NSFont Class Reference

Cocoa > Text & Fonts

2007-06-08

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

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

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

NSFont 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	Font Handling
Declared in	NSFont.h
Related sample code	DockTile IBFragmentView Quartz Composer QCTV Quartz Composer WWDC 2005 TextEdit TextEditPlus

Overview

`NSFont` objects represent fonts to an application, providing access to characteristics of the font and assistance in laying out glyphs relative to one another. Font objects are also used to establish the current font when drawing in an `NSView`, using the `set` (page 36) method.

You don't create `NSFont` objects using the `alloc` and `init` methods. Instead, you use either `fontWithDescriptor:size:` (page 14) or `fontWithName:size:` (page 16) to look up an available font and alter its size or matrix to your needs. These methods check for an existing font object with the specified characteristics, returning it if there is one. Otherwise, they look up the font data requested and create the appropriate object. `NSFont` also defines a number of methods for getting standard system fonts, such as `systemFontOfSize:` (page 20), `userFontOfSize:` (page 24), and `messageFontOfSize:` (page 18). To request the default size for these standard fonts, pass a negative number or 0 as the font size.

Adopted Protocols

NSCoding

```
- encodeWithCoder:
- initWithCoder:
```

NSCopying

`- copyWithZone:`

Tasks

Creating Arbitrary Fonts

`+ fontWithName:size:` (page 16)
Creates a font object for the specified font name and font size.

`+ fontWithDescriptor:size:` (page 14)
Returns a font object for the specified font descriptor and font size.

`+ fontWithDescriptor:textTransform:` (page 14)
Returns a font object for the specified font descriptor and text transform.

`+ fontWithName:matrix:` (page 15)
Returns a font object for the specified font name and matrix.

Creating User Fonts

`+ userFontOfSize:` (page 24)
Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), in the specified size.

`+ userFixedPitchFontOfSize:` (page 23)
Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), when that font should be fixed-pitch, in the specified size.

Creating System Fonts

`+ boldSystemFontOfSize:` (page 13)
Returns the Aqua system font used for standard interface items that are rendered in boldface type in the specified size.

`+ controlContentFontOfSize:` (page 13)
Returns the font used for the content of controls in the specified size.

`+ labelFontOfSize:` (page 16)
Returns the Aqua font used for standard interface labels in the specified size.

`+ menuFontOfSize:` (page 18)
Returns the font used for menu items, in the specified size.

`+ menuBarFontOfSize:` (page 17)
Returns the font used for menu bar items, in the specified size.

`+ messageFontOfSize:` (page 18)
Returns the font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

+ `paletteFontOfSize:` (page 19)
  Returns the font used for palette window title bars, in the specified size.

+ `systemFontOfSize:` (page 20)
  Returns the Aqua system font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

+ `titleBarFontOfSize:` (page 22)
  Returns the font used for window title bars, in the specified size.

+ `toolTipsFontOfSize:` (page 22)
  Returns the font used for tool tips labels, in the specified size.

Using a Font to Draw

– `set` (page 36)
  Establishes the receiver as the current font for PostScript `show` and other text-drawing operators.

– `setInContext:` (page 36)
  Establishes the receiver as the current font for the specified graphics context.

Getting General Font Information

– `coveredCharacterSet` (page 27)
  Returns an `NSCharacterSet` object containing all of the nominal characters renderable by the receiver, which is all of the entries mapped in the receiver's '`cmap`' table.

– `fontDescriptor` (page 28)
  Returns the receiver's font descriptor.

– `isFixedPitch` (page 31)
  Returns a Boolean value indicating whether all glyphs in the receiver have the same advancement.

– `mostCompatibleStringEncoding` (page 33)
  Returns the string encoding that works best with the receiver, where there are the fewest possible unmatched characters in the string encoding and glyphs in the font.

– `renderingMode` (page 34)
  Returns the rendering mode of the receiver.

Getting Information About Glyphs

– `glyphWithName:` (page 30)
  Returns the named encoded glyph, or –1 if the receiver contains no such glyph.

Getting Metrics Information

+ `labelFontSize` (page 17)
  Returns the size of the standard label font.

+ `smallSystemFontSize` (page 20)
  Returns the size of the standard small system font.

`+ systemFontSize` (page 21)
: Returns the size of the standard system font.

`+ systemFontSizeForControlSize:` (page 21)
: Returns the font size used for the specified control size.

`- advancementForGