lock focus and returns `NO`, when the receiver is hidden and the current context is drawing to the screen (as opposed to a printing context).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
VideoViewer

Declared In
`NSView.h`

lockFocusIfCanDrawInContext:

Locks the focus to the receiver atomically if drawing can occur in the specified graphics context.

```
- (BOOL)lockFocusIfCanDrawInContext:(NSGraphicsContext *)context
```

Parameters

`context`
: The graphics context in which drawing might occur. See the discussion for the implications of the type of context.

Return Value
`YES` if successful; otherwise, returns `NO`.

Discussion
Your thread will not be preempted by other threads between the `canDraw` method and the lock.

If the *`context`* parameter represents the context for the window containing the view, then all of the necessary transformations are applied. This includes the application of the receiver's bounds and frame transforms along with any transforms it inherited from its ancestors. If *`context`* specifies any other graphics context, then only the receiver's bounds transform is applied.

Special Considerations

Important: This method was declared in Mac OS X v10.4, but is not used in that release. It currently does nothing and returns `NO`. However, it might be implemented in a future release.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- lockFocus` (page 69)
`- lockFocusIfCanDraw` (page 70)

Declared In
`NSView.h`

menuForEvent:

Overridden by subclasses to return a context-sensitive pop-up menu for a given mouse-down event.

```
- (NSMenu *)menuForEvent:(NSEvent *)theEvent
```

Parameters
theEvent
An object representing a mouse-down event.

Discussion
The receiver can use information in the mouse event, such as its location over a particular element of the receiver, to determine what kind of menu to return. For example, a text object might display a text-editing menu when the cursor lies over text and a menu for changing graphics attributes when the cursor lies over an embedded image.

The default implementation returns the receiver's normal menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ defaultMenu` (page 24)
`- menu` (`NSResponder`)

Declared In
`NSView.h`

mouse:inRect:

Returns whether a region of the receiver contains a specified point, accounting for whether the receiver is flipped or not.

```
- (BOOL)mouse:(NSPoint)aPoint inRect:(NSRect)aRect
```

Parameters

aPoint
: A point that is expressed in the receiver's coordinate system. This point generally represents the hot spot of the mouse cursor.

aRect
: A rectangle that is expressed in the receiver's coordinate system.

Return Value

`YES` if `aRect` contains `aPoint`, `NO` otherwise.

Discussion

Point-in-rectangle functions generally assume that the bottom edge of a rectangle is outside of the rectangle boundaries, while the upper edge is inside the boundaries. This method views *aRect* from the point of view of the user—that is, this method always treats the bottom edge of the rectangle as the one closest to the bottom edge of the user's screen. By making this adjustment, this function ensures consistent mouse-detection behavior from the user's perspective.

Never use the Foundation's `NSPointInRect` function as a substitute for this method. It doesn't account for flipped coordinate systems.

Availability

Available in Mac OS X v10.0 and later.

See Also

`– hitTest:` (page 63)

`– isFlipped` (page 65)

`NSMouseInRect` (Foundation functions)

`– convertPoint:fromView:` (page 40)

Declared In

`NSView.h`

mouseDownCanMoveWindow

Returns `YES` if the receiver does not need to handle a mouse down and can pass it through to superviews; `NO` if it needs to handle the mouse down.

```
- (BOOL)mouseDownCanMoveWindow
```

Discussion

This allows iApp-type applications to determine the region by which a window can be moved. By default, this method returns `NO` if the view is opaque; otherwise, it returns `YES`. Subclasses can override this method to return a different value.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

DockTile

SpeedometerView

WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In
`NSView.h`

needsDisplay

Returns `YES` if the receiver needs to be displayed, as indicated using `setNeedsDisplay:` (page 103) and `setNeedsDisplayInRect:` (page 103); returns `NO` otherwise.

```
- (BOOL)needsDisplay
```

Discussion
The `displayIfNeeded...` methods check this status to avoid unnecessary drawing, and all display methods clear this status to indicate that the view object is up to date.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

needsPanelToBecomeKey

Overridden by subclasses to return `YES` if the receiver requires its panel, which might otherwise avoid becoming key, to become the key window so that it can handle keyboard input.

```
- (BOOL)needsPanelToBecomeKey
```

Discussion
Such a subclass should also override `acceptsFirstResponder` to return `YES`. The default implementation returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- becomesKeyOnlyIfNeeded` (`NSPanel`)

Related Sample Code
Clock Control

Declared In
`NSView.h`

needsToDrawRect:

Returns whether the specified rectangle intersects any part of the area that the receiver is being asked to draw.

```
- (BOOL)needsToDrawRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining a region of the receiver.

Discussion

You typically send this message from within a `drawRect:` (page 55) implementation. It gives you a convenient way to determine whether any part of a given graphical entity might need to be drawn. It is optimized to efficiently reject any rectangle that lies outside the bounding box of the area the receiver is being asked to draw in `drawRect:`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSView.h`

nextKeyView

Returns the view object following the receiver in the key view loop, or `nil` if there is none.

```
- (NSView *)nextKeyView
```

Discussion

This view should, if possible, be made first responder when the user navigates forward from the receiver using keyboard interface control.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- nextValidKeyView` (page 74)
`- setNextKeyView:` (page 104)
`- previousKeyView` (page 78)
`- previousValidKeyView` (page 79)
`- selectNextKeyView:` (NSWindow)
`- selectKeyViewFollowingView:` (NSWindow)
`- selectPreviousKeyView:` (NSWindow)
`- selectKeyViewPrecedingView:` (NSWindow)

Related Sample Code

TrackBall

Declared In

`NSView.h`

nextValidKeyView

Returns the closest view object in the key view loop that follows the receiver and actually accepts first responder status, or `nil` if there is none.

```
- (NSView *)nextValidKeyView
```

Discussion
This method ignores hidden views when it determines the next valid key view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `nextKeyView` (page 74)
- `setNextKeyView:` (page 104)
- `previousKeyView` (page 78)
- `previousValidKeyView` (page 79)
- `selectNextKeyView:` (NSWindow)
- `selectKeyViewFollowingView:` (NSWindow)
- `selectPreviousKeyView:` (NSWindow)
- `selectKeyViewPrecedingView:` (NSWindow)
- `setHidden:` (page 101)

Declared In
`NSView.h`

opaqueAncestor

Returns the receiver's closest opaque ancestor (including the receiver itself).

```
- (NSView *)opaqueAncestor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isOpaque` (page 66)
- `displayRectIgnoringOpacity:` (page 50)
- `displayIfNeededIgnoringOpacity` (page 49)
- `displayIfNeededInRectIgnoringOpacity:` (page 50)

Declared In
`NSView.h`

pageFooter

Returns a default footer string that includes the current page number and page count.

```
- (NSAttributedString *)pageFooter
```

Discussion
A printable view class can override this method to substitute its own content in place of the default value. You should not need to call this method directly. The printing system calls it once per page during printing.

Footers are generated only if the user defaults contain the key `NSPrintHeaderAndFooter` with the value `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- pageHeader` (page 76)

Declared In
`NSView.h`

pageHeader

Returns a default header string that includes the print job title and date.

```
- (NSAttributedString *)pageHeader
```

Discussion
Typically, the print job title is the same as the window title. A printable view class can override this method to substitute its own content in place of the default value. You should not need to call this method directly. The printing system calls it once per page during printing.

Headers are generated only if the user defaults contain the key `NSPrintHeaderAndFooter` with the value `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- pageFooter` (page 75)

Declared In
`NSView.h`

performKeyEquivalent:

Implemented by subclasses to respond to key equivalents (also known as keyboard shortcuts).

```
- (BOOL)performKeyEquivalent:(NSEvent *)theEvent
```

Parameters

theEvent
The key-down event object representing a key equivalent.

Return Value
`YES` if *theEvent* is a key equivalent that the receiver handled, `NO` if it is not a key equivalent that it should handle.

Discussion
If the receiver's key equivalent is the same as the characters of the key-down event *theEvent*, as returned by `charactersIgnoringModifiers`, the receiver should take the appropriate action and return `YES`. Otherwise, it should return the result of invoking `super`'s implementation. The default implementation of this method simply passes the message down the view hierarchy (from superviews to subviews) and returns `NO` if none of the receiver's subviews responds `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- performMnemonic:` (page 77)
`- keyDown:` (NSWindow)

Declared In
`NSView.h`

performMnemonic:

Implemented by subclasses to respond to mnemonics.

```
- (BOOL)performMnemonic:(NSString *)aString
```

Parameters

aString
A string representing the mnemonic to handle.

Discussion
If the receiver's mnemonic is the same as the characters of the string *aString*, the receiver should take the appropriate action and return `YES`. Otherwise, it should return the result of invoking `super`'s implementation. The default implementation of this method simply passes the message down the view hierarchy (from superviews to subviews) and returns `NO` if none of the receiver's subviews responds `YES`. Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- performKeyEquivalent:` (page 76)
`- keyDown:` (NSWindow)

Declared In
`NSView.h`

postsBoundsChangedNotifications

Returns `YES` if the receiver posts notifications to the default notification center whenever its bounds rectangle changes; returns `NO` otherwise.

```
- (BOOL)postsBoundsChangedNotifications
```

Discussion
See `setPostsBoundsChangedNotifications:` (page 104) for a list of methods that result in notifications.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

postsFrameChangedNotifications

Returns `YES` if the receiver posts notifications to the default notification center whenever its frame rectangle changes; returns `NO` otherwise.

```
- (BOOL)postsFrameChangedNotifications
```

Discussion
See `setPostsBoundsChangedNotifications:` (page 104) for a list of methods that result in notifications.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

preservesContentDuringLiveResize

Returns `YES` if the view supports the optimization of live-resize operations by preserving content that has not moved; otherwise, returns `NO`.

```
- (BOOL)preservesContentDuringLiveResize
```

Discussion
The default is `NO`. If your view supports the content preservation feature, you should override this method and have your implementation return `YES`.

Content preservation lets your view decide what to redraw during a live resize operation. If your view supports this feature, you should also provide a custom implementation of `setFrameSize:` (page 101) that invalidates the portions of your view that actually need to be redrawn.

For information on how to implement this feature in your views, see *Cocoa Performance Guidelines*.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setFrameSize:` (page 101)

Declared In
`NSView.h`

previousKeyView

Returns the view object preceding the receiver in the key view loop, or `nil` if there is none.

```
- (NSView *)previousKeyView
```

Discussion
This view should, if possible, be made first responder when the user navigates backward from the receiver using keyboard interface control.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `previousValidKeyView` (page 79)
- `nextKeyView` (page 74)
- `nextValidKeyView` (page 74)
- `setNextKeyView:` (page 104)
- `selectNextKeyView:` (NSWindow)
- `selectKeyViewFollowingView:` (NSWindow)
- `selectPreviousKeyView:` (NSWindow)
- `selectKeyViewPrecedingView:` (NSWindow)

Declared In
`NSView.h`

previousValidKeyView

Returns the closest view object in the key view loop that precedes the receiver and actually accepts first responder status, or `nil` if there is none.

```
- (NSView *)previousValidKeyView
```

Discussion
This method ignores hidden views when it determines the previous valid key view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `previousKeyView` (page 78)
- `nextValidKeyView` (page 74)
- `nextKeyView` (page 74)
- `setNextKeyView:` (page 104)
- `selectNextKeyView:` (NSWindow)
- `selectKeyViewFollowingView:` (NSWindow)
- `selectPreviousKeyView:` (NSWindow)
- `selectKeyViewPrecedingView:` (NSWindow)
- `setHidden:` (page 101)

Declared In
`NSView.h`

print:

This action method opens the Print panel, and if the user chooses an option other than canceling, prints the receiver and all its subviews to the device specified in the Print panel.

```
- (void)print:(id)sender
```

Parameters

`sender`

The object that sent the message.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- dataWithEPSInsideRect:` (page 46)

`- writeEPSInsideRect:toPasteboard:` (page 123)

Declared In

`NSView.h`

printJobTitle

Returns the receiver's print job title.

```
- (NSString *)printJobTitle
```

Discussion

The default implementation first tries the window's `NSDocument` display name (`displayName`), then the window's title.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

rectForPage:

Implemented by subclasses to determine the portion of the receiver to be printed for the page number page.

```
- (NSRect)rectForPage:(NSInteger)pageNumber
```

Parameters

`pageNumber`

An integer indicating a page number. Page numbers are one-based—that is pages run from one to *N*.

Return Value

A rectangle defining the region of the receiver to be printed for *`pageNumber`*. This method returns `NSZeroRect` if *`pageNumber`* is outside the receiver's bounds.

Discussion

If the receiver responded `YES` to an earlier `knowsPageRange:` (page 68) message, this method is invoked for each page it specified in the out parameters of that message. The receiver is later made to display this rectangle in order to generate the image for this page.

If an `NSView` object responds `NO` to `knowsPageRange:` (page 68), this method isn't invoked by the printing mechanism.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– adjustPageHeightNew:top:bottom:limit:` (page 30)
`– adjustPageWidthNew:left:right:limit:` (page 31)

Declared In
`NSView.h`

rectPreservedDuringLiveResize

Returns the rectangle identifying the portion of your view that did not change during a live resize operation.

```
- (NSRect)rectPreservedDuringLiveResize
```

Discussion
The returned rectangle is in the coordinate system of your view and reflects the space your view previously occupied. This rectangle may be smaller or the same size as your view's current bounds, depending on whether the view grew or shrunk.

If your view does not support content preservation during live resizing, the returned rectangle will be empty. To support content preservation, override `preservesContentDuringLiveResize` (page 78) in your view and have your implementation return `YES`.

Note: The window containing your view must also support content preservation. To enable support for this feature in your window, use the `setPreservesContentDuringLiveResize:` method of `NSWindow`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– getRectsExposedDuringLiveResize:count:` (page 61)
`– preservesContentDuringLiveResize` (page 78)

Declared In
`NSView.h`

reflectScrolledClipView:

Notifies a clip view's superview that either the clip view's bounds rectangle or the document view's frame rectangle has changed, and that any indicators of the scroll position need to be adjusted.

```
- (void)reflectScrolledClipView:(NSClipView *)aClipView
```

Parameters

`aClipView`
The `NSClipView` object whose superview is to be notified.

Discussion
`NSScrollView` implements this method to update its `NSScroller` objects.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSClipView.h`

registeredDraggedTypes

Returns the array of pasteboard drag types that the view can accept.

```
- (NSArray *)registeredDraggedTypes
```

Discussion
This method returns the types registered by calling `registerForDraggedTypes:` (page 82). Each element of the array is a global string constant; see "Constants" for descriptions of the pasteboard string constants. The returned elements are in no particular order, but the array is guaranteed not to contain duplicate entries.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSView.h`

registerForDraggedTypes:

Registers the pasteboard types that the receiver will accept as the destination of an image-dragging session.

```
- (void)registerForDraggedTypes:(NSArray *)pboardTypes
```

Parameters

`pboardTypes`
An array of pasteboard types, each a global string constant. See "`Constants`" for descriptions of the pasteboard string constants.

Discussion
Registering an `NSView` object for dragged types automatically makes it a candidate destination object for a dragging session. As such, it must properly implement some or all of the `NSDraggingDestination` protocol methods. As a convenience, `NSView` provides default implementations of these methods. See the `NSDraggingDestination` protocol specification for details.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– registeredDraggedTypes` (page 82)
`– unregisterDraggedTypes` (page 114)

Declared In
`NSView.h`

releaseGState

Frees the receiver's graphics state object, if it has one.

```
- (void)releaseGState
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- allocateGState` (page 32)

Declared In
`NSView.h`

removeAllToolTips

Removes all tool tips assigned to the receiver.

```
- (void)removeAllToolTips
```

Discussion
This method operates on tool tips created using either `addToolTipRect:owner:userData:` (page 28) or `setToolTip:` (page 107).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

removeCursorRect:cursor:

Completely removes a cursor rectangle from the receiver.

```
- (void)removeCursorRect:(NSRect)aRect cursor:(NSCursor *)aCursor
```

Parameters

aRect
: A rectangle defining a region of the receiver. Must match a value previously specified using `addCursorRect:cursor:` (page 26).

aCursor
: An object representing a cursor. Must match a value previously specified using `addCursorRect:cursor:` (page 26).

Discussion
You should rarely need to use this method. `resetCursorRects` (page 87), which is invoked any time cursor rectangles need to be rebuilt, should establish only the cursor rectangles needed. If you implement `resetCursorRects` (page 87) in this way, you can then simply modify the state that `resetCursorRects` (page 87) uses to build its cursor rectangles and then invoke the `NSWindow` method `invalidateCursorRectsForView:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- discardCursorRects` (page 48)

Declared In
`NSView.h`

removeFromSuperview

Unlinks the receiver from its superview and its window, removes it from the responder chain, and invalidates its cursor rectangles.

```
- (void)removeFromSuperview
```

Discussion
The receiver is also released; if you plan to reuse it, be sure to retain it before sending this message and to release it as appropriate when adding it as a subview of another `NSView`.

Never invoke this method during display.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- addSubview:` (page 26)
`- addSubview:positioned:relativeTo:` (page 27)
`- removeFromSuperviewWithoutNeedingDisplay` (page 84)

Related Sample Code
FancyAbout
GLChildWindowDemo
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TextEditPlus

Declared In
`NSView.h`

removeFromSuperviewWithoutNeedingDisplay

Unlinks the receiver from its superview and its window and removes it from the responder chain, but does not invalidate its cursor rectangles to cause redrawing.

```
- (void)removeFromSuperviewWithoutNeedingDisplay
```

Discussion
The receiver is also released; if you plan to reuse it, be sure to retain it before sending this message and to release it as appropriate when adding it as a subview of another view.

Unlike its counterpart, `removeFromSuperview` (page 84), this method can be safely invoked during display.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- addSubview:` (page 26)
`- addSubview:positioned:relativeTo:` (page 27)

Related Sample Code
CoreRecipes

Declared In
`NSView.h`

removeToolTip:

Removes the tool tip identified by specified tag.

```
- (void)removeToolTip:(NSToolTipTag)tag
```

Parameters

`tag`
An integer tag that is the value returned by a previous `addToolTipRect:owner:userData:` (page 28) message.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

removeTrackingArea:

Removes a given tracking area from the receiver.

```
- (void)removeTrackingArea:(NSTrackingArea *)trackingArea
```

Parameters

`trackingArea`
The tracking area to remove from the receiver.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

removeTrackingRect:

Removes the tracking rectangle identified by a tag.

```
- (void)removeTrackingRect:(NSTrackingRectTag)aTag
```

Parameters

`aTag`

An integer value identifying a tracking rectangle. It was returned by a previously sent `addTrackingRect:owner:userData:assumeInside:` (page 29) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- addTrackingRect:owner:userData:assumeInside:` (page 29)

`- removeTrackingArea:` (page 85)

Declared In

`NSView.h`

renewGState

Invalidates the receiver's graphics state object, if it has one.

```
- (void)renewGState
```

Discussion

The receiver's graphics state object will be regenerated using `setUpGState` (page 107) the next time the receiver is focused for drawing

Availability

Available in Mac OS X v10.0 and later.

See Also

`- lockFocus` (page 69)

Related Sample Code

GLSLShowpiece

VideoViewer

Declared In

`NSView.h`

replaceSubview:with:

Replaces one of the receiver's subviews with another view.

```
- (void)replaceSubview:(NSView *)oldView with:(NSView *)newView
```

Parameters

`oldView`

The view to be replaced by `newView`. May not be `nil`.

`newView`

The view to replace `oldView`. May not be `nil`.

Discussion

This method does nothing if *`oldView`* is not a subview of the receiver.

Neither `oldView` nor `newView` may be `nil`, and the behavior is undefined if either of these parameters is `nil`.

This method causes `oldView` to be released; if you plan to reuse it, be sure to retain it before sending this message and to release it as appropriate when adding it as a subview of another NSView.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- addSubview:` (page 26)
`- addSubview:positioned:relativeTo:` (page 27)

Declared In
`NSView.h`

resetCursorRects

Overridden by subclasses to define their default cursor rectangles.

```
- (void)resetCursorRects
```

Discussion
A subclass's implementation must invoke `addCursorRect:cursor:` (page 26) for each cursor rectangle it wants to establish. The default implementation does nothing.

Application code should never invoke this method directly; it's invoked automatically as described in "Handling Tracking-Rectangle and Cursor-Update Events in Views." Use the `invalidateCursorRectsForView:` method instead to explicitly rebuild cursor rectangles.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- visibleRect` (page 120)

Related Sample Code
TextLinks

Declared In
`NSView.h`

resizeSubviewsWithOldSize:

Informs the receiver's subviews that the receiver's bounds rectangle size has changed.

```
- (void)resizeSubviewsWithOldSize:(NSSize)oldBoundsSize
```

Parameters

`oldBoundsSize`
The previous size of the receiver's bounds rectangle.

Discussion
If the receiver is configured to autoresize its subviews, this method is automatically invoked by any method that changes the receiver's frame size.

The default implementation sends `resizeWithOldSuperviewSize:` (page 88) to the receiver's subviews with *`oldBoundsSize`* as the argument. You shouldn't invoke this method directly, but you can override it to define a specific retiling behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAutoresizesSubviews:` (page 92)

Declared In
`NSView.h`

resizeWithOldSuperviewSize:

Informs the receiver that the bounds size of its superview has changed.

```
- (void)resizeWithOldSuperviewSize:(NSSize)oldBoundsSize
```

Parameters

`oldBoundsSize`
The previous size of the superview's bounds rectangle.

Discussion
This method is normally invoked automatically from `resizeSubviewsWithOldSize:` (page 87).

The default implementation resizes the receiver according to the autoresizing options listed under the `setAutoresizingMask:` (page 93) method description. You shouldn't invoke this method directly, but you can override it to define a specific resizing behavior.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

rotateByAngle:

Rotates the receiver's bounds rectangle by a specified degree value around the origin of the coordinate system, (0.0, 0.0).

```
- (void)rotateByAngle:(CGFloat)angle
```

Parameters

`angle`
A `float` value specifying the angle of rotation, in degrees.

Discussion
See the `setBoundsRotation:` (page 95) method description for more information. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewBoundsDidChangeNotification` (page 127) to the default notification center if the receiver is configured to do so.

Availability
Available in Mac OS X v10.0 and later.

See Also
– `setFrameRotation:` (page 100)
– `setPostsBoundsChangedNotifications:` (page 104)

Declared In
`NSView.h`

scaleUnitSquareToSize:

Scales the receiver's coordinate system so that the unit square scales to the specified dimensions.

```
- (void)scaleUnitSquareToSize:(NSSize)newUnitSize
```

Parameters

newUnitSize
An `NSSize` structure specifying the new unit size.

Discussion
For example, a *newUnitSize* of (0.5, 1.0) causes the receiver's horizontal coordinates to be halved, in turn doubling the width of its bounds rectangle. Note that scaling is performed from the origin of the coordinate system, (0.0, 0.0), not the origin of the bounds rectangle; as a result, both the origin and size of the bounds rectangle are changed. The frame rectangle remains unchanged.

This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewBoundsDidChangeNotification` (page 127) to the default notification center if the receiver is configured to do so.

Availability
Available in Mac OS X v10.0 and later.

See Also
– `setBoundsSize:` (page 96)
– `setPostsBoundsChangedNotifications:` (page 104)

Declared In
`NSView.h`

scrollClipView:toPoint:

Notifies the superview of a clip view that the clip view needs to reset the origin of its bounds rectangle.

```
- (void)scrollClipView:(NSClipView *)aClipView toPoint:(NSPoint)newOrigin
```

Parameters

aClipView
: The `NSClipView` object whose superview is to be notified.

newOrigin
: A point that specifies the new origin of the clip view's bounds rectangle.

Discussion

The superview of *aClipView* should then send a `scrollToPoint:` message to *aClipView* with *newOrigin* as the argument. This mechanism is provided so the `NSClipView` object's superview can coordinate scrolling of multiple tiled clip views.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- scrollToPoint:` (NSClipView)

Declared In

`NSClipView.h`

scrollPoint:

Scrolls the receiver's closest ancestor `NSClipView` object so a point in the receiver lies at the origin of the clip view's bounds rectangle.

```
- (void)scrollPoint:(NSPoint)aPoint
```

Parameters

aPoint
: The point in the receiver to scroll to.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- autoscroll:` (page 34)
`- scrollToPoint:` (`NSClipView`)
`- isDescendantOf:` (page 64)

Declared In

`NSView.h`

scrollRect:by:

Copies the visible portion of the receiver's rendered image within a region and lays that portion down again at a specified offset .

```
- (void)scrollRect:(NSRect)aRect by:(NSSize)offset
```

Parameters

aRect
A rectangle defining a region of the receiver.

offset
A `NSSize` structure that specifies an offset from from *aRect*'s origin.

Discussion
This method is useful during scrolling or translation of the coordinate system to efficiently move as much of the receiver's rendered image as possible without requiring it to be redrawn, following these steps:

1. Invoke `scrollRect:by:` (page 90) to copy the rendered image.

2. Move the view object's origin or scroll it within its superview.

3. Calculate the newly exposed rectangles and invoke either `setNeedsDisplay:` (page 103) or `setNeedsDisplayInRect:` (page 103) to draw them.

You should rarely need to use this method, however. The `scrollPoint:` (page 90), `scrollRectToVisible:` (page 91), and `autoscroll:` (page 34) methods automatically perform optimized scrolling.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBoundsOrigin:` (page 94)
- `translateOriginToPoint:` (page 112)

Declared In
`NSView.h`

scrollRectToVisible:

Scrolls the receiver's closest ancestor `NSClipView` object the minimum distance needed so a specified region of the receiver becomes visible in the clip view.

```
- (BOOL)scrollRectToVisible:(NSRect)aRect
```

Parameters

aRect
The rectangle to be made visible in the clip view.

Discussion
`YES` if any scrolling is performed; otherwise returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autoscroll:` (page 34)
- `scrollToPoint:` (NSClipView)
- `isDescendantOf:` (page 64)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In
`NSView.h`

setAlphaValue:

Sets the opacity of the receiver.

