
NSImage Class Reference

[Cocoa > Graphics & Imaging](#)



2009-01-06



Apple Inc.
© 2009 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Bonjour, Carbon, Cocoa, iChat, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Aperture, Finder, and Shuffle are trademarks of Apple Inc.

Adobe, Acrobat, and PostScript are trademarks or registered trademarks of Adobe Systems Incorporated in the U.S. and/or other countries.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSImage Class Reference 9

Overview	9
Adopted Protocols	9
Tasks	10
Initializing a New NSImage Object	10
Setting the Image Attributes	10
Referring to Images by Name	10
Determining the Supported Image Types	11
Working With Image Representations	11
Setting the Image Representation Selection Criteria	11
Managing the Focus	12
Drawing the Image	12
Working With Alignment Metadata	13
Setting the Image Storage Options	13
Setting the Image Drawing Options	13
Assigning a Delegate	14
Producing TIFF Data for the Image	14
Managing Incremental Loads	14
Class Methods	15
canInitWithPasteboard:	15
imageFileTypes	15
imageName:	16
imagePasteboardTypes	17
imageTypes	18
imageUnfilteredFileTypes	18
imageUnfilteredPasteboardTypes	19
imageUnfilteredTypes	19
Instance Methods	20
addRepresentation:	20
addRepresentations:	20
alignmentRect	21
backgroundColor	21
bestRepresentationForDevice:	22
cacheDepthMatchesImageDepth	22
cacheMode	23
cancelIncrementalLoad	23
compositeToPoint:fromRect:operation:	23
compositeToPoint:fromRect:operation:fraction:	25
compositeToPoint:operation:	25
compositeToPoint:operation:fraction:	26
delegate	27

dissolveToPoint:fraction: 27
dissolveToPoint:fromRect:fraction: 28
drawAtPoint:fromRect:operation:fraction: 29
drawInRect:fromRect:operation:fraction: 30
drawRepresentation:inRect: 30
initWithReferencingFile: 31
initWithReferencingURL: 32
initWithContentsOfFile: 33
initWithContentsOfURL: 33
initWithData: 34
initWithIconRef: 34
initWithPasteboard: 34
initWithSize: 35
isCachedSeparately 36
isDataRetained 36
isFlipped 37
isTemplate 37
isValid 37
lockFocus 38
lockFocusOnRepresentation: 39
matchesOnMultipleResolution 39
name 40
prefersColorMatch 40
recache 41
removeRepresentation: 41
representations 41
scalesWhenResized 42
setAlignmentRect: 42
setBackgroundColor: 43
setCacheDepthMatchesImageDepth: 43
setCachedSeparately: 44
setCacheMode: 44
setDataRetained: 45
setDelegate: 45
setFlipped: 46
setMatchesOnMultipleResolution: 47
setName: 47
setPrefersColorMatch: 48
setScalesWhenResized: 48
setSize: 49
setTemplate: 50
setUsesEPSOnResolutionMismatch: 50
size 51
TIFFRepresentation 51
TIFFRepresentationUsingCompression:factor: 52
unlockFocus 53

- usesEPSOnResolutionMismatch 53
- Delegate Methods 54
 - image:didLoadPartOfRepresentation:withValidRows: 54
 - image:didLoadRepresentation:withStatus: 54
 - image:didLoadRepresentationHeader: 55
 - image:willLoadRepresentation: 55
 - imageDidNotDraw:inRect: 56
- Constants 56
 - NSCompositingOperation 56
 - NSImageLoadStatus 59
 - NSImageCacheMode 60
 - Button Template Images 61
 - Multiple Documents Drag Image 65
 - Sharing Permissions Named Images 66
 - System Entity Images 67
 - Toolbar Named Images 68
 - View Type Template Images 69

Document Revision History 71

Index 73

Tables

NSImage Class Reference 9

Table 1	Default pasteboard types for image representations	35
Table 2	Placeholder values for compositing equations	59

NSImage Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Cocoa Drawing Guide
Declared in	NSGraphics.h NSImage.h
Related sample code	ImageBackground MyPhoto QTKitMovieShuffler RGB Image Sketch-112

Overview

An `NSImage` object is a high-level class for manipulating image data. You use this class to load existing images or create new ones and composite them into a view or other image. This class works in conjunction with one or more image representation objects (subclasses of `NSImageRep`), which manage the actual image data.

Adopted Protocols

NSCoding

`encodeWithCoder:`
`initWithCoder:`

NSCopying

`copyWithZone:`

Tasks

Initializing a New NSImage Object

- [initWithReferencingFile:](#) (page 31)
Initializes and returns an `NSImage` instance and associates it with the specified file.
- [initWithReferencingURL:](#) (page 32)
Initializes and returns an `NSImage` instance and associates it with the specified URL.
- [initWithContentsOfFile:](#) (page 33)
Initializes and returns an `NSImage` instance with the contents of the specified file.
- [initWithContentsOfURL:](#) (page 33)
Initializes and returns an `NSImage` instance with the contents of the specified URL.
- [initWithData:](#) (page 34)
Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object.
- [initWithPasteboard:](#) (page 34)
Initializes and returns an `NSImage` instance with data from the specified pasteboard.
- [initWithSize:](#) (page 35)
Initializes and returns an `NSImage` instance whose size is set to the specified value.
- [initWithIconRef:](#) (page 34)
Initializes the image object with a Carbon-style icon resource.

Setting the Image Attributes

- [setSize:](#) (page 49)
Sets the width and height of the image.
- [size](#) (page 51)
Returns the size of the receiver.
- [isTemplate](#) (page 37)
Returns a Boolean value indicating whether the image is a template image.
- [setTemplate:](#) (page 50)
Sets whether the image represents a template image.

Referring to Images by Name

- + [imageName:](#) (page 16)
Returns the `NSImage` instance associated with the specified name.
- [setName:](#) (page 47)
Registers the receiver under the specified name.
- [name](#) (page 40)
Returns the name associated with the receiver, if any.

Determining the Supported Image Types

- + [canInitWithPasteboard:](#) (page 15)
Tests whether the receiver can create an instance of itself using pasteboard data.
- + [imageTypes](#) (page 18)
Returns an array of UTI strings identifying the image types supported by the registered `NSImageRep` objects, either directly or through a user-installed filter service.
- + [imageUnfilteredTypes](#) (page 19)
Returns an array of UTI strings identifying the image types supported directly by the registered `NSImageRep` objects.
- + [imageFileTypes](#) (page 15)
Returns an array of strings identifying the image types supported by the registered `NSImageRep` objects.
- + [imageUnfilteredFileTypes](#) (page 18)
Returns an array of strings identifying the file types supported directly by the registered `NSImageRep` objects.
- + [imagePasteboardTypes](#) (page 17)
Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.
- + [imageUnfilteredPasteboardTypes](#) (page 19)
Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.

Working With Image Representations

- [addRepresentation:](#) (page 20)
Adds the specified image representation object to to the receiver.
- [addRepresentations:](#) (page 20)
Adds an array of image representation objects to the receiver.
- [bestRepresentationForDevice:](#) (page 22)
Returns the best representation for the device with the specified characteristics.
- [representations](#) (page 41)
Returns an array containing all of the receiver's image representations.
- [removeRepresentation:](#) (page 41)
Removes the specified image representation from the receiver and releases it.

Setting the Image Representation Selection Criteria

- [setPrefersColorMatch:](#) (page 48)
Sets whether choosing an image representation favors color matching over resolution matching.
- [prefersColorMatch](#) (page 40)
Returns a Boolean value indicating whether the image prefers to choose image representations using color matching or resolution matching.

- [setUsesEPSOnResolutionMismatch:](#) (page 50)
Sets whether EPS image representations are preferred when no other representations match the resolution of the device.
- [usesEPSOnResolutionMismatch](#) (page 53)
Returns a Boolean value indicating whether EPS representations are preferred when no other representations match the resolution of the device.
- [setMatchesOnMultipleResolution:](#) (page 47)
Sets whether image representations whose resolutions are integral multiples of the device resolution are considered a match.
- [matchesOnMultipleResolution](#) (page 39)
Returns a Boolean value indicating whether image representations whose resolution is an integral multiple of the device resolution are considered a match.

Managing the Focus

- [lockFocus](#) (page 38)
Prepares the image to receive drawing commands.
- [lockFocusOnRepresentation:](#) (page 39)
Prepares the specified image representation to receive drawing commands.
- [unlockFocus](#) (page 53)
Removes the focus from the receiver.

Drawing the Image

- [drawAtPoint:fromRect:operation:fraction:](#) (page 29)
Draws all or part of the image at the specified point in the current coordinate system.
- [drawInRect:fromRect:operation:fraction:](#) (page 30)
Draws all or part of the image in the specified rectangle in the current coordinate system.
- [drawRepresentation:inRect:](#) (page 30)
Draws the image using the specified image representation object.
- [compositeToPoint:operation:](#) (page 25)
Composites the entire image to the specified point in the current coordinate system.
- [compositeToPoint:fromRect:operation:](#) (page 23)
Composites a portion of the image to the specified point in the current coordinate system.
- [compositeToPoint:fromRect:operation:fraction:](#) (page 25)
Composites a portion of the image at the specified opacity to the current coordinate system.
- [compositeToPoint:operation:fraction:](#) (page 26)
Composites the entire image at the specified opacity in the current coordinate system.
- [dissolveToPoint:fraction:](#) (page 27)
Composites the entire image to the specified location using the `NSCompositeSourceOver` operator.
- [dissolveToPoint:fromRect:fraction:](#) (page 28)
Composites a portion of the image to the specified location using the `NSCompositeSourceOver` operator.

- [imageDidNotDraw:inRect:](#) (page 56) *delegate method*
Sent to the delegate when the image object is unable, for whatever reason, to lock focus on its image or draw in the specified rectangle.

Working With Alignment Metadata

- [alignmentRect](#) (page 21)
Returns alignment metadata that your code can use to position the image during layout.
- [setAlignmentRect:](#) (page 42)
Sets the alignment metadata that your code can use to position the image during layout.

Setting the Image Storage Options

- [setCachedSeparately:](#) (page 44)
Sets whether each image representation uses a separate offscreen window to cache its contents.
- [isCachedSeparately](#) (page 36)
Returns a Boolean value indicating whether each image representation caches its contents in a separate offscreen window.
- [setDataRetained:](#) (page 45)
Sets whether the receiver retains its source image data.
- [isDataRetained](#) (page 36)
Returns a Boolean value indicating whether the receiver retains its source image data.
- [setCacheDepthMatchesImageDepth:](#) (page 43)
Sets whether the receiver's offscreen window caches use the same bit depth as the image data itself.
- [cacheDepthMatchesImageDepth](#) (page 22)
Returns a Boolean value indicating whether an image's offscreen window caches use the same bit depth as the image data itself.
- [cacheMode](#) (page 23)
Returns the receiver's caching mode.
- [setCacheMode:](#) (page 44)
Set the receiver's caching mode.

Setting the Image Drawing Options

- [isValid](#) (page 37)
Returns a Boolean value indicating whether an image representation from the receiver can be drawn.
- [setScaleWhenResized:](#) (page 48)
Sets whether different-sized image representations are scaled to fit the receiver's size.
- [scalesWhenResized](#) (page 42)
Returns a Boolean value indicating whether image representations are scaled to fit the receiver's size.
- [setBackgroundcolor:](#) (page 43)
Sets the background color of the image.

- [backgroundColor](#) (page 21)
Returns the background color of image.
- [setFlipped:](#) (page 46)
Sets whether the polarity of the y axis is inverted when drawing an image.
- [isFlipped](#) (page 37)
Returns a Boolean value indicating whether the image uses a flipped coordinate system.
- [recache](#) (page 41)
Invalidates and frees the offscreen caches of all image representations.

Assigning a Delegate

- [setDelegate:](#) (page 45)
Sets the delegate object of the receiver.
- [delegate](#) (page 27)
Returns the delegate object of the receiver

Producing TIFF Data for the Image

- [TIFFRepresentation](#) (page 51)
Returns a data object containing TIFF data for all of the image representations in the receiver.
- [TIFFRepresentationUsingCompression:factor:](#) (page 52)
Returns a data object containing TIFF data with the specified compression settings for all of the image representations in the receiver.

Managing Incremental Loads

- [cancelIncrementalLoad](#) (page 23)
Cancels the current download operation immediately, if the image is being incrementally loaded.
- [image:didLoadPartOfRepresentation:withValidRows:](#) (page 54) *delegate method*
During incremental loading, this method is called repeatedly to inform the delegate that more of the image data is available.
- [image:didLoadRepresentation:withStatus:](#) (page 54) *delegate method*
For incremental loading, this method is invoked when the specified image has been loaded and decompressed as fully as is possible.
- [image:didLoadRepresentationHeader:](#) (page 55) *delegate method*
During incremental loading, this method is called once enough data has been read to determine the size of the image.
- [image:willLoadRepresentation:](#) (page 55) *delegate method*
For incremental loading, this method is invoked when you first attempt to draw the image or otherwise access the bitmap data.

Class Methods

canInitWithPasteboard:

Tests whether the receiver can create an instance of itself using pasteboard data.

```
+ (BOOL)canInitWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

The pasteboard containing the image data.

Return Value

YES if the receiver knows how to handle the data on the pasteboard; otherwise, NO.

Discussion

This method uses the `NSImageRep` class method `imageUnfilteredPasteboardTypes` to find a class that can handle the data in the specified pasteboard. If you create your own `NSImageRep` subclasses, override the `imageUnfilteredPasteboardTypes` method to notify `NSImage` of the pasteboard types your class supports.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [imagePasteboardTypes](#) (page 17)

Related Sample Code

CocoaDragAndDrop

Declared In

`NSImage.h`

imageFileTypes

Returns an array of strings identifying the image types supported by the registered `NSImageRep` objects.

```
+ (NSArray *)imageFileTypes
```

Return Value

An array of `NSString` objects, each of which identifies a single supported file type. The array can include encoded HFS file types as well as filename extensions.

Discussion

This list includes all file types supported by registered subclasses of `NSImageRep` plus those that can be converted to a supported type by a user-installed filter service. You can pass the array returned by this method directly to the `runModalForTypes:` method of `NSOpenPanel`.

When creating a subclass of `NSImageRep`, do not override this method. Instead, override the `imageUnfilteredFileTypes` method to notify `NSImage` of the file types your class supports directly.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [imageUnfilteredFileTypes](#) (page 18)

Related Sample Code

DeskPictAppDockMenu

TrackBall

Declared In

NSImage.h

imageName:

Returns the `NSImage` instance associated with the specified name.

```
+ (id)imageName:(NSString *)name
```

Parameters

name

The name associated with the desired image.

Return Value

The `NSImage` object associated with the specified name, or `nil` if no such image was found.

Discussion

This method searches for named images in several places, returning the first image it finds matching the given name. The order of the search is as follows:

1. Search for an object whose name was set explicitly using the `setName:` method and currently resides in the image cache.
2. Search the application's main bundle for a file whose name matches the specified string. (For information on how the bundle is searched, see "Searching for Bundle Resources" in *Bundle Programming Guide*.)
3. Search the Application Kit framework for a shared image with the specified name.

When looking for files in the application bundle, it is better (but not required) to include the filename extension in the `name` parameter. When naming an image with the `setName:` method, it is also convention not to include filename extensions in the names you specify. That way, you can easily distinguish between images you have named explicitly and those you want to load from the application's bundle.

One particularly useful image you can retrieve is your application's icon. This image is set by Cocoa automatically and referenced by the string `@NSApplicationIcon`. Icons for other applications can be obtained through the use of methods declared in the `NSWorkspace` class. You can also retrieve some standard system images using Cocoa defined constants; for more information, see the Constants section of this class.

If an application is linked in Mac OS X v10.5 or later, images requested using this method and whose name ends in the word "Template" are automatically marked as template images.

The `NSImage` class may cache a reference to the returned image object for performance in some cases. However, the class holds onto cached objects only while the object exists. If the image object is subsequently released, either because its retain count was 0 or it was not referenced anywhere in a garbage-collected application, the object may be quietly removed from the cache. Thus, if you plan to hold onto a returned image object, you must retain it like you would any Cocoa object. You can clear an image object from the cache explicitly by calling the object's `setName:` method and passing `nil` for the image name.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setName:](#) (page 47)
- [name](#) (page 40)
- [iconForFile:](#) (NSWorkspace)
- + [imageFileTypes](#) (page 15)

Related Sample Code

Dicey
EnhancedDataBurn
GridCalendar
ImageMapExample
iSpend

Declared In

NSImage.h

imagePasteboardTypes

Returns an array of strings identifying the pasteboard types supported directly by the registered NSImageRep objects.

```
+ (NSArray *)imagePasteboardTypes
```

Return Value

An array of NSString objects, each of which identifies a single supported pasteboard type. By default, this list contains the NSPDFPboardType, NSPICTPboardType, NSPostScriptPboardType, and NSTIFFPboardType types.

Discussion

This list includes all pasteboard types supported by registered subclasses of NSImageRep plus those that can be converted to a supported type by a user-installed filter service.

When creating a subclass of NSImageRep, do not override this method. Instead, override the `imageUnfilteredPasteboardTypes` method to notify NSImage of the pasteboard types your class supports.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [imageUnfilteredPasteboardTypes](#) (page 19)

Related Sample Code

CocoaDragAndDrop
GLChildWindowDemo
Sketch-112

Declared In

NSImage.h

imageTypes

Returns an array of UTI strings identifying the image types supported by the registered `NSImageRep` objects, either directly or through a user-installed filter service.

```
+ (NSArray *)imageTypes
```

Return Value

An array of `NSString` objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include `"public.image"`, `"public.jpeg"`, and `"public.tiff"`. For a list of supported types, see `UTCoreTypes.h`.

Discussion

The returned list includes UTIs all file types supported by registered subclasses of `NSImageRep` plus those that can be converted to a supported type by a user-installed filter service. You can use the returned UTI strings with any method that supports UTIs.

You should not override this method directly. Instead, you should override the `imageTypes` method of `NSImageRep`.

Availability

Available in Mac OS X v10.5 and later.

See Also

+ [imageUnfilteredTypes](#) (page 19)

Declared In

`NSImage.h`

imageUnfilteredFileTypes

Returns an array of strings identifying the file types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imageUnfilteredFileTypes
```

Return Value

An array of `NSString` objects, each of which identifies a single supported file type. File types are identified by file extension and HFS file types.

Discussion

The returned list does not contain pasteboard types that are available only through a user-installed filter service.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [imageFileTypes](#) (page 15)

Declared In

`NSImage.h`

imageUnfilteredPasteboardTypes

Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imageUnfilteredPasteboardTypes
```

Return Value

An array of `NSString` objects, each of which identifies a single supported pasteboard type.

Discussion

The returned list does not contain pasteboard types that are supported only through a user-installed filter service.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [imagePasteboardTypes](#) (page 17)

Declared In

`NSImage.h`

imageUnfilteredTypes

Returns an array of UTI strings identifying the image types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imageUnfilteredTypes
```

Return Value

An array of `NSString` objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include `"public.image"`, `"public.jpeg"`, and `"public.tiff"`. For a list of supported types, see `UTCoreTypes.h`.

Discussion

The returned list includes UTI strings only for those file types that are supported directly by registered subclasses of `NSImageRep`. It does not include types that are supported through user-installed filter services. You can use the returned UTI strings with any method that supports UTIs.

You should not override this method directly. Instead, you should override the `imageUnfilteredTypes` method of `NSImageRep`.

Availability

Available in Mac OS X v10.5 and later.

See Also

+ [imageTypes](#) (page 18)

Declared In

`NSImage.h`

Instance Methods

addRepresentation:

Adds the specified image representation object to to the receiver.

```
- (void)addRepresentation:(NSImageRep *)imageRep
```

Parameters

imageRep

The image representation to add.

Discussion

After invoking this method, you may need to explicitly set features of the new image representation, such as the size, number of colors, and so on. This is true particularly when the `NSImage` object has multiple image representations to choose from. See `NSImageRep` and its subclasses for the methods you use to complete initialization.

Any representation added by this method is retained by the receiver. Image representations cannot be shared among multiple `NSImage` objects.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [representations](#) (page 41)
- [removeRepresentation:](#) (page 41)

Related Sample Code

ColorMatching

Image Difference

Monochrome Image

PDFView

Reducer

Declared In

`NSImage.h`

addRepresentations:

Adds an array of image representation objects to the receiver.

```
- (void)addRepresentations:(NSArray *)imageReps
```

Parameters

imageReps

An array of `NSImageRep` objects.

Discussion

After invoking this method, you may need to explicitly set features of the new image representations, such as their size, number of colors, and so on. This is true particularly when the `NSImage` object has multiple image representations to choose from. See `NSImageRep` and its subclasses for the methods you use to complete initialization.

Representations added by this method are retained by the receiver. Image representations cannot be shared among multiple `NSImage` objects.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [representations](#) (page 41)
- [removeRepresentation:](#) (page 41)

Declared In

`NSImage.h`

alignmentRect

Returns alignment metadata that your code can use to position the image during layout.

- `(NSRect)alignmentRect`

Return Value

A rectangle containing the layout information for the image. If not set, the returned rectangle has an origin of (0, 0) and a size that matches the size of the image.

Discussion

The returned rectangle is merely a hint that your own code can use to determine positioning. The `NSImage` class does not use this rectangle during drawing. However, instances of `NSCell` typically use this information when laying out images within their own boundaries.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setAlignmentRect:](#) (page 42)

Declared In

`NSImage.h`

backgroundColor

Returns the background color of image.

- `(NSColor *)backgroundColor`

Return Value

The background color of the image. The default color is transparent, as returned by the `clearColor` method of `NSColor`.

Discussion

The background color is visible only if the drawn image representation does not completely cover all of the pixels available for the image's current size.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSImage.h

bestRepresentationForDevice:

Returns the best representation for the device with the specified characteristics.

```
- (NSImageRep *)bestRepresentationForDevice:(NSDictionary *)deviceDescription
```

Parameters

deviceDescription

A dictionary of attributes for the specified device, or `nil` to specify the current device. For a list of dictionary keys and values appropriate to display and print devices, see the constants in `NSScreen`.

Return Value

The image representation that most closely matches the specified criteria.

Discussion

If *deviceDescription* is `nil`, this method uses the attributes of the device on which the content is to be drawn.

For information on how the "best" representation is chosen, see the Images chapter of *Cocoa Drawing Guide*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [representations](#) (page 41)
- [prefersColorMatch](#) (page 40)
- [matchesOnMultipleResolution](#) (page 39)
- [usesEPSOnResolutionMismatch](#) (page 53)

Related Sample Code

LayerBackedOpenGLView

NSOpenGL Fullscreen

PDF Annotation Editor

Sketch-112

Declared In

NSImage.h

cacheDepthMatchesImageDepth

Returns a Boolean value indicating whether an image's offscreen window caches use the same bit depth as the image data itself.

- (BOOL)cacheDepthMatchesImageDepth

Return Value

YES if the offscreen window caches use the same bit depth as the image data; otherwise, NO. The default value is NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setCacheDepthMatchesImageDepth:](#) (page 43)

Declared In

NSImage.h

cacheMode

Returns the receiver's caching mode.

- (NSImageCacheMode)cacheMode

Return Value

A value indicating the caching mode. For a list of possible values, see [NSImageCacheMode](#) (page 60). This value is set to `NSImageCacheDefault` by default.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [setCacheMode:](#) (page 44)

Declared In

NSImage.h

cancelIncrementalLoad

Cancels the current download operation immediately, if the image is being incrementally loaded.

- (void)cancelIncrementalLoad

Discussion

This call has no effect if the image is not loading.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSImage.h

compositeToPoint:fromRect:operation:

Composites a portion of the image to the specified point in the current coordinate system.

```
- (void)compositeToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op
```

Parameters

aPoint

The point at which to draw the image, specified in the current coordinate system.

srcRect

The portion of the image you want to draw, specified in the image's coordinate system.

op

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in [“Constants”](#) (page 56).

Discussion

This method draws the specified portion of the image without checking the bounds rectangle you pass into the *srcRect* parameter. If you specify a source rectangle that strays outside of the image's bounds rectangle, it is conceivable that you could composite parts of the offscreen cache window that do not belong to the receiver's image. You can avoid this problem by using the [setCachedSeparately:](#) (page 44) method to force the image to be cached in its own offscreen window, which results in the content being clipped to the window rectangle. Alternatively, you can use the [drawAtPoint:fromRect:operation:fraction:](#) (page 29) method, which checks the source rectangle before drawing.

During drawing, the image is composited from its offscreen window cache. Because the offscreen cache is not created until the image representation is first used, this method may need to render the image before compositing. Bitmap representations in particular are not cached until they are explicitly rendered. You can use the [lockFocus](#) (page 38) and [unlockFocus](#) (page 53) methods to force the cached version to be created.

Compositing part of an image is as efficient as compositing the whole image, but printing just part of an image is not. When printing, it's necessary to draw the whole image and rely on a clipping path to be sure that only the desired portion appears.

During printing, this method ignores the *op* parameter. Even though this parameter is ignored, this method attempts to render the image as close as possible to its appearance when the compositing operation is used on the screen. In either case, the best image representation is chosen for the printing context.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the [drawAtPoint:fromRect:operation:fraction:](#) or [drawInRect:fromRect:operation:fraction:](#) method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dissolveToPoint:fromRect:fraction:](#) (page 28)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 29)
- [drawInRect:fromRect:operation:fraction:](#) (page 30)

Declared In

NSImage.h

compositeToPoint:fromRect:operation:fraction:

Composites a portion of the image at the specified opacity to the current coordinate system.

```
- (void)compositeToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect  
    operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

aPoint

The point at which to draw the image, specified in the current coordinate system.

srcRect

The portion of the image you want to draw, specified in the image's coordinate system.

op

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in “Constants” (page 56).

delta

The desired opacity of the image, specified as a value between 0.0 and 1.0, with 1.0 representing total opacity. Values larger than 1.0 are interpreted as 1.0. This method always expects to render something, so for values that are equal to or less than 0, this method renders at full opacity.

Discussion

Behaves the same as [compositeToPoint:fromRect:operation:](#) (page 23) except that you can specify the amount of opacity to use when drawing the image.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the [drawAtPoint:fromRect:operation:fraction:](#) or [drawInRect:fromRect:operation:fraction:](#) method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dissolveToPoint:fromRect:fraction:](#) (page 28)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 29)
- [drawInRect:fromRect:operation:fraction:](#) (page 30)

Declared In

NSImage.h

compositeToPoint:operation:

Composites the entire image to the specified point in the current coordinate system.

```
- (void)compositeToPoint:(NSPoint)aPoint operation:(NSCompositingOperation)op
```

Parameters

aPoint

The point at which to draw the image, specified in the current coordinate system.

op

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in “Constants” (page 56).

Discussion

This method draws the receiver's best image representation at the specified point in the currently focused view. The entire image is drawn using its current size information. During drawing, the image is composited from its offscreen window cache. Because the offscreen cache is not created until the image representation is first used, this method may need to render the image before compositing. Bitmap representations in particular are not cached until they are explicitly rendered. You can use the [lockFocus](#) (page 38) and [unlockFocus](#) (page 53) methods to force the cached version to be created.

During printing, this method ignores the *op* parameter. Even though this parameter is ignored, this method attempts to render the image as close as possible to its appearance when the compositing operation is used on the screen. In either case, the best image representation is chosen for the printing context.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dissolveToPoint:fraction:](#) (page 27)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 29)
- [drawInRect:fromRect:operation:fraction:](#) (page 30)

Related Sample Code

ColorMatching
EnhancedAudioBurn
Image Difference
RGB ValueTransformers
Sketch-112

Declared In

NSImage.h

compositeToPoint:operation:fraction:

Composites the entire image at the specified opacity in the current coordinate system.

```
- (void)compositeToPoint:(NSPoint)aPoint operation:(NSCompositingOperation)op
    fraction:(CGFloat)delta
```

Parameters

aPoint

The point at which to draw the image, specified in the current coordinate system.

op

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in “Constants” (page 56).

delta

The desired opacity of the image, specified as a value between 0.0 and 1.0, with 1.0 representing total opacity. Values larger than 1.0 are interpreted as 1.0. This method always expects to render something, so for values that are equal to or less than 0, this method renders at full opacity.

Discussion

Behaves the same as `compositeToPoint:operation:` (page 25) except that you can specify the amount of opacity to use when drawing the image.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dissolveToPoint:fraction:](#) (page 27)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 29)
- [drawInRect:fromRect:operation:fraction:](#) (page 30)

Declared In

NSImage.h

delegate

Returns the delegate object of the receiver

- (id)delegate

Return Value

The current delegate object, or `nil` if no delegate has been set.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setDelegate:](#) (page 45)

Declared In

NSImage.h

dissolveToPoint:fraction:

Composites the entire image to the specified location using the `NSCompositeSourceOver` operator.

- (void)dissolveToPoint:(NSPoint)aPoint fraction:(CGFloat)delta

Parameters*aPoint*

The point at which to draw the image, specified in the current coordinate system.

delta

The desired opacity of the image, specified as a value between 0.0 and 1.0. A value of 0.0 renders the image totally transparent while 1.0 renders it fully opaque. Values larger than 1.0 are interpreted as 1.0.

Discussion

Except for the choice of compositing operator, this method behaves in the same way as the [compositeToPoint:operation:](#) (page 25) method. During printing, the *delta* parameter is ignored.

If the source image contains alpha information, this operation may promote the destination `NSWindow` object to contain alpha information.

To slowly dissolve this image onto another, you can invoke this method (or the [dissolveToPoint:fromRect:fraction:](#) (page 28) method) repeatedly with an ever-increasing *delta* value. Because the *delta* parameter refers to the visible fraction of the source image, increasing the value causes the source image to replace the destination content gradually. You should generally perform this type of operation using a buffered window or other offscreen drawing environment.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSImage.h`

dissolveToPoint:fromRect:fraction:

Composites a portion of the image to the specified location using the `NSCompositeSourceOver` operator.

```
- (void)dissolveToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect
    fraction:(CGFloat)delta
```

Parameters*aPoint*

The point at which to draw the image, specified in the current coordinate system.

srcRect

The portion of the image you want to draw, specified in the image's coordinate system.

delta

The desired opacity of the image, specified as a value between 0.0 and 1.0. A value of 0.0 renders the image totally transparent while 1.0 renders it fully opaque. Values larger than 1.0 are interpreted as 1.0.

Discussion

Except for the choice of compositing operator, this method behaves in the same way as the [compositeToPoint:fromRect:operation:](#) (page 23) method. During printing, the *delta* parameter is ignored.

If the source image contains alpha information, this operation may promote the destination `NSWindow` object to contain alpha information.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dissolveToPoint:fraction:](#) (page 27)

Declared In

NSImage.h

drawAtPoint:fromRect:operation:fraction:

Draws all or part of the image at the specified point in the current coordinate system.

```
(void)drawAtPoint:(NSPoint)point fromRect:(NSRect)srcRect
      operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

point

The location in the current coordinate system at which to draw the image.

srcRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle are specified in the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

op

The compositing operation to use when drawing the image. See the [NSCompositingOperation](#) (page 56) constants.

delta

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

Discussion

The image content is drawn at its current resolution and is not scaled unless the CTM of the current coordinate system itself contains a scaling factor. The image is otherwise positioned and oriented using the current coordinate system.

Unlike the [compositeToPoint:fromRect:operation:](#) (page 23) and [compositeToPoint:fromRect:operation:fraction:](#) (page 25) methods, this method checks the rectangle you pass to the `srcRect` parameter and makes sure it does not lie outside the image bounds.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dissolveToPoint:fraction:](#) (page 27)

- [drawInRect:fromRect:operation:fraction:](#) (page 30)

Related Sample Code

Reducer

Declared In

NSImage.h

drawInRect:fromRect:operation:fraction:

Draws all or part of the image in the specified rectangle in the current coordinate system.

```
- (void)drawInRect:(NSRect)dstRect fromRect:(NSRect)srcRect
  operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

dstRect

The rectangle in which to draw the image, specified in the current coordinate system.

srcRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

op

The compositing operation to use when drawing the image. See the [NSCompositingOperation](#) (page 56) constants.

delta

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

Discussion

If the `srcRect` and `dstRect` rectangles have different sizes, the source portion of the image is scaled to fit the specified destination rectangle. The image is otherwise positioned and oriented using the current coordinate system.

Unlike the [compositeToPoint:fromRect:operation:](#) (page 23) and [compositeToPoint:fromRect:operation:fraction:](#) (page 25) methods, this method checks the rectangle you pass to the `srcRect` parameter and makes sure it does not lie outside the image bounds.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dissolveToPoint:fraction:](#) (page 27)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 29)

Related Sample Code

Clock Control
 CocoaVideoFrameToNSImage
 Transformed Image
 WebKitDOMElementPlugIn

Declared In

NSImage.h

drawRepresentation:inRect:

Draws the image using the specified image representation object.

```
- (BOOL)drawRepresentation:(NSImageRep *)imageRep inRect:(NSRect)dstRect
```

Parameters*imageRep*

The image representation object to be drawn.

dstRect

The rectangle in which to draw the image representation, specified in the current coordinate system.

Return Value

YES if the image was successfully drawn; otherwise, returns NO.

Discussion

This method fills the specified rectangle with the image's current background color and then sends a message to the specified image representation asking if to draw itself. If the image supports the ability to scale itself when it is resized, this method sends a `drawInRect:` message; otherwise, it sends a `drawAtPoint:` message.

You should not call this method directly; an `NSImage` object uses it to cache and print its image representations. You can override this method to change the way images are rendered into their caches and onto the printed page. For example, you could scale or rotate the coordinate system before sending this message to `super` to continue rendering the image representation.

If the background color is fully transparent and the image data is not being cached, the specified rectangle is not to be filled before the representation draws.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [backgroundColor](#) (page 21)

Declared In

`NSImage.h`

initWithReferencingFile:

Initializes and returns an `NSImage` instance and associates it with the specified file.

```
- (id)initWithReferencingFile:(NSString *)filename
```

Parameters*filename*

A full or relative path name specifying the file with the desired image data. Relative paths must be relative to the current working directory.

Return Value

An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

Discussion

This method initializes the image object lazily. It does not actually open the specified file or create any image representations from its data until an application attempts to draw the image or request information about it.

The *filename* parameter should include the file extension that identifies the type of the image data. The mechanism that actually creates the image representation for *filename* looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Because this method doesn't actually create image representations for the image data, your application should do error checking before attempting to use the image; one way to do so is by invoking the [isValid](#) (page 37) method to check whether the image can be drawn.

This method invokes [setDataRetained:](#) (page 45) with an argument of `YES`, thus enabling it to hold onto its filename. When archiving an image created with this method, only the image's filename is written to the archive.

If the cached version of the image uses less memory than the original image data, the original data is flushed and the cached image is used. (This can occur for images whose resolution is greater than 72 dpi.) If you resize the image by less than 50%, the data is loaded in again from the file. If you expect the file to change or be deleted, you should use [initWithContentsOfFile:](#) (page 33) instead.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Clock Control

PictureTaker

Declared In

NSImage.h

initWithReferencingURL:

Initializes and returns an `NSImage` instance and associates it with the specified URL.

```
- (id)initWithReferencingURL:(NSURL *)url
```

Parameters

url

The URL identifying the image.

Return Value

An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

Discussion

This method initializes the image object lazily. It does not attempt to retrieve the data from the specified URL or create any image representations from that data until an application attempts to draw the image or request information about it.

This *url* parameter should include a file extension that identifies the type of the image data. The mechanism that actually creates the image representation looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Because this method doesn't actually create image representations for the image data, your application should do error checking before attempting to use the image; one way to do so is by invoking the [isValid](#) (page 37) method to check whether the image can be drawn.

This method invokes [setDataRetained:](#) (page 45) with an argument of `YES`, thus enabling it to hold onto its URL. When archiving an image created with this method, only the image's URL is written to the archive.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

TrackBall

Declared In

NSImage.h

initWithContentsOfFile:

Initializes and returns an `NSImage` instance with the contents of the specified file.

```
- (id)initWithContentsOfFile:(NSString *)filename
```

Parameters

filename

A full or relative path name specifying the file with the desired image data. Relative paths must be relative to the current working directory.

Return Value

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified file.

Discussion

Unlike [initWithReferencingFile:](#) (page 31), which initializes an `NSImage` object lazily, this method immediately opens the specified file and creates one or more image representations from its data.

The *filename* parameter should include the file extension that identifies the type of the image data. This method looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSImage.h

initWithContentsOfURL:

Initializes and returns an `NSImage` instance with the contents of the specified URL.

```
- (id)initWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL

The URL identifying the image.

Return Value

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified URL.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CocoaCreateMovie

Declared In

NSImage.h

initWithData:

Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object.

```
- (id)initWithData:(NSData *)data
```

Parameters*data*

The data object containing the image data.

Return Value

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified data object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSImage.h

initWithIconRef:

Initializes the image object with a Carbon-style icon resource.

```
- (id)initWithIconRef:(IconRef)iconRef
```

Parameters*iconRef*

A reference to a Carbon icon resource.

Return Value

An initialized `NSImage` instance.

Discussion

Creates one or more bitmap image representations, one for each size icon contained in the `IconRef` data structure. This initialization method automatically retains the data in the `iconRef` parameter and loads the bitmaps from that data file lazily.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSImage.h

initWithPasteboard:

Initializes and returns an `NSImage` instance with data from the specified pasteboard.

```
- (id)initWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters*pasteboard*

The pasteboard containing the image data.

Return ValueAn initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the pasteboard.**Discussion**The specified pasteboard should contain a type supported by one of the registered `NSImageRep` subclasses. Table 1 lists the default pasteboard types and file extensions for several `NSImageRep` subclasses.**Table 1** Default pasteboard types for image representations

Image representation class	Default pasteboard type	Default file extensions
<code>NSBitmapImageRep</code>	<code>NSTIFFPboardType</code>	tiff, gif, jpg, and others
<code>NSPDFImageRep</code>	<code>NSPDFPboardType</code>	pdf
<code>NSEPSImageRep</code>	<code>NSPostscriptPboardType</code>	eps
<code>NSPICIImageRep</code>	<code>NSPICTPboardType</code>	pict

If the specified pasteboard contains the value `NSFileNamesPboardType`, each filename on the pasteboard should have an extension supported by one of the registered `NSImageRep` subclasses. You can use the `imageUnfilteredFileTypes` method of a given subclass to obtain the list of supported types for that class.

Availability

Available in Mac OS X v10.0 and later.

Declared In`NSImage.h`**initWithSize:**Initializes and returns an `NSImage` instance whose size is set to the specified value.

- (id)initWithSize:(NSSize)aSize

Parameters*aSize*

The size of the image, measured in points.

Return ValueAn initialized `NSImage` instance.**Discussion**

This method does not add any image representations to the image object.. It is permissible to initialize the receiver by passing a size of (0.0, 0.0); however, the receiver's size must be set to a non-zero value before the `NSImage` object is used or an exception will be raised.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setSize:](#) (page 49)

Related Sample Code

Dicey

Image Difference

RGB Image

RGB ValueTransformers

Sketch-112

Declared In

NSImage.h

isCachedSeparately

Returns a Boolean value indicating whether each image representation caches its contents in a separate offscreen window.

- (BOOL)isCachedSeparately

Return Value

YES if the image representations cache their content in separate offscreen windows; otherwise, NO. The default value is NO.

Discussion

If this method returns NO, it means that the image may be cached in a shared window but is not required to be. Images are cached in a shared window if they have the same general attributes, such as color space, resolution, and bit depth.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSImage.h

isDataRetained

Returns a Boolean value indicating whether the receiver retains its source image data.

- (BOOL)isDataRetained

Return Value

YES if the image retains its source data; otherwise, NO. The default value is NO with some exceptions, which are covered in the discussion.

Discussion

For image objects initialized using either the [initWithReferencingFile:](#) (page 31) or [initWithReferencingURL:](#) (page 32) method, this value is YES by default. The reason is that for these methods, data retention simply involves retaining the filename or URL.

Data retention increases the memory used by the `NSImage` object and its image representations.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSImage.h`

isFlipped

Returns a Boolean value indicating whether the image uses a flipped coordinate system.

- (BOOL)isFlipped

Return Value

YES if the image's coordinate system is flipped; otherwise, NO. The default is NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setFlipped:](#) (page 46)

Declared In

`NSImage.h`

isTemplate

Returns a Boolean value indicating whether the image is a template image.

- (BOOL)isTemplate

Return Value

YES if the image is a template image; otherwise, NO.

Discussion

Template images consist of black and clear colors (and an alpha channel). Template images are not intended to be used as standalone images and are usually mixed with other content to create the desired final appearance.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setTemplate:](#) (page 50)

Declared In

`NSImage.h`

isValid

Returns a Boolean value indicating whether an image representation from the receiver can be drawn.

- (BOOL)isValid

Return Value

YES if the receiver can be drawn; otherwise, NO.

Discussion

If the receiver is initialized with an existing image file, but the corresponding image data is not yet loaded into memory, this method loads the data and expands it as needed. If the receiver contains no image representations and no associated image file, this method creates a valid cached image representation and initializes it to the default bit depth. This method returns NO in cases where the file or URL from which it was initialized is nonexistent or when the data in an existing file is invalid.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithReferencingFile:](#) (page 31)
- [initWithReferencingURL:](#) (page 32)

Declared In

NSImage.h

lockFocus

Prepares the image to receive drawing commands.

- (void)lockFocus

Discussion

This method sets the current drawing context to the area of the offscreen window used to cache the receiver's contents. Subsequent drawing commands are composited to this offscreen window. If the offscreen drawing area already has some content, any new drawing commands are composited with that content. This method does not modify the original image data directly.

When locking focus, this method chooses the best image representation object available and locks focus on that object. If the receiver has no image representations, this method creates one with the default depth and locks focus on it. For information on how the "best" representation is chosen, see the Images chapter of *Cocoa Drawing Guide*.

A successful `lockFocus` message must be balanced with a matching `unlockFocus` (page 53) message to the same `NSImage` object. These messages bracket the code that draws the image.

If `lockFocus` is unable to focus on the image, it raises an `NSImageCacheException`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [bestRepresentationForDevice:](#) (page 22)
- [isValid](#) (page 37)
- [prefersColorMatch](#) (page 40)
- [representations](#) (page 41)

Related Sample Code

Dicey

Image Difference

RGB Image

RGB ValueTransformers

Sketch-112

Declared In

NSImage.h

lockFocusOnRepresentation:

Prepares the specified image representation to receive drawing commands.

```
- (void)lockFocusOnRepresentation:(NSImageRep *)imageRepresentation
```

Parameters

imageRepresentation

An image representation belonging to the receiver, or `nil` if you want the receiver to choose which image representation to use.

Discussion

This method sets the current drawing context to the area of the offscreen window used to cache the specified image representation's contents. Subsequent drawing commands are composited to this offscreen window. If the offscreen drawing area already has some content, any new drawing commands are composited with that content. This method does not modify the original image data directly.

If *imageRepresentation* is `nil`, this method acts like the [lockFocus](#) (page 38) method, setting the focus to the best representation for the `NSImage` object.

A successful `lockFocusOnRepresentation:` message must be balanced with a matching [unlockFocus](#) (page 53) message to the same `NSImage` object. These messages bracket the code that draws the image.

If `lockFocusOnRepresentation:` is unable to focus on the specified image representation, it raises an `NSImageCacheException`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [isValid](#) (page 37)

Declared In

NSImage.h

matchesOnMultipleResolution

Returns a Boolean value indicating whether image representations whose resolution is an integral multiple of the device resolution are considered a match.

```
- (BOOL)matchesOnMultipleResolution
```

Return Value

YES if image representations whose resolution is an integral multiple of the device resolution are considered a match; otherwise, NO.

Discussion

When this method returns NO, only image representations whose resolution is exactly the same as the device resolution are considered matches. If this method returns YES and multiple image representations fit this criteria, the one whose resolution is closest to the device resolution is chosen.

The default value is YES.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setMatchesOnMultipleResolution:](#) (page 47)

Declared In

NSImage.h

name

Returns the name associated with the receiver, if any.

- (NSString *)name

Return Value

The name associated with the receiver, or nil if no name is assigned.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setName:](#) (page 47)

Declared In

NSImage.h

prefersColorMatch

Returns a Boolean value indicating whether the image prefers to choose image representations using color matching or resolution matching.

- (BOOL)prefersColorMatch

Return Value

YES if color matching is preferred over resolution matching; otherwise NO if resolution matching is preferred.

Discussion

Both color matching and resolution matching may influence the choice of an image representation. This method simply indicates which technique is used first during the selection process. The default value is YES.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setPrefersColorMatch:](#) (page 48)

Declared In

NSImage.h

recache

Invalidates and frees the offscreen caches of all image representations.

- (void)recache

Discussion

If you modify an image representation, you must send a [recache](#) (page 41) message to the corresponding image object to force the changes to be recached. The next time any image representation is drawn, it is asked to recreate its cached image. If you do not send this message, the image representation may use the old cache data. This method simply clears the cached image data; it does not delete the `NSCachedImageRep` objects associated with any image representations.

If you do not plan to use an image again right away, you can free its caches to reduce the amount of memory consumed by your program.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSImage.h

removeRepresentation:

Removes the specified image representation from the receiver and releases it.

- (void)removeRepresentation:(NSImageRep *)*imageRep*

Parameters

imageRep

The image representation object you want to remove.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [representations](#) (page 41)

Declared In

NSImage.h

representations

Returns an array containing all of the receiver's image representations.

- (NSArray *)representations

Return Value

An array containing zero or more `NSImageRep` objects.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CocoaCreateMovie

OpenGLCompositorLab

Reducer

Declared In

`NSImage.h`

scalesWhenResized

Returns a Boolean value indicating whether image representations are scaled to fit the receiver's size.

- (BOOL)scalesWhenResized

Return Value

YES if image representations are scaled to fit the receiver; otherwise, NO. The default value is NO.

Discussion

Images are not resized during drawing if this method returns YES. They are only resized when you change the size by sending the receiver a `setSize:` (page 49) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setScaleWhenResized:](#) (page 48)

Declared In

`NSImage.h`

setAlignmentRect:

Sets the alignment metadata that your code can use to position the image during layout.

- (void)setAlignmentRect:(NSRect)rect

Parameters

rect

The alignment rectangle for the image.

Discussion

Alignment rectangles specify baselines that you can use to position the content of an image more accurately. These baselines are merely hints that your own code can use to determine positioning and are not used internally by `NSImage` itself during drawing. For example, if you have a 20 x 20 pixel icon that includes a glow effect, you might set the alignment rectangle to `{{2, 2}, {16, 16}}` to indicate the position of the underlying icon without the glow effect.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [alignmentRect](#) (page 21)

Declared In

NSImage.h

setBackground-color:

Sets the background color of the image.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The new background color for the image.

Discussion

The background color is visible only if the drawn image representation does not completely cover all of the pixels available for the image's current size. The background color is ignored for cached image representations; such caches are always created with a white background. This method does not cause the receiver to recache itself.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [recache](#) (page 41)

- [backgroundColor](#) (page 21)

Declared In

NSImage.h

setCacheDepthMatchesImageDepth:

Sets whether the receiver's offscreen window caches use the same bit depth as the image data itself.

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

Parameters

flag

YES if the offscreen caches use the same bit-depth associated with the image data; otherwise, NO to indicate they should use the default bit depth.

Discussion

This method does not cause the receiver to recache itself. The default depth limit is equal to the bit depth of the deepest screen on the system.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [cacheDepthMatchesImageDepth](#) (page 22)
- [lockFocus](#) (page 38)
- [recache](#) (page 41)

Declared In

NSImage.h

setCachedSeparately:

Sets whether each image representation uses a separate offscreen window to cache its contents.

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

Parameters*flag*

YES if you want each of the receiver's image representation objects to use a separate offscreen window for caching; otherwise, NO.

Discussion

If you specify NO, a representation can be cached together with other images, though in practice it might not be. This method does not invalidate any existing caches.

If you plan to resize an NSImage object frequently, it is usually more efficient to cache its representations separately. In some situations, you might also want to enable separate caching if you plan to use the [compositeToPoint:fromRect:operation:](#) (page 23) or [compositeToPoint:fromRect:operation:fraction:](#) (page 25) methods to draw the image.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [recache](#) (page 41)

Declared In

NSImage.h

setCacheMode:

Set the receiver's caching mode.

```
- (void)setCacheMode:(NSImageCacheMode)mode
```

Parameters*mode*

The caching mode to use with this image. For a list of possible values, see [NSImageCacheMode](#) (page 60).

Discussion

The caching mode determines when the receiver's image representations use offscreen caches. Offscreen caches speed up rendering time but do so by using extra memory. In the default caching mode (`NSImageCacheDefault`), each image representation chooses the caching technique that produces the

fastest drawing times. For example, in the default mode, the `NSPDFImageRep` and `NSEPSImageRep` classes use the `NSImageCacheAlways` mode but the `NSBitmapImageRep` class uses the `NSImageCacheBySize` mode.

For more information on image caching behavior, see the Images chapter of *Cocoa Drawing Guide*.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [cacheMode](#) (page 23)

Declared In

`NSImage.h`

setDataRetained:

Sets whether the receiver retains its source image data.

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

Parameters

flag

YES if you want the source image data to be retained; otherwise NO.

Discussion

Retention of the source image data is important if the source of the image data could change, be moved, or be deleted. Data retention is also useful if you plan to resize an image frequently; otherwise, resizing occurs on a cached copy of the image, which can lose image quality during successive scaling operations. With data retention enabled, the image is resized from the original source data.

If the responsibility for drawing the image is delegated to another object, there is no reason to retain the image data. Similarly, if the source of the image data is not expected to change or you do not plan to resize the image, you do not need to retain the data. In fact, retaining the data leads to increased memory usage, which could have a negative impact on performance.

If you create your image object using the [initWithReferencingFile:](#) (page 31) method, the only data retained is the name of the source file.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSImage.h`

setDelegate:

Sets the delegate object of the receiver.

```
- (void)setDelegate:(id)anObject
```

Parameters*anObject*

The new delegate object.

Availability

Available in Mac OS X v10.0 and later.

See Also- [delegate](#) (page 27)**Declared In**

NSImage.h

setFlipped:

Sets whether the polarity of the y axis is inverted when drawing an image.

- (void)setFlipped:(BOOL)*flag***Parameters***flag*

YES if you want the image data to be inverted before drawing; otherwise, NO.

Discussion

If *flag* is YES, the y-axis of the image's internal coordinate system is inverted, with the origin in the upper-left corner and the positive y axis extending downward. This method affects only the coordinate system used internally by the image and the orientation of the image when it is drawn; it does not affect the coordinate system used to specify the position of an image in a view. This method does not cause the receiver to recache itself.

If you set *flag* to YES and then lock focus and draw into the image, the content you draw is cached in the inverted (flipped) orientation. Changing the value for *flag* does not affect the orientation of the cached image.

Availability

Available in Mac OS X v10.0 and later.

See Also- [isFlipped](#) (page 37)- [recache](#) (page 41)**Related Sample Code**

DockTile

Sketch-112

SpeedometerView

WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In

NSImage.h

setMatchesOnMultipleResolution:

Sets whether image representations whose resolutions are integral multiples of the device resolution are considered a match.

- (void)setMatchesOnMultipleResolution:(BOOL)*flag*

Parameters

flag

YES if image representations whose resolution is an integral multiple of the device resolution should be considered a match; otherwise, NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [matchesOnMultipleResolution](#) (page 39)

Declared In

NSImage.h

setName:

Registers the receiver under the specified name.

- (BOOL)setName:(NSString *)*aString*

Parameters

aString

The name to associate with the receiver.

Return Value

YES if the receiver was successfully registered with the given name; otherwise, NO.

Discussion

If the receiver is already registered under a different name, this method unregisters the other name. If a different image is registered under the name specified in *aString*, this method does nothing and returns NO.

When naming an image using this method, it is convention not to include filename extensions in the names you specify. That way, you can easily distinguish between images you have named explicitly and those you want to load from the application's bundle. For information about the rules used to search for images, and for information about the ownership policy of named images, see the `imageNamed:` method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [name](#) (page 40)

+ [imageNamed:](#) (page 16)

Related Sample Code

Clock Control

QTKitMovieShuffler

Declared In

NSImage.h

setPrefersColorMatch:

Sets whether choosing an image representation favors color matching over resolution matching.

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

Parameters*flag*

YES if the receiver should match the color capabilities of the rendering device first; otherwise, NO to indicate that resolution matching is preferred.

Discussion

Both color matching and resolution matching may influence the choice of an image representation. You use this method to choose which technique should be used first during the selection process.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [prefersColorMatch](#) (page 40)

Declared In

NSImage.h

setScaleWhenResized:

Sets whether different-sized image representations are scaled to fit the receiver's size.

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

Parameters*flag*

YES if image representations are scaled to fit; otherwise NO.

Discussion

Most images (especially those loaded from files and URLs) contain only a single image representation whose size is the same as the receiver. It is possible to add image representations using the [addRepresentation:](#) (page 20) or [addRepresentations:](#) (page 20) methods but doing so is rarely necessary because modern hardware is powerful enough to resize and scale images quickly. The only reason to consider creating new representations is if each representation contains a customized version of the image at a specific size. (TIFF images may also contain a thumbnail version of an image, which is stored using a separate image representation.) If you pass YES in the *flag* parameter, and subsequently send a [setSize:](#) (page 49) message to the receiver, all such image representations would be scaled to the same size. Scaling of bitmap images usually results in the interpolation of the bitmap data.

This method does not invalidate the caches of any of the receiver's image representations. The caches are not invalidated until you change the image size using a [setSize:](#) (page 49) message. Scaling affects only the cached offscreen data for a given image representation.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [scalesWhenResized](#) (page 42)

Related Sample Code

CocoaDragAndDrop

CoreRecipes

FunkyOverlayWindow

MyCustomColorPicker

Declared In

NSImage.h

setSize:

Sets the width and height of the image.

- (void)setSize:(NSSize)aSize

Parameters

aSize

The new size of the image, measured in points.

Discussion

The size of an `NSImage` object must be set before it can be used. If the size of the image hasn't already been set when an image representation is added, the size is taken from the image representation's data. For EPS images, the size is taken from the image's bounding box. For TIFF images, the size is taken from the `ImageLength` and `ImageWidth` attributes.

Changing the size of an `NSImage` after it has been used effectively resizes the image. Changing the size invalidates all its caches and frees them. When the image is next composited, the selected representation will draw itself in an offscreen window to recreate the cache.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [size](#) (page 51)

- [initWithSize:](#) (page 35)

- [setScaleWhenResized:](#) (page 48)

Related Sample Code

Aperture Edit Plugin - Borders & Titles

CocoaDragAndDrop

CoreRecipes

FunkyOverlayWindow

MyCustomColorPicker

Declared In

NSImage.h

setTemplate:

Sets whether the image represents a template image.