```
- (void)setAlphaValue:(CGFloat)viewAlpha
```

Parameters

`viewAlpha`
: The desired opacity of the receiver.

Discussion
This method sets the value of the `opacity` property of the receiver's layer. Possible values are between 0.0 (transparent) and 1.0 (opaque). The default is 1.0.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setAutoresizesSubviews:

Determines whether the receiver automatically resizes its subviews when its frame size changes.

```
- (void)setAutoresizesSubviews:(BOOL)flag
```

Parameters

`flag`
: If `YES`, the receiver invokes `resizeSubviewsWithOldSize:` (page 87) whenever its frame size changes; if `NO`, it doesn't.

Discussion
View objects do autoresize their subviews by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- autoresizesSubviews` (page 33)

Declared In
`NSView.h`

setAutoresizingMask:

Determines how the receiver's `resizeWithOldSuperviewSize:` (page 88) method changes its frame rectangle.

```
- (void)setAutoresizingMask:(NSUInteger)mask
```

Parameters

`mask`

An integer bit mask. *`mask`* can be specified by combining using the C bitwise OR operator any of the options described in "`Resizing masks`" (page 124).

Discussion

Where more than one option along an axis is set, `resizeWithOldSuperviewSize:` (page 88) by default distributes the size difference as evenly as possible among the flexible portions. For example, if `NSViewWidthSizable` and `NSViewMaxXMargin` are set and the superview's width has increased by 10.0 units, the receiver's frame and right margin are each widened by 5.0 units.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `autoresizingMask` (page 34)
- `resizeSubviewsWithOldSize:` (page 87)
- `setAutoresizesSubviews:` (page 92)

Related Sample Code

PredicateEditorSample

Quartz Composer QCTV

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

WhackedTV

Declared In

`NSView.h`

setBackgroundFilters:

An array of CoreImage filters that are applied to the receiver's background.

```
- (void)setBackgroundFilters:(NSArray *)filters
```

Parameters

`filters`

An array of CoreImage filters.

Discussion

This method sets the value of the `backgroundFilters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability

Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setBounds:

Sets the receiver's bounds rectangle.

```
- (void)setBounds:(NSRect)boundsRect
```

Parameters

boundsRect
A rectangle defining the new bounds of the receiver.

Discussion
The bounds rectangle determines the origin and scale of the receiver's coordinate system within its frame rectangle. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewBoundsDidChangeNotification` (page 127) to the default notification center if the receiver is configured to do so.

After calling this method, `NSView` creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `bounds` (page 36)
- `setBoundsRotation:` (page 95)
- `setBoundsOrigin:` (page 94)
- `setBoundsSize:` (page 96)
- `setFrame:` (page 98)
- `setPostsBoundsChangedNotifications:` (page 104)

Declared In
`NSView.h`

setBoundsOrigin:

Sets the origin of the receiver's bounds rectangle to a specified point,

```
- (void)setBoundsOrigin:(NSPoint)newOrigin
```

Parameters

newOrigin
A point specifying the new bounds origin of the receiver.

Discussion
In setting the new bounds origin, this method effectively shifts the receiver's coordinate system so *`newOrigin`* lies at the origin of the receiver's frame rectangle. It neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` or `setNeedsDisplay:` (page 103).

This method posts an `NSViewBoundsDidChangeNotification` (page 127) to the default notification center if the receiver is configured to do so.

After calling this method, `NSView` creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `translateOriginToPoint:` (page 112)
- `bounds` (page 36)
- `setBoundsRotation:` (page 95)
- `setBounds:` (page 94)
- `setBoundsSize:` (page 96)
- `setPostsBoundsChangedNotifications:` (page 104)

Related Sample Code
Polygons

Declared In
`NSView.h`

setBoundsRotation:

Sets the rotation of the receiver's bounds rectangle to a specific degree value.

```
- (void)setBoundsRotation:(CGFloat)angle
```

Parameters

`angle`
A `float` value specifying the angle of rotation, in degrees.

Discussion
Positive values indicate counterclockwise rotation, negative clockwise. Rotation is performed around the coordinate system origin, (0.0, 0.0), which need not coincide with that of the frame rectangle or the bounds rectangle. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewBoundsDidChangeNotification` (page 127) to the default notification center if the receiver is configured to do so.

Bounds rotation affects the orientation of the drawing within the view object's frame rectangle, but not the orientation of the frame rectangle itself. Also, for a rotated bounds rectangle to enclose all the visible areas of its view object—that is, to guarantee coverage over the frame rectangle—it must also contain some areas that aren't visible. This can cause unnecessary drawing to be requested, which may affect performance. It may be better in many cases to rotate the coordinate system in the `drawRect:` (page 55) method rather than use this method.

After calling this method, `NSView` creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rotateByAngle:` (page 88)
- `boundsRotation` (page 37)
- `setFrameRotation:` (page 100)
- `setPostsBoundsChangedNotifications:` (page 104)

Declared In
`NSView.h`

setBoundsSize:

Sets the size of the receiver's bounds rectangle to specified dimensions, inversely scaling its coordinate system relative to its frame rectangle.

```
- (void)setBoundsSize:(NSSize)newSize
```

Parameters

newSize
An `NSSize` structure specifying the new width and height of the receiver's bounds rectangle.

Discussion
For example, a view object with a frame size of (100.0, 100.0) and a bounds size of (200.0, 100.0) draws half as wide along the x axis. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewBoundsDidChangeNotification` (page 127) to the default notification center if the receiver is configured to do so.

After calling this method, `NSView` creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `bounds` (page 36)
- `setBoundsRotation:` (page 95)
- `setBounds:` (page 94)
- `setBoundsOrigin:` (page 94)

`- setPostsBoundsChangedNotifications:` (page 104)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSView.h`

setCompositingFilter:

Sets a CoreImage filter that is used to composite the receiver's contents with the background.

```
- (void)setCompositingFilter:(CIFilter *)filter
```

Parameters

filter
    A CoreImage filter.

Discussion
This method sets the value of the `compositingFilter` (page 40) property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setContentFilters:

Sets the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.

```
- (void)setContentFilters:(NSArray *)filters
```

Parameters

filters
    An array of CoreImage filters.

Discussion
This method sets the value of the `filters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setFocusRingType:

Sets the type of focus ring to be drawn around the receiver.

```
- (void)setFocusRingType:(NSFocusRingType)focusRingType
```

Parameters

focusRingType

An `enum` constant identifying a type of focus ring. Possible values are listed in `NSFocusRingType`. You can specify `NSFocusRingTypeNone` to indicate you do not want your view to have a focus ring.

Discussion

This method only sets the desired focus ring type and does not cause the view to draw the actual focus ring. You are responsible for drawing the focus ring in your view's drawRect: (page 55) method whenever your view is made the first responder.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- focusRingType` (page 59)

Declared In

`NSView.h`

setFrame:

Sets the receiver's frame rectangle to the specified rectangle.

```
- (void)setFrame:(NSRect)frameRect
```

Parameters

frameRect

The new frame rectangle for the view.

Discussion

This method, in setting the frame rectangle, repositions and resizes the receiver within the coordinate system of its superview. It neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewFrameDidChangeNotification` (page 128) to the default notification center if the receiver is configured to do so.

If your view does not use a custom bounds rectangle, this method also sets your view bounds to match the size of the new frame. You specify a custom bounds rectangle by calling `setBounds:` (page 94), `setBoundsOrigin:` (page 94), `setBoundsRotation:` (page 95), or `setBoundsSize:` (page 96)explicitly. Once set, NSView creates an internal transform to convert from frame coordinates to bounds coordinates. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- frame` (page 59)

- `setFrameRotation:` (page 100)
- `setFrameOrigin:` (page 99)
- `setFrameSize:` (page 101)
- `setPostsFrameChangedNotifications:` (page 105)

Related Sample Code

PDFKitLinker2

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

TipWrapper

Declared In

`NSView.h`

setFrameCenterRotation:

Rotates the frame of the receiver about the layer's position.

```
- (void)setFrameCenterRotation:(CGFloat)angle
```

Parameters

angle

The angle to rotate the frame around the center of the receiver.

Discussion

If the application has altered the layer's `anchorPoint` property, the behavior is undefined. Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSView.h`

setFrameOrigin:

Sets the origin of the receiver's frame rectangle to the specified point, effectively repositioning it within its superview.

```
- (void)setFrameOrigin:(NSPoint)newOrigin
```

Parameters

newOrigin

The point that is the new origin of the receiver's frame.

Discussion

This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewFrameDidChangeNotification` (page 128) to the default notification center if the receiver is configured to do so.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frame` (page 59)
- `setFrameSize:` (page 101)
- `setFrame:` (page 98)
- `setFrameRotation:` (page 100)
- `setPostsFrameChangedNotifications:` (page 105)

Related Sample Code
CoreRecipes
iSpend
Quartz Composer QCTV
Reducer
WhackedTV

Declared In
`NSView.h`

setFrameRotation:

Sets the rotation of the receiver's frame rectangle to a specified degree value, rotating it within its superview without affecting its coordinate system.

```
- (void)setFrameRotation:(CGFloat)angle
```

Parameters

angle
A `float` value indicating the degree of rotation.

Discussion
Positive values indicate counterclockwise rotation, negative clockwise. Rotation is performed around the origin of the frame rectangle.

This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewFrameDidChangeNotification` (page 128) to the default notification center if the receiver is configured to do so.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frameRotation` (page 60)
- `setBoundsRotation:` (page 95)

Declared In
`NSView.h`

setFrameSize:

Sets the size of the receiver's frame rectangle to the specified dimensions, resizing it within its superview without affecting its coordinate system.

```
- (void)setFrameSize:(NSSize)newSize
```

Parameters

newSize

An `NSSize` structure specifying the new height and width of the frame rectangle.

Discussion

This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

This method posts an `NSViewFrameDidChangeNotification` (page 128) to the default notification center if the receiver is configured to do so.

In Mac OS X version 10.4 and later, you can override this method to support content preservation during live resizing. In your overridden implementation, include some conditional code to be executed only during a live resize operation. Your code must invalidate any portions of your view that need to be redrawn.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `frame` (page 59)
- `setFrameOrigin:` (page 99)
- `setFrame:` (page 98)
- `setFrameRotation:` (page 100)
- `setPostsFrameChangedNotifications:` (page 105)

Related Sample Code

CoreRecipes

MyPhoto

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

WhackedTV

Declared In

`NSView.h`

setHidden:

Sets whether the receiver is hidden.

```
- (void)setHidden:(BOOL)flag
```

Parameters

flag

`YES` if the receiver is to be hidden, `NO` otherwise.

Discussion
A hidden view disappears from its window and does not receive input events. It remains in its superview's list of subviews, however, and participates in autoresizing as usual. The Application Kit also disables any cursor rectangle, tool-tip rectangle, or tracking rectangle associated with a hidden view. Hiding a view with subviews has the effect of hiding those subviews and any view descendants they might have. This effect is implicit and does not alter the hidden state of the receiver's descendants as reported by `isHidden` (page 65).

Hiding the view that is the window's current first responder causes the view's next valid key view (`nextValidKeyView` (page 74)) to become the new first responder. A hidden view remains in the `nextKeyView` (page 74) chain of views it was previously part of, but is ignored during keyboard navigation.

Availability
Available in Mac OS X v10.3 and later.

See Also
`– isHidden` (page 65)
`– isHiddenOrHasHiddenAncestor` (page 66)

Related Sample Code
CarbonCocoaCoreImageTab
iSpend
Reducer
StickiesExample

Declared In
`NSView.h`

setKeyboardFocusRingNeedsDisplayInRect:

Invalidates the area around the focus ring.

```
- (void)setKeyboardFocusRingNeedsDisplayInRect:(NSRect)rect
```

Parameters
rect
The rectangle of the control or cell defining the area around the focus ring. *rect* will be expanded to include the focus ring for invalidation.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`NSView.h`

setLayer:

Sets the Core Animation layer that the receiver uses for layer-backing to the specified layer.

```
- (void)setLayer:(CALayer *)newLayer
```

Parameters

newLayer

A Core Animation layer to use as the receiver's backing store.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSView.h`

setNeedsDisplay:

Controls whether the receiver's entire bounds is marked as needing display.

```
- (void)setNeedsDisplay:(BOOL)flag
```

Parameters

flag

If `YES`, marks the receiver's entire bounds as needing display; if `NO`, marks it as not needing display.

Discussion

Whenever the data or state used for drawing a view object changes, the view should be sent a setNeedsDisplay: message. `NSView` objects marked as needing display are automatically redisplayed on each pass through the application's event loop. (View objects that need to redisplay before the event loop comes around can of course immediately be sent the appropriate `display...` method.)

Availability

Available in Mac OS X v10.0 and later.

See Also

`– setNeedsDisplayInRect:` (page 103)

`– needsDisplay` (page 73)

Related Sample Code

Aperture Edit Plugin - Borders & Titles

Squiggles

WhackedTV

Declared In

`NSView.h`

setNeedsDisplayInRect:

Marks the region of the receiver within the specified rectangle as needing display, increasing the receiver's existing invalid region to include it.

```
- (void)setNeedsDisplayInRect:(NSRect)invalidRect
```

Parameters

invalidRect

The rectangular region of the receiver to mark as invalid; it should be specified in the coordinate system of the receiver.

Discussion
A later `displayIfNeeded...` method will then perform drawing only within the invalid region. View objects marked as needing display are automatically redisplayed on each pass through the application's event loop. (View objects that need to redisplay before the event loop comes around can of course immediately be sent the appropriate `display...` method.)

Availability
Available in Mac OS X v10.0 and later.

See Also
`– setNeedsDisplay:` (page 103)
`– needsDisplay` (page 73)

Related Sample Code
Reducer
Sketch-112
WhackedTV

Declared In
`NSView.h`

setNextKeyView:

Inserts a specified view object after the receiver in the key view loop of the receiver's window.

```
- (void)setNextKeyView:(NSView *)aView
```

Parameters

aView
  The `NSView` object to insert.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– nextKeyView` (page 74)
`– nextValidKeyView` (page 74)
`– previousKeyView` (page 78)
`– previousValidKeyView` (page 79)

Declared In
`NSView.h`

setPostsBoundsChangedNotifications:

Controls whether the receiver informs observers when its bounds rectangle changes.

```
- (void)setPostsBoundsChangedNotifications:(BOOL)flag
```

Parameters

`flag`

If `YES`, the receiver will post notifications to the default notification center whenever its bounds rectangle changes; if *`flag`* is `NO` it won't.

Discussion

Note that if *`flag`* is `YES` and bounds notifications are suppressed, when the bounds change notification is reenabled the view will immediately post a single such notification if its bounds changed during this time. This will happen even if there has been no net change in the view's bounds.

The following methods can result in notification posting:

- `setBounds:` (page 94)
- `setBoundsOrigin:` (page 94)
- `setBoundsRotation:` (page 95)
- `setBoundsSize:` (page 96)
- `translateOriginToPoint:` (page 112)
- `scaleUnitSquareToSize:` (page 89)
- `rotateByAngle:` (page 88)

Availability

Available in Mac OS X v10.0 and later.

See Also

`– postsBoundsChangedNotifications` (page 77)

Declared In

`NSView.h`

setPostsFrameChangedNotifications:

Controls whether the receiver informs observers when its frame rectangle changes.

```
- (void)setPostsFrameChangedNotifications:(BOOL)flag
```

Parameters

`flag`

If `YES`, the receiver will post notifications to the default notification center whenever its frame rectangle changes; if *`flag`* is `NO` it won't.

Discussion

Note that if *`flag`* is `YES` and frame notifications are suppressed, when the frame change notification is reenabled the view will immediately post a single such notification if its frame changed during this time. This will happen even if there has been no net change in the view's frame.

The following methods can result in notification posting:

- `setFrame:` (page 98)
- `setFrameOrigin:` (page 99)
- `setFrameRotation:` (page 100)
- `setFrameSize:` (page 101)

Availability
Available in Mac OS X v10.0 and later.

See Also
`- postsFrameChangedNotifications` (page 78)

Declared In
`NSView.h`

setShadow:

Sets the shadow drawn by the receiver.

```
- (void)setShadow:(NSShadow *)shadow
```

Parameters

`shadow`
    An instance of `NSShadow`.

Discussion
This method sets the `shadowColor`,`shadowOffset`, `shadowOpacity` and `shadowRadius` properties of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setSubviews:

Sets the receiver's subviews to the specified subviews.

```
- (void)setSubviews:(NSArray *)newSubviews
```

Parameters

`newSubviews`
    An array of subviews. The *`newSubviews`* array can consist of existing subviews of the receiver or other views that have `nil` as their superview. If *`newSubviews`* is `nil`, or contains duplicated views, or if any of its members have a superview other than `nil` or the reciever, an invalid argument exception is thrown.

Discussion
Using this method you can: reorder the receiver's existing subviews, add or remove subviews en masse, replace all of the receiver's subviews with a new set of subviews, or remove all the receiver's subviews.

Given a valid array of views in `newSubviews`, `setSubviews:` (page 106) performs any required sorting of the subviews array, as well as sending any `addSubview:` (page 26) and `removeFromSuperview` (page 84) messages as necessary to leave the receiver with the requested new array of subviews. Any member of `newSubviews` that isn't already a subview of the receiver is added. Any member of the view's existing `subviews` array that isn't in `newSubviews` is removed. And any views that are in both `subviews` (page 110) and `newSubviews` are moved in the subviews array as needed, without being removed and re-added.

This method marks the affected view and window areas as needing display.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setToolTip:

Sets the tool tip text for the view to *string*.

```
- (void)setToolTip:(NSString *)string
```

Parameters

string
A string that contains the text to use for the tool tip. If `nil`, it cancels tool tip display for the view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `toolTip` (page 112)

Declared In
`NSView.h`

setUpGState

Overridden by subclasses to (re)initialize the receiver's graphics state object.