```
- (void)setTemplate:(BOOL)isTemplate
```

Parameters

isTemplate

Specify YES if the image is a template image; otherwise, NO.

Discussion

Images you mark as template images should consist of only black and clear colors. You can use the alpha channel in the image to adjust the opacity of black content, however.

Template images are not intended to be used as standalone images. They are always mixed with other content and processed to create the desired appearance. You can mark an image as a “template image” to notify clients who care that the image contains only black and clear content. The most common use for template images is in image cells. For example, you might use a template image to provide the content for a button or segmented control. Cocoa cells take advantage of the nature of template images—that is, their simplified color scheme and use of transparency—to improve the appearance of the corresponding control in each of its supported states.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [isTemplate](#) (page 37)

Declared In

NSImage.h

setUsesEPSOnResolutionMismatch:

Sets whether EPS image representations are preferred when no other representations match the resolution of the device.

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

Parameters

flag

YES if EPS image representations are preferred; otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [usesEPSOnResolutionMismatch](#) (page 53)

- [setMatchesOnMultipleResolution:](#) (page 47)

Declared In

NSImage.h

size

Returns the size of the receiver.

- (NSSize) size

Return Value

The size of the receiver or (0.0, 0.0) if no size has been set and the size cannot be determined from any of the receiver's image representations.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setSize:](#) (page 49)

Related Sample Code

Clock Control

Reducer

RGB Image

Sketch-112

Transformed Image

Declared In

NSImage.h

TIFFRepresentation

Returns a data object containing TIFF data for all of the image representations in the receiver.

- (NSData *)TIFFRepresentation

Return Value

A data object containing the TIFF data, or `nil` if the TIFF data could not be created.

Discussion

You can use the returned data object to write the TIFF data to a file. For each image representation, this method uses the TIFF compression option associated with that representation or `NSTIFFCompressionNone`, if no option is set.

If one of the receiver's image representations does not support the creation of TIFF data natively (PDF and EPS images, for example), this method creates the TIFF data from that representation's cached content.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [TIFFRepresentationUsingCompression:factor:](#) (page 52)

- [representationUsingType:properties:](#) (`NSBitmapImageRep`)

- [TIFFRepresentation](#) (`NSBitmapImageRep`)

- [TIFFRepresentationUsingCompression:factor:](#) (`NSBitmapImageRep`)

Related Sample Code

bMoviePaletteCocoa
 GLSLShowpiece
 NURBSSurfaceVertexProg
 Sketch-112
 Vertex Optimization

Declared In

NSImage.h

TIFFRepresentationUsingCompression:factor:

Returns a data object containing TIFF data with the specified compression settings for all of the image representations in the receiver.

```
- (NSData *)TIFFRepresentationUsingCompression:(NSTIFFCompression)comp
  factor:(float)aFloat
```

Parameters

comp

The type of compression to use. For a list of values, see the constants in `NSBitmapImageRep`.

aFloat

Provides a hint for compression types that implement variable compression ratios. Currently, only JPEG compression uses a compression factor.

Return Value

A data object containing the TIFF data, or `nil` if the TIFF data could not be created.

Discussion

You can use the returned data object to write the TIFF data to a file. If the specified compression isn't applicable, no compression is used. If a problem is encountered during generation of the TIFF data, this method may raise an exception.

If one of the receiver's image representations does not support the creation of TIFF data natively (PDF and EPS images, for example), this method creates the TIFF data from that representation's cached content.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [TIFFRepresentation](#) (page 51)
- `representationUsingType:properties:` (`NSBitmapImageRep`)
- `TIFFRepresentation` (`NSBitmapImageRep`)
- `TIFFRepresentationUsingCompression:factor:` (`NSBitmapImageRep`)

Related Sample Code

PDFKitLinker2

Declared In

NSImage.h

unlockFocus

Removes the focus from the receiver.

- (void)unlockFocus

Discussion

This message must be sent after a successful `lockFocus` or `lockFocusOnRepresentation:` message and the completion of any intermediate drawing commands. This method restores the focus to the previous owner, if any.

Do not send this message if the preceding call to lock focus raised an `NSImageCacheException`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Dicey

Image Difference

RGB Image

RGB ValueTransformers

Sketch-112

Declared In

NSImage.h

usesEPSOnResolutionMismatch

Returns a Boolean value indicating whether EPS representations are preferred when no other representations match the resolution of the device.

- (BOOL)usesEPSOnResolutionMismatch

Return Value

YES if EPS image representations are preferred; otherwise NO.

Discussion

The default value is NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setUsesEPSOnResolutionMismatch:](#) (page 50)
- [matchesOnMultipleResolution](#) (page 39)

Declared In

NSImage.h

Delegate Methods

image:didLoadPartOfRepresentation:withValidRows:

During incremental loading, this method is called repeatedly to inform the delegate that more of the image data is available.

```
- (void)image:(NSImage *)image didLoadPartOfRepresentation:(NSImageRep *)rep
  withValidRows:(NSInteger)rows
```

Parameters

image

The image object whose contents are being loaded.

rep

The image representation object that is receiving and processing the image data.

rows

The number of rows of data that have been decompressed.

Discussion

This method is optional; incremental loading will continue if the delegate does not implement it.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSImage.h

image:didLoadRepresentation:withStatus:

For incremental loading, this method is invoked when the specified image has been loaded and decompressed as fully as is possible.

```
- (void)image:(NSImage *)image didLoadRepresentation:(NSImageRep *)rep
  withStatus:(NSImageLoadStatus)status
```

Parameters

image

The image object whose contents are being loaded.

rep

The image representation object that loaded the image data.

status

The status of the load operation. For a list of possible values, see [“Constants”](#) (page 56).

Discussion

The delegate must implement this method if it wants to support the incremental loading of images. In that case, you must also set up the image object to be loaded lazily, by initializing it using the [initByReferencingFile:](#) (page 31) or [initByReferencingURL:](#) (page 32) method.

If an error occurs during downloading or decompression, the *status* parameter is set to `NSImageLoadStatusInvalidData`, `NSImageLoadStatusUnexpectedEOF`, or `NSImageLoadStatusReadError`. If the download was cancelled, the *status* parameter is set to `NSImageLoadStatusCancelled`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSImage.h`

image:didLoadRepresentationHeader:

During incremental loading, this method is called once enough data has been read to determine the size of the image.

```
- (void)image:(NSImage *)image didLoadRepresentationHeader:(NSImageRep *)rep
```

Parameters

image

The image object whose contents are being loaded.

rep

The image representation object that is receiving and processing the image data.

Discussion

By the time this method is called, the `NSBitmapImageRep` object specified in the *rep* parameter is valid and has allocated the memory needed to store the bitmap. The bitmap itself is filled with the image's background color. This method is optional; incremental loading will continue if the delegate does not implement it.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSImage.h`

image:willLoadRepresentation:

For incremental loading, this method is invoked when you first attempt to draw the image or otherwise access the bitmap data.

```
- (void)image:(NSImage *)image willLoadRepresentation:(NSImageRep *)rep
```

Parameters

image

The image object whose contents need to be loaded.

rep

The image representation object that was accessed.

Discussion

Downloading of the image begins immediately after this method returns. This method is optional; incremental loading will continue if the delegate does not implement it.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSImage.h

imageDidNotDraw:inRect:

Sent to the delegate when the image object is unable, for whatever reason, to lock focus on its image or draw in the specified rectangle.

```
- (NSImage *)imageDidNotDraw:(id)sender inRect:(NSRect)aRect
```

Parameters

sender

The NSImage object that encountered the problem.

aRect

The rectangle that the image object was attempting to draw.

Return Value

An NSImage to draw in place of the one in *sender*, or *nil* if the delegate wants to draw the image itself.

Discussion

The delegate can do one of the following:

- Return another NSImage object to draw in the sender's place.
- Draw the image itself and return *nil*.
- Simply return *nil* to indicate that *sender* should give up on the attempt at drawing the image.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSImage.h

Constants

NSCompositingOperation

These constants specify compositing operators described in terms of having source and destination images, each having an opaque and transparent region. The destination image after the operation is defined in terms of the source and destination before images.