```
- (void)setUpGState
```

Discussion
This method is automatically invoked when the graphics state object created using `allocateGState` (page 32) needs to be initialized. The default implementation does nothing. Your subclass can override it to set the current font, line width, or any other graphics state parameter except coordinate transformations and the clipping path—these are established by the frame and bounds rectangles and by methods such as `scaleUnitSquareToSize:` (page 89) and `translateOriginToPoint:` (page 112). Note that `drawRect:` can further transform the coordinate system and clipping path for whatever temporary effects it needs.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allocateGState` (page 32)
- `renewGState` (page 86)

Declared In
`NSView.h`

setWantsLayer:

Specifes whether the receiver and its subviews use a Core Animation layer as a backing store.

```
- (void)setWantsLayer:(BOOL)flag
```

Parameters

flag
: `YES` if the receiver and its subviews should use a Core Animation layer as its backing store, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

shadow

Returns the shadow drawn by the receiver

```
- (NSShadow *)shadow
```

Return Value
An instance of NSShadow that is created using the `shadowColor`,`shadowOffset`, `shadowOpacity` and `shadowRadius` properties of the receiver's layer.

Discussion
Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

shouldDelayWindowOrderingForEvent:

Overridden by subclasses to allow the user to drag images from the receiver without its window moving forward and possibly obscuring the destination and without activating the application.

```
- (BOOL)shouldDelayWindowOrderingForEvent:(NSEvent *)theEvent
```

Parameters

theEvent
: An object representing an initial mouse-down event.

Return Value
If this method returns `YES`, the normal window-ordering and activation mechanism is delayed (not necessarily prevented) until the next mouse-up event. If it returns `NO`, then normal ordering and activation occur.

Discussion
Never invoke this method directly; it's invoked automatically for each mouse-down event directed at the NSView.

An `NSView` subclass that allows dragging should implement this method to return `YES` if *`theEvent`* is potentially the beginning of a dragging session or of some other context where window ordering isn't appropriate. This method is invoked before a `mouseDown:` message for *`theEvent`* is sent. The default implementation returns `NO`.

If, after delaying window ordering, the receiver actually initiates a dragging session or similar operation, it should also send a `preventWindowOrdering` message to `NSApp`, which completely prevents the window from ordering forward and the activation from becoming active. `preventWindowOrdering` is sent automatically by the `NSView` `dragImage:...` and `dragFile:...` methods.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

shouldDrawColor

Returns `NO` if the receiver is being drawn in an `NSWindow` object (as opposed, for example, to being printed) and the window object can't store color; otherwise returns `YES`.

```
- (BOOL)shouldDrawColor
```

Discussion
A view object can base its drawing behavior on the return value of this method to improve its appearance in grayscale windows.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawRect:` (page 55)
- `canStoreColor` (NSWindow)

Declared In
`NSView.h`

sortSubviewsUsingFunction:context:

Orders the receiver's immediate subviews using the specified comparator function.

```
- (void)sortSubviewsUsingFunction:(NSComparisonResult (*)(id, id, void *))compare
    context:(void *)context
```

Parameters

`compare`

A pointer to the comparator function. This function must take as arguments two subviews to be ordered and contextual data (supplied in *`context`* which may be arbitrary data used to help in the comparison. The comparator function should return `NSOrderedAscending` if the first subview should be ordered lower, `NSOrderedDescending` if the second subview should be ordered lower, and `NSOrderedSame` if their ordering isn't important.

`context`

Arbitrary data that might help the comparator function `compare` in its decisions.

Availability

Available in Mac OS X v10.0 and later.

See Also

`sortedArrayUsingFunction:context:` (`NSArray`)

Declared In

`NSView.h`

subviews

Return the receiver's immediate subviews.

```
- (NSArray *)subviews
```

Return Value

Returns an array containing the receiver's subviews.

Discussion

The order of the subviews may be considered as being back-to-front, but this does not imply invalidation and drawing behavior. The order is based on the order of the receiver's subviews as specified in the nib file from which they were unarchived or the programmatic interface for modifying the receiver's subview list. This ordering is also the reverse of the order in which hit-testing is done.

Note: The contents of this array may change at any time. If you intend to manipulate this array you should copy it rather than simply retain.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- superview` (page 111)
`- addSubview:` (page 26)
`- addSubview:positioned:relativeTo:` (page 27)
`- removeFromSuperview` (page 84)

Related Sample Code

CarbonCocoaCoreImageTab

CoreRecipes

MyPhoto

Reducer

WhackedTV

Declared In
`NSView.h`

superview

Returns the receiver's superview, or `nil` if it has none.

```
- (NSView *)superview
```

Discussion
When applying this method iteratively or recursively, be sure to compare the returned view object to the content view of the window to avoid proceeding out of the view hierarchy.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- window` (page 122)
`- subviews` (page 110)
`- removeFromSuperview` (page 84)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
Reducer
TextEditPlus
WebKitDOMElementPlugIn

Declared In
`NSView.h`

tag

Returns the receiver's tag, an integer that you can use to identify view objects in your application.

```
- (NSInteger)tag
```

Discussion
The default implementation returns –1. Subclasses can override this method to provide individual tags, possibly adding storage and a `setTag:` method (which `NSView` doesn't define).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- viewWithTag:` (page 120)

Related Sample Code
CIVideoDemoGL
GLChildWindowDemo
Quartz2DBasics

Declared In
`NSView.h`

toolTip

Returns the text for the view's tool tip.

```
- (NSString *)toolTip
```

Return Value
The tool tip text or `nil` if the view doesn't currently display tool tip text

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setToolTip:` (page 107)

Declared In
`NSView.h`

trackingAreas

Returns an array of the receiver's tracking areas.

```
- (NSArray *)trackingAreas
```

Return Value
An array of the receiver's tracking areas (instances of `NSTrackingArea`). If the receiver has no tracking areas, returns an empty array.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
MenuItemView

Declared In
`NSView.h`

translateOriginToPoint:

Translates the receiver's coordinate system so that its origin moves to a new location.

```
- (void)translateOriginToPoint:(NSPoint)newOrigin
```

Parameters

`newOrigin`
A point that specifies the new origin.

Discussion
In the process, the origin of the receiver's bounds rectangle is shifted by (`-newOrigin.x`, `-newOrigin.y`). This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 48) or `setNeedsDisplay:` (page 103).

Note the difference between this method and setting the bounds origin. Translation effectively moves the image inside the bounds rectangle, while setting the bounds origin effectively moves the rectangle over the image. The two are in a sense inverse, although translation is cumulative, and setting the bounds origin is absolute.

This method posts an `NSViewBoundsDidChangeNotification` (page 127) to the default notification center if the receiver is configured to do so.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– setBoundsOrigin:` (page 94)
`– setBounds:` (page 94)
`– setPostsBoundsChangedNotifications:` (page 104)

Related Sample Code
Polygons

Declared In
`NSView.h`

translateRectsNeedingDisplayInRect:by:

Translates the display rectangles by the specified delta.

```
- (void)translateRectsNeedingDisplayInRect:(NSRect)clipRectby:(NSSize)delta
```

Parameters
`clipRect`
A rectangle defining the region of the receiver, typically the receiver's bounds.

`delta`
A NSSize structure that specifies an offset from from aRect's origin.

Discussion
The method is similar to the scrollRect:by: method:

1. It collects the receiving view's dirty rectangles.
2. Clears all dirty rectangles in the intersection of *`clipRect`* and the view's bounds.
3. Shifts the retrieved rectangles by the *`delta`* offset.
4. Clips the result to the intersection of *`clipRect`* and the view's bounds
5. Draws the resultant rectangles in the view.

The developer must ensure that *`clipRect`* and *`delta`* are pixel-aligned in order to guarantee correct drawing. See View Coordinate Conversion in *View Programming Guide for Cocoa* for a description of how to pixel-align view coordinates.

Note: In most cases developers should use the `scrollRect:by:` (page 90) method instead.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

unlockFocus

Balances an earlier `lockFocus` (page 69) or `lockFocusIfCanDraw` (page 70) message; restoring the focus to the previously focused view is necessary.

```
- (void)unlockFocus
```

Discussion
Raises an `NSInvalidArgumentException` if invoked on the wrong view.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– allocateGState` (page 32)

Related Sample Code
MungSaver
QTKitFrameStepper

Declared In
`NSView.h`

unregisterDraggedTypes

Unregisters the receiver as a possible destination in a dragging session.