```
typedef enum _NSCompositingOperation {
    NSCompositeClear          = 0,
    NSCompositeCopy           = 1,
    NSCompositeSourceOver     = 2,
    NSCompositeSourceIn       = 3,
    NSCompositeSourceOut      = 4,
    NSCompositeSourceAtop     = 5,
    NSCompositeDestinationOver = 6,
    NSCompositeDestinationIn  = 7,
    NSCompositeDestinationOut = 8,
    NSCompositeDestinationAtop = 9,
    NSCompositeXOR            = 10,
    NSCompositePlusDarker     = 11,
    NSCompositeHighlight      = 12,
    NSCompositePlusLighter    = 13
} NSCompositingOperation;
```

Constants

NSCompositeClear

Transparent. ($R = 0$)**Available in Mac OS X v10.0 and later.****Declared in NSGraphics.h.**

NSCompositeCopy

Source image. ($R = S$)**Available in Mac OS X v10.0 and later.****Declared in NSGraphics.h.**

NSCompositeSourceOver

Source image wherever source image is opaque, and destination image elsewhere. ($R = S + D*(1 - Sa)$)**Available in Mac OS X v10.0 and later.****Declared in NSGraphics.h.**

NSCompositeSourceIn

Source image wherever both images are opaque, and transparent elsewhere. ($R = S*Da$)**Available in Mac OS X v10.0 and later.****Declared in NSGraphics.h.**

NSCompositeSourceOut

Source image wherever source image is opaque but destination image is transparent, and transparent elsewhere. ($R = S*(1 - Da)$)**Available in Mac OS X v10.0 and later.****Declared in NSGraphics.h.**

NSCompositeSourceAtop

Source image wherever both images are opaque, destination image wherever destination image is opaque but source image is transparent, and transparent elsewhere. ($R = S*Da + D*(1 - Sa)$)**Available in Mac OS X v10.0 and later.****Declared in NSGraphics.h.**

NSCompositeDestinationOver

Destination image wherever destination image is opaque, and source image elsewhere. ($R = S*(1 - D_a) + D$)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeDestinationIn

Destination image wherever both images are opaque, and transparent elsewhere. ($R = D*S_a$)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeDestinationOut

Destination image wherever destination image is opaque but source image is transparent, and transparent elsewhere. ($R = D*(1 - S_a)$)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeDestinationAtop

Destination image wherever both images are opaque, source image wherever source image is opaque but destination image is transparent, and transparent elsewhere. ($R = S*(1 - D_a) + D*S_a$)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeXOR

Exclusive OR of source and destination images. ($R = S*(1 - D_a) + D*(1 - S_a)$)

Works only with black and white images and is not recommended for color contexts.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositePlusDarker

Sum of source and destination images, with color values approaching 0 as a limit. ($R = \text{MAX}(0, (1 - D) + (1 - S))$)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeHighlight

Source image wherever source image is opaque, and destination image elsewhere. (**Deprecated.** Mapped to `NSCompositeSourceOver`.)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositePlusLighter

Sum of source and destination images, with color values approaching 1 as a limit. ($R = \text{MIN}(1, S + D)$)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

Discussion

These compositing operators are defined in and used by [compositeToPoint:fromRect:operation:](#) (page 23), [compositeToPoint:operation:](#) (page 25), [compositeToPoint:fromRect:operation:fraction:](#) (page 25), [compositeToPoint:operation:fraction:](#) (page 26),

[drawAtPoint:fromRect:operation:fraction:](#) (page 29), and [drawInRect:fromRect:operation:fraction:](#) (page 30). They are also used by drawing methods in other classes that take a compositing operator.

The equations after each constant represent the mathematical formulas used to calculate the color value of the resulting pixel. Table 2 lists the meaning of each placeholder value in the equations.

Table 2 Placeholder values for compositing equations

Para	Para
R	The premultiplied result color.
S	The source color
D	The destination color
Sa	The alpha value of the source color
Da	The alpha value of the destination color

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSGraphics.h

NSImageLoadStatus

These constants are status values passed to the incremental loading delegate method [image:didLoadRepresentation:withStatus:](#) (page 54).

```
typedef enum {
    NSImageLoadStatusCompleted,
    NSImageLoadStatusCancelled,
    NSImageLoadStatusInvalidData,
    NSImageLoadStatusUnexpectedEOF,
    NSImageLoadStatusReadError
} NSImageLoadStatus;
```

Constants

NSImageLoadStatusCompleted

Enough data has been provided to completely decompress the image.

Available in Mac OS X v10.2 and later.

Declared in NSImage.h.

NSImageLoadStatusCancelled

Image loading was canceled.

The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in NSImage.h.

`NSImageLoadStatusInvalidData`

An error occurred during image decompression.

The image data is probably corrupt. The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusUnexpectedEOF`

Not enough data was available for full decompression of the image.

The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusReadError`

Not enough data was available for full decompression of the image.

The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSImage.h`

NSImageCacheMode

These constants specify the caching policy on a per `NSImage` basis. The caching policy is set using [cacheMode](#) (page 23) and [setCacheMode:](#) (page 44).

```
typedef enum {
    NSImageCacheDefault,
    NSImageCacheAlways,
    NSImageCacheBySize,
    NSImageCacheNever
} NSImageCacheMode;
```

Constants

`NSImageCacheDefault`

Caching is unspecified.

Use the image rep's default.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageCacheAlways`

Always generate a cache when drawing.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageCacheBySize`

Cache if cache size is smaller than the original data.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageCacheNever`

Never cache; always draw direct.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

Discussion

The following table specifies the default caching policy for the various types of image representation.

Image Rep Class	Default caching policy
<code>NSBitmapImageRep</code>	<code>NSImageCacheBySize</code> . Cache if bitmap is 32-bits in 16-bit world or greater than 72 dpi.
<code>NSPICTImageRep</code>	<code>NSImageCacheBySize</code> . Same reasoning as <code>NSBitmapImageRep</code> in the event the PICT contains a bitmap.
<code>NSPDFImageRep</code>	<code>NSImageCacheAlways</code>
<code>NSCIImageRep</code>	<code>NSImageCacheBySize</code> . Cache if the bitmap depth does not match the screen depth or the resolution is greater than 72 dpi.
<code>NSEPSImageRep</code>	<code>NSImageCacheAlways</code>
<code>NSCustomImageRep</code>	<code>NSImageCacheAlways</code>

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSImage.h`

Button Template Images

Images representing standard artwork and icons that you can use in your applications

```

NSString *const NSImageNameQuickLookTemplate;
NSString *const NSImageNameBluetoothTemplate;
NSString *const NSImageNameIChatTheaterTemplate;
NSString *const NSImageNameSlideshowTemplate;
NSString *const NSImageNameActionTemplate;
NSString *const NSImageNameSmartBadgeTemplate;
NSString *const NSImageNamePathTemplate;
NSString *const NSImageNameInvalidDataFreestandingTemplate;
NSString *const NSImageNameLockLockedTemplate;
NSString *const NSImageNameLockUnlockedTemplate;
NSString *const NSImageNameGoRightTemplate;
NSString *const NSImageNameGoLeftTemplate;
NSString *const NSImageNameRightFacingTriangleTemplate;
NSString *const NSImageNameLeftFacingTriangleTemplate;
NSString *const NSImageNameAddTemplate;
NSString *const NSImageNameRemoveTemplate;
NSString *const NSImageNameRevealFreestandingTemplate;
NSString *const NSImageNameFollowLinkFreestandingTemplate;
NSString *const NSImageNameEnterFullScreenTemplate;
NSString *const NSImageNameExitFullScreenTemplate;
NSString *const NSImageNameStopProgressTemplate;
NSString *const NSImageNameStopProgressFreestandingTemplate;
NSString *const NSImageNameRefreshTemplate;
NSString *const NSImageNameRefreshFreestandingTemplate;

```

Constants

NSImageNameQuickLookTemplate

A Quick Look template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameBluetoothTemplate

A Bluetooth template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameIChatTheaterTemplate

An iChat Theater template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameSlideshowTemplate

A slideshow template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameActionTemplate

An action menu template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameSmartBadgeTemplate

A badge for a “smart” item. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNamePathTemplate


A path button template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameInvalidDataFreestandingTemplate

An invalid data template image. Place this icon to the right of any fields containing invalid data. You

can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameLockLockedTemplate

A locked lock template image. Use to indicate locked content. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


NSImageNameLockUnlockedTemplate

An unlocked lock template image. Use to indicate modifiable content that can be locked. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameGoRightTemplate

A “go forward” template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameGoLeftTemplate

A “go back” template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameRightFacingTriangleTemplate

A generic right-facing triangle template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


NSImageNameLeftFacingTriangleTemplate

A generic left-facing triangle template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameAddTemplate

An add item template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameRemoveTemplate`

A remove item template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameRevealFreestandingTemplate`

A reveal contents template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameFollowLinkFreestandingTemplate`

A link template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameEnterFullScreenTemplate`

An enter full-screen mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameExitFullScreenTemplate`

An exit full-screen mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameStopProgressTemplate`

A stop progress button template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameStopProgressFreestandingTemplate`

A stop progress template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameRefreshTemplate`

A refresh template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRefreshFreestandingTemplate`

A refresh template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion

To access these images, pass the specified constant to the `imageNamed:` (page 16) method.

Images with the word “Template” in their title identify shapes that are not intended as standalone images. You would typically use these icons as the custom image for a button, or you might apply them to a cell in a control. For example, you might use the `NSImageNameLockLockedTemplate` image to indicate an item is not modifiable. Template images should use black and clear colors only and it is fine to include varying levels of alpha.

Images with the word “Freestanding” in their title can be used to implement borderless buttons. You do not need to include any extra bezel artwork behind such images.

You should always use named images according to their intended purpose, and not according to how the image appears when loaded. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameRefreshFreestandingTemplate` would correspond to an image named “NSRefreshFreestandingTemplate” in Interface Builder.

Declared In

`NSImage.h`


Multiple Documents Drag Image

Drag images you can use in your applications.

```
NSString *const NSImageNameMultipleDocuments;
```

Constants

`NSImageNameMultipleDocuments`

A drag image for multiple items. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion

To access this image, pass the specified constant to the `imageNamed:` (page 16) method.

You can use this icon as the drag image when dragging multiple items. You should not use this image for any other intended purpose, however. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “imageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameMultipleDocuments` would correspond to an image named “NSMultipleDocuments” in Interface Builder.

Declared In

NSImage.h


Sharing Permissions Named Images

Images representing sharing permission icons that you can use in your applications.

```
NSString *const NSImageNameUser;
NSString *const NSImageNameUserGroup;
NSString *const NSImageNameEveryone;
```

Constants


NSImageNameUser

Permissions for a single user. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameUserGroup

Permissions for a group of users. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameEveryone

Permissions for all users. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

Discussion

To access these images, pass the specified constant to the `imageName:` (page 16) method.

You should use these images to reflect user and group permission or sharing information. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “imageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameEveryone` would correspond to an image named “NSEveryone” in Interface Builder.

Declared In

NSImage.h

System Entity Images

Images representing Finder items.