```
- (void)unregisterDraggedTypes
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`– registerForDraggedTypes:` (page 82)

Declared In
`NSView.h`

updateTrackingAreas

Invoked automatically when the view's geometry changes such that its tracking areas need to be recalculated.

```
- (void)updateTrackingAreas
```

Discussion
You should override this method to remove out of date tracking areas and add recomputed tracking areas; your implementation should call `super`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

viewDidEndLiveResize

Informs the receiver of the end of a live resize.

```
- (void)viewDidEndLiveResize
```

Discussion
In the simple case, a view is sent `viewWillStartLiveResize` (page 119) before the first resize operation on the containing window and `viewDidEndLiveResize` after the last resize operation. A view that is repeatedly added and removed from a window during live resize will receive only one `viewWillStartLiveResize` (on the first time it is added to the window) and one `viewDidEndLiveResize` (when the window has completed the live resize operation). This allows a superview such as `NSBrowser` object to add and remove its `NSMatrix` subviews during live resize without the `NSMatrix` receiving multiple calls to these methods.

A view might allocate data structures to cache-drawing information in `viewWillStartLiveResize` (page 119) and should clean up these data structures in `viewDidEndLiveResize`. In addition, a view that does optimized drawing during live resize might want to do full drawing after `viewDidEndLiveResize`, although a view should not assume that it has a drawing context in `viewDidEndLiveResize` (since it may have been removed from the window during live resize). A view that wants to redraw itself after live resize should call `[self setNeedsDisplay:YES]` in `viewDidEndLiveResize`.

A view subclass should call `super` from these methods.

Availability
Available in Mac OS X v10.1 and later.

See Also
`- viewWillStartLiveResize` (page 119)
`- inLiveResize` (page 64)

Related Sample Code
MyPhoto

Declared In
`NSView.h`

viewDidHide

Invoked when the receiver is hidden, either directly, or in response to an ancestor being hidden.

```
- (void)viewDidHide
```

Discussion

The receiver receives this message when its `isHiddenOrHasHiddenAncestor` (page 66) state goes from `NO` to `YES`. This will happen when the view or an ancestor is marked as hidden, or when the view or an ancestor is inserted into a new view hierarchy.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSView.h`

viewDidMoveToSuperview

Informs the receiver that its superview has changed (possibly to `nil`).

```
- (void)viewDidMoveToSuperview
```

Discussion

The default implementation does nothing; subclasses can override this method to perform whatever actions are necessary.

Availability

Available in Mac OS X v10.0 and later.

See Also

`– viewDidMoveToWindow` (page 116)
`– viewWillMoveToSuperview:` (page 118)
`– viewWillMoveToWindow:` (page 118)

Declared In

`NSView.h`

viewDidMoveToWindow

Informs the receiver that it has been added to a new view hierarchy.

```
- (void)viewDidMoveToWindow
```

Discussion

The default implementation does nothing; subclasses can override this method to perform whatever actions are necessary.

`window` (page 122) may return `nil` when this method is invoked, indicating that the receiver does not currently reside in any window. This occurs when the receiver has just been removed from its superview or when the receiver has just been added to a superview that does not itself have a window. Overrides of this method may choose to ignore such cases if they are not of interest.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `viewDidMoveToSuperview` (page 116)
- `viewWillMoveToSuperview:` (page 118)
- `viewWillMoveToWindow:` (page 118)

Related Sample Code
Clock Control
Dicey
GLChildWindowDemo
SpotlightAPI
WhackedTV

Declared In
`NSView.h`

viewDidUnhide

Invoked when the receiver is unhidden, either directly, or in response to an ancestor being unhidden

```
- (void)viewDidUnhide
```

Discussion
The receiver receives this message when its isHiddenOrHasHiddenAncestor state goes from `YES` to `NO`. This can happen when the view or an ancestor is marked as not hidden, or when the view or an ancestor is removed from its containing view hierarchy.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

viewWillDraw

Informs the receiver that it will be required to draw content.

```
- (void)viewWillDraw
```

Discussion
In response to receiving one of the display... messages the receiver will recurse down the view hierarchy, sending this message to each of the views that may be involved in the display operation.

Subclasses can override this method to move or resize views, mark additional areas as requiring display, or other actions that can best be deferred until they are required for drawing. During the recursion, sending of `setNeedsDisplay:` (page 103) and `setNeedsDisplayInRect:` (page 103) messages to views in the hierarchy that's about to be drawn is valid and supported, and will affect the assessment of the total area to be rendered in that drawing pass.

A subclass's implementation of `viewWillDraw` can use the existing NSView `getRectsBeingDrawn:count:` (page 60) method to obtain a list of rectangles that bound the affected area, enabling it to restrict its efforts to that area.

The following is an example of a generic subclass implementation:

```
- (void)viewWillDraw {
    // Perform some operations before recursing for descendants.

    // Now recurse to handle all our descendants.
    // Overrides must call up to super like this.
    [super viewWillDraw];

    // Perform some operations that might depend on descendants
    //  already having had a chance to update.
}
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

viewWillMoveToSuperview:

Informs the receiver that its superview is about to change to the specified superview (which may be `nil`).

```
- (void)viewWillMoveToSuperview:(NSView *)newSuperview
```

Parameters

`newSuperview`
A view object that will be the new superview of the receiver.

Discussion
Subclasses can override this method to perform whatever actions are necessary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `viewDidMoveToSuperview` (page 116)
- `viewDidMoveToWindow` (page 116)
- `viewWillMoveToWindow:` (page 118)

Declared In
`NSView.h`

viewWillMoveToWindow:

Informs the receiver that it's being added to the view hierarchy of the specified window object (which may be `nil`).

```
- (void)viewWillMoveToWindow:(NSWindow *)newWindow
```

Parameters

newWindow

A window object that will be at the root of the receiver's new view hierarchy.

Discussion

Subclasses can override this method to perform whatever actions are necessary.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `viewDidMoveToSuperview` (page 116)
- `viewDidMoveToWindow` (page 116)
- `viewWillMoveToSuperview:` (page 118)

Declared In

`NSView.h`

viewWillStartLiveResize

Informs the receiver of the start of a live resize.

```
- (void)viewWillStartLiveResize
```

Discussion

In the simple case, a view is sent `viewWillStartLiveResize` before the first resize operation on the containing window and `viewDidEndLiveResize` (page 115) after the last resize operation. A view that is repeatedly added and removed from a window during live resize will receive only one `viewWillStartLiveResize` (on the first time it is added to the window) and one `viewDidEndLiveResize` (when the window has completed the live resize operation). This allows a superview such as `NSBrowser` object to add and remove its `NSMatrix` subviews during live resize without the `NSMatrix` object receiving multiple calls to these methods.

A view might allocate data structures to cache-drawing information in `viewWillStartLiveResize` and should clean up these data structures in `viewDidEndLiveResize` (page 115). In addition, a view that does optimized drawing during live resize might want to do full drawing after `viewDidEndLiveResize`, although a view should not assume that it has a drawing context in `viewDidEndLiveResize` (since it may have been removed from the window during live resize). A view that wants to redraw itself after live resize should call `[self setNeedsDisplay:YES]` in `viewDidEndLiveResize`.

A view subclass should call `super` from these methods.

Availability

Available in Mac OS X v10.1 and later.

See Also

- `viewDidEndLiveResize` (page 115)
- `inLiveResize` (page 64)

Related Sample Code

MyPhoto

Declared In

`NSView.h`

viewWithTag:

Returns the receiver's nearest descendant (including itself) with a specific tag, or `nil` if no subview has that tag.

```
- (id)viewWithTag:(NSInteger)aTag
```

Parameters

aTag

An integer identifier associated with a view object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- tag` (page 111)

Related Sample Code

PrefsPane

Declared In

`NSView.h`

visibleRect

Returns the portion of the receiver not clipped by its superviews.

```
- (NSRect)visibleRect
```

Return Value

A rectangle defining the unclipped portion of the receiver.

Discussion

Visibility for this method is defined quite simply and doesn't account for whether other `NSView` objects (or windows) overlap the receiver or whether the receiver has a window at all. This method returns `NSZeroRect` if the receiver is effectively hidden.

During a printing operation the visible rectangle is further clipped to the page being imaged.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setHidden:` (page 101)

`- isVisible` (NSWindow)

`- documentVisibleRect` (NSScrollView)

`- documentVisibleRect` (NSClipView)

Related Sample Code

GLChildWindowDemo

LiveVideoMixer3

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

TextLinks

Declared In
`NSView.h`

wantsDefaultClipping

Returns whether the Application Kit's default clipping provided to `drawRect:` (page 55) implementations is in effect.

```
- (BOOL)wantsDefaultClipping
```

Return Value
`YES` if the default clipping is in effect, `NO` otherwise. By default, this method returns `YES`.

Discussion
Subclasses may override this method to return `NO` if they want to suppress the default clipping. They may want to do this in situations where drawing performance is critical to avoid the cost of setting up, enforcing, and cleaning up the clip path

A view that overrides this method to refuse the default clipping must either set up whatever clipping it requires or constrain its drawing exactly to the list of rectangles returned by `getRectsBeingDrawn:count:` (page 60). Failing to do so could result in corruption of other drawing in the view's window.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
VideoViewer

Declared In
`NSView.h`

wantsLayer

Returns a Boolean value that indicates whether the receiver is using a layer as it's backing store.

```
- (BOOL)wantsLayer
```

Return Value
`YES` if the receiver is using a Core Animation layer as its backing store, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

widthAdjustLimit

Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as small images or text columns from being divided across pages.

```
- (CGFloat)widthAdjustLimit
```

Discussion
This fraction is used to calculate the right edge limit for a `adjustPageWidthNew:left:right:limit:` (page 31) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- heightAdjustLimit` (page 62)

Declared In
`NSView.h`

willRemoveSubview:

Overridden by subclasses to perform additional actions before subviews are removed from the receiver.

```
- (void)willRemoveSubview:(NSView *)subview
```

Parameters

subview
The subview that will be removed.

Discussion
This method is invoked when *subview* receives a `removeFromSuperview` (page 84) message or *subview* is removed from the receiver due to it being added to another view with `addSubview:` (page 26).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

window

Returns the receiver's window object, or `nil` if it has none.

```
- (NSWindow *)window
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- superview` (page 111)

Related Sample Code
Dicey
GLChildWindowDemo
QTCoreVideo201
Sketch-112
WhackedTV

Declared In
`NSView.h`

writeEPSInsideRect:toPasteboard:

Writes EPS data that draws the region of the receiver within a specified rectangle onto a pasteboard.

```
- (void)writeEPSInsideRect:(NSRect)aRect toPasteboard:(NSPasteboard *)pboard
```

Parameters

aRect
A rectangle defining the region.

pboard
An object representing a pasteboard.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– dataWithEPSInsideRect:` (page 46)

Declared In
`NSView.h`

writePDFInsideRect:toPasteboard:

Writes PDF data that draws the region of the receiver within a specified rectangle onto a pasteboard.

```
- (void)writePDFInsideRect:(NSRect)aRect toPasteboard:(NSPasteboard *)pboard
```

Parameters

aRect
A rectangle defining the region.

pboard
An object representing a pasteboard.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– dataWithPDFInsideRect:` (page 47)

Declared In
`NSView.h`

Constants

NSBorderType

These constants specify the type of a view's border.

```
typedef enum _NSBorderType {
    NSNoBorder     = 0,
    NSLineBorder   = 1,
    NSBezelBorder  = 2,
    NSGrooveBorder = 3
} NSBorderType;
```

Constants

`NSBezelBorder`

A concave border that makes the view look sunken.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSGrooveBorder`

A thin border that looks etched around the image.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSLineBorder`

A black line border around the view.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSNoBorder`

No border.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

Resizing masks

These constants are used by `setAutoresizingMask:` (page 93).

```
enum {
   NSViewNotSizable     = 0,
   NSViewMinXMargin     = 1,
   NSViewWidthSizable   = 2,
   NSViewMaxXMargin     = 4,
   NSViewMinYMargin     = 8,
   NSViewHeightSizable  = 16,
   NSViewMaxYMargin     = 32
};
```

Constants

`NSViewNotSizable`
: The receiver cannot be resized.

  Available in Mac OS X v10.0 and later.

  Declared in `NSView.h`.

`NSViewMinXMargin`
: The left margin between the receiver and its superview is flexible.

  Available in Mac OS X v10.0 and later.

  Declared in `NSView.h`.

`NSViewWidthSizable`
: The receiver's width is flexible.

  Available in Mac OS X v10.0 and later.

  Declared in `NSView.h`.

`NSViewMaxXMargin`
: The right margin between the receiver and its superview is flexible.

  Available in Mac OS X v10.0 and later.

  Declared in `NSView.h`.

`NSViewMinYMargin`
: The bottom margin between the receiver and its superview is flexible.

  Available in Mac OS X v10.0 and later.

  Declared in `NSView.h`.

`NSViewHeightSizable`
: The receiver's height is flexible.

  Available in Mac OS X v10.0 and later.

  Declared in `NSView.h`.

`NSViewMaxYMargin`
: The top margin between the receiver and its superview is flexible.

  Available in Mac OS X v10.0 and later.

  Declared in `NSView.h`.

Declared In

`NSView.h`

NSToolTipTag

This type describes the rectangle used to identify a tool tip rectangle.

```
typedef NSInteger NSToolTipTag;
```

Discussion
See the methods `addToolTipRect:owner:userData:` (page 28) and`removeToolTip:` (page 85).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

NSTrackingRectTag

This type describes the rectangle used to track the mouse.

```
typedef NSInteger NSTrackingRectTag;
```

Discussion
See the methods `addTrackingRect:owner:userData:assumeInside:` (page 29) and `removeTrackingRect:` (page 85).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

Full screen mode

These constants are keys that you can use in the options dictionary in `enterFullScreenMode:withOptions:` (page 58) and `exitFullScreenModeWithOptions:` (page 58).

```
NSString * const NSFullScreenModeAllScreens;
NSString * const NSFullScreenModeSetting;
NSString * const NSFullScreenModeWindowLevel;
```

Constants

`NSFullScreenModeAllScreens`

Key whose corresponding value specifies whether the view should take over all screens.

The corresponding value is an instance of `NSNumber` containing a Boolean value.

Available in Mac OS X v10.5 and later.

Declared in `NSView.h`.

`NSFullScreenModeSetting`

Key whose corresponding value specifies the the full screen mode setting.

The corresponding value is an instance of `NSDictionary` that contains keys specified in `Display Mode Standard Properties` and `Display Mode Optional Properties` in *Quartz Display Services Reference*.

Available in Mac OS X v10.5 and later.

Declared in `NSView.h`.

`NSFullScreenModeWindowLevel`

Key whose corresponding value specifies the screen mode window level.

The corresponding value is an instance of `NSNumber` containing an integer value.

Available in Mac OS X v10.5 and later.

Declared in `NSView.h`.

Declared In
`NSView.h`

Notifications

NSViewBoundsDidChangeNotification

Posted whenever the `NSView`'s bounds rectangle changes independently of the frame rectangle, if the `NSView` is configured using `setPostsBoundsChangedNotifications:` (page 104) to post such notifications.

The notification object is the `NSView` object whose bounds rectangle has changed. This notification does not contain a *`userInfo`* dictionary.

The following methods can result in notification posting:

`setBounds:` (page 94)
`setBoundsOrigin:` (page 94)
`setBoundsRotation:` (page 95)
`setBoundsSize:` (page 96)
`translateOriginToPoint:` (page 112)
`scaleUnitSquareToSize:` (page 89)
`rotateByAngle:` (page 88)

Note that the bounds rectangle resizes automatically to track the frame rectangle. Because the primary change is that of the frame rectangle, however, `setFrame:` (page 98) and `setFrameSize:` (page 101) don't result in a bounds-changed notification.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

NSViewFocusDidChangeNotification

Deprecated notification that was posted for an `NSView` object and each of its descendants (recursively) whenever the frame or bounds geometry of the view changed.

In Mac OS X v10.4 and later, this notification is no longer posted. In earlier version of Mac OS X, use `NSViewBoundsDidChangeNotification` and `NSViewFrameDidChangeNotification` instead to get the same information provided by this notification.

The notification object is the view whose geometry changed. This notification does not contain a *`userInfo`* dictionary.

Availability
Deprecated in Mac OS X v10.4 and later.

See Also
`NSViewBoundsDidChangeNotification` (page 127)
`NSViewFrameDidChangeNotification` (page 128)

Declared In
`NSView.h`

NSViewFrameDidChangeNotification

Posted whenever the view's frame rectangle changes, if the view is configured using `setPostsFrameChangedNotifications:` (page 105) to post such notifications.

The notification object is the `NSView` object whose frame rectangle has changed. This notification does not contain a *`userInfo`* dictionary.

The following methods can result in notification posting:

- `setFrame:` (page 98)
- `setFrameOrigin:` (page 99)
- `setFrameRotation:` (page 100)
- `setFrameSize:` (page 101)

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

NSViewDidUpdateTrackingAreasNotification

Posted whenever an `NSView` object recalculates its tracking areas. It is sent after the view receives `updateTrackingAreas` (page 115).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

NSViewGlobalFrameDidChangeNotification

Posted whenever an `NSView` object that has attached surfaces (that is, `NSOpenGLContext` objects) moves to a different screen, or other cases where the `NSOpenGLContext` object needs to be updated. The notification object is the surface's view. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`NSView.h`

Document Revision History

This table describes the changes to *NSView Class Reference*.

Date	Notes
2009-02-04	Corrected typos.
2009-01-06	Updated description of the bounds method to reflect its behavior when drawing the view in an NSOpenGLContext.
2008-10-15	TBD
2008-03-11	Corrected information about the isOpaque method that's found in the discussion of the drawRect: method.
2007-10-31	Corrected typos.
2007-07-24	Added new API introduced in Mac OS X v10.5. Clarified discardCursorRects. Improved description of final parameter of addTrackingRect:owner:userData:assumeInside:
2006-05-23	First publication of this content as a separate document.

Index