```
NSString *const NSImageNameBonjour;
NSString *const NSImageNameDotMac;
NSString *const NSImageNameComputer;
NSString *const NSImageNameFolderBurnable;
NSString *const NSImageNameFolderSmart;
NSString *const NSImageNameNetwork;
```

Constants

NSImageNameBonjour

A Bonjour icon.



Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameDotMac

A Dot Mac icon.



Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameComputer

A computer icon.



Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameFolderBurnable

A burnable folder icon.



Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameFolderSmart

A smart folder icon.



Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameNetwork

A network icon.



Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion

To access these images, pass the specified constant to the `imageNamed:` (page 16) method.

You should use these images to reflect specific elements of the Mac OS X environment. For example, you might use the burnable folder icon if your software allows the user to organize content for burning onto an optical disk. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameNetwork` would correspond to an image named “NSNetwork” in Interface Builder.

Declared In

`NSImage.h`


Toolbar Named Images

Images that you can use in application toolbars.

```
NSString *const NSImageNameUserAccounts;
NSString *const NSImageNamePreferencesGeneral;
NSString *const NSImageNameAdvanced;
NSString *const NSImageNameInfo;
NSString *const NSImageNameFontPanel;
NSString *const NSImageNameColorPanel;
```

Constants


`NSImageNameUserAccounts`

User account toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNamePreferencesGeneral`

General preferences toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameAdvanced`

Advanced preferences toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameInfo`

An information toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameFontPanel

A font panel toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameColorPanel

A color panel toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

Discussion

To access these images, pass the specified constant to the `imageNameNamed:` (page 16) method.

You should use these images as icons for toolbar items. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameColorPanel` would correspond to an image named “NSColorPanel” in Interface Builder.

Declared In

NSImage.h


View Type Template Images

Images used in segmented controls to switch the current view type.

```
NSString *const NSImageNameIconViewTemplate;
NSString *const NSImageNameListViewTemplate;
NSString *const NSImageNameColumnViewTemplate;
NSString *const NSImageNameFlowViewTemplate;
```

Constants


NSImageNameIconViewTemplate

An icon view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameListViewTemplate

A list view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


`NSImageNameColumnViewTemplate`

A column view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFlowViewTemplate`

A cover flow view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion

To access these images, pass the specified constant to the `imageNamed:` (page 16) method.

Images with the word “Template” in their title identify shapes that are not intended as standalone images. You would typically use these icons as the custom image for a button, or you might apply them to a cell in a control. For example, you might use the `NSImageNameIconViewTemplate` image to indicate an item is not modifiable. Template images should use black and clear colors only and it is fine to include varying levels of alpha.

You should use these images in conjunction with the buttons (usually part of a segmented control) that change the current viewing mode. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameFlowViewTemplate` would correspond to an image named “NSFlowViewTemplate” in Interface Builder.

Declared In

`NSImage.h`

Document Revision History

This table describes the changes to *NSImage Class Reference*.

Date	Notes
2009-01-06	Updated the description of the alignmentRect method.
2008-10-15	Updated descriptions of the imageNamed: and compositing methods.
2007-10-31	Updated for Mac OS X v10.5. Fixed a delegate method name.
	Documented constants representing standard system images.
2006-10-03	Updated descriptions for the imageFileTypes, imageUnfilteredFileTypes, imagePasteboardTypes, and imageUnfilteredPasteboardTypes methods.
	Updated the drawing routine parameter descriptions to reflect the use of NSZeroRect to specify the entire image.
2006-06-28	Added a link to an explanation of how bundles are searched in relation to the imageNamed: method.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`addRepresentation:` [instance method 20](#)
`addRepresentations:` [instance method 20](#)
`alignmentRect` [instance method 21](#)

B

`backgroundColor` [instance method 21](#)
`bestRepresentationForDevice:` [instance method 22](#)
[Button Template Images 61](#)

C

`cacheDepthMatchesImageDepth` [instance method 22](#)
`cacheMode` [instance method 23](#)
`cancelIncrementalLoad` [instance method 23](#)
`canInitWithPasteboard:` [class method 15](#)
`compositeToPoint:fromRect:operation:` [instance method 23](#)
`compositeToPoint:fromRect:operation:fraction:` [instance method 25](#)
`compositeToPoint:operation:` [instance method 25](#)
`compositeToPoint:operation:fraction:` [instance method 26](#)

D

`delegate` [instance method 27](#)
`dissolveToPoint:fraction:` [instance method 27](#)
`dissolveToPoint:fromRect:fraction:` [instance method 28](#)
`drawAtPoint:fromRect:operation:fraction:` [instance method 29](#)
`drawInRect:fromRect:operation:fraction:` [instance method 30](#)

`drawRepresentation:inRect:` [instance method 30](#)

I

`image:didLoadPartOfRepresentation:withValidRows:<NSObject>` [delegate method 54](#)
`image:didLoadRepresentation:withStatus:<NSObject>` [delegate method 54](#)
`image:didLoadRepresentationHeader:<NSObject>` [delegate method 55](#)
`image:willLoadRepresentation:<NSObject>` [delegate method 55](#)
`imageDidNotDraw:inRect:<NSObject>` [delegate method 56](#)
`imageFileTypes` [class method 15](#)
`imageNamed:` [class method 16](#)
`imagePasteboardTypes` [class method 17](#)
`imageTypes` [class method 18](#)
`imageUnfilteredFileTypes` [class method 18](#)
`imageUnfilteredPasteboardTypes` [class method 19](#)
`imageUnfilteredTypes` [class method 19](#)
`initWithReferencingFile:` [instance method 31](#)
`initWithReferencingURL:` [instance method 32](#)
`initWithContentsOfFile:` [instance method 33](#)
`initWithContentsOfURL:` [instance method 33](#)
`initWithData:` [instance method 34](#)
`initWithIconRef:` [instance method 34](#)
`initWithPasteboard:` [instance method 34](#)
`initWithSize:` [instance method 35](#)
`isCachedSeparately` [instance method 36](#)
`isDataRetained` [instance method 36](#)
`isFlipped` [instance method 37](#)
`isTemplate` [instance method 37](#)
`isValid` [instance method 37](#)

L

`lockFocus` [instance method 38](#)
`lockFocusOnRepresentation:` [instance method 39](#)

M

matchesOnMultipleResolution **instance method** 39
 Multiple Documents Drag Image 65

N

name **instance method** 40
 NSCompositeClear **constant** 57
 NSCompositeCopy **constant** 57
 NSCompositeDestinationAtop **constant** 58
 NSCompositeDestinationIn **constant** 58
 NSCompositeDestinationOut **constant** 58
 NSCompositeDestinationOver **constant** 58
 NSCompositeHighlight **constant** 58
 NSCompositePlusDarker **constant** 58
 NSCompositePlusLighter **constant** 58
 NSCompositeSourceAtop **constant** 57
 NSCompositeSourceIn **constant** 57
 NSCompositeSourceOut **constant** 57
 NSCompositeSourceOver **constant** 57
 NSCompositeXOR **constant** 58
 NSCompositingOperation **data type** 56
 NSImageCacheAlways **constant** 60
 NSImageCacheBySize **constant** 61
 NSImageCacheDefault **constant** 60
 NSImageCacheMode **data type** 60
 NSImageCacheNever **constant** 61
 NSImageLoadStatus **data type** 59
 NSImageLoadStatusCancelled **constant** 59
 NSImageLoadStatusCompleted **constant** 59
 NSImageLoadStatusInvalidData **constant** 60
 NSImageLoadStatusReadError **constant** 60
 NSImageLoadStatusUnexpectedEOF **constant** 60
 NSImageNameActionTemplate **constant** 62
 NSImageNameAddTemplate **constant** 63
 NSImageNameAdvanced **constant** 68
 NSImageNameBluetoothTemplate **constant** 62
 NSImageNameBonjour **constant** 67
 NSImageNameColorPanel **constant** 69
 NSImageNameColumnViewTemplate **constant** 70
 NSImageNameComputer **constant** 67
 NSImageNameDotMac **constant** 67
 NSImageNameEnterFullScreenTemplate **constant** 64
 NSImageNameEveryone **constant** 66
 NSImageNameExitFullScreenTemplate **constant** 64
 NSImageNameFlowViewTemplate **constant** 70
 NSImageNameFolderBurnable **constant** 67
 NSImageNameFolderSmart **constant** 67
 NSImageNameFollowLinkFreestandingTemplate **constant** 64
 NSImageNameFontPanel **constant** 69

NSImageNameGoLeftTemplate **constant** 63
 NSImageNameGoRightTemplate **constant** 63
 NSImageNameIChatTheaterTemplate **constant** 62
 NSImageNameIconViewTemplate **constant** 69
 NSImageNameInfo **constant** 68
 NSImageNameInvalidDataFreestandingTemplate **constant** 63
 NSImageNameLeftFacingTriangleTemplate **constant** 63
 NSImageNameListViewTemplate **constant** 69
 NSImageNameLockLockedTemplate **constant** 63
 NSImageNameLockUnlockedTemplate **constant** 63
 NSImageNameMultipleDocuments **constant** 65
 NSImageNameNetwork **constant** 67
 NSImageNamePathTemplate **constant** 63
 NSImageNamePreferencesGeneral **constant** 68
 NSImageNameQuickLookTemplate **constant** 62
 NSImageNameRefreshFreestandingTemplate **constant** 64
 NSImageNameRefreshTemplate **constant** 64
 NSImageNameRemoveTemplate **constant** 64
 NSImageNameRevealFreestandingTemplate **constant** 64
 NSImageNameRightFacingTriangleTemplate **constant** 63
 NSImageNameSlideshowTemplate **constant** 62
 NSImageNameSmartBadgeTemplate **constant** 63
 NSImageNameStopProgressFreestandingTemplate **constant** 64
 NSImageNameStopProgressTemplate **constant** 64
 NSImageNameUser **constant** 66
 NSImageNameUserAccounts **constant** 68
 NSImageNameUserGroup **constant** 66

P

prefersColorMatch **instance method** 40

R

recache **instance method** 41
 removeRepresentation: **instance method** 41
 representations **instance method** 41

S

scalesWhenResized **instance method** 42
 setAlignmentRect: **instance method** 42
 setBackgroundColor: **instance method** 43

setCacheDepthMatchesImageDepth: [instance method 43](#)
setCachedSeparately: [instance method 44](#)
setCacheMode: [instance method 44](#)
setDataRetained: [instance method 45](#)
setDelegate: [instance method 45](#)
setFlipped: [instance method 46](#)
setMatchesOnMultipleResolution: [instance method 47](#)
setName: [instance method 47](#)
setPrefersColorMatch: [instance method 48](#)
setScalesWhenResized: [instance method 48](#)
setSize: [instance method 49](#)
setTemplate: [instance method 50](#)
setUsesEPSOnResolutionMismatch: [instance method 50](#)
Sharing Permissions Named Images [66](#)
size [instance method 51](#)
System Entity Images [67](#)

T

TIFFRepresentation [instance method 51](#)
TIFFRepresentationUsingCompression:factor:
 [instance method 52](#)
Toolbar Named Images [68](#)

U

unlockFocus [instance method 53](#)
usesEPSOnResolutionMismatch [instance method 53](#)

V

View Type Template Images [69](#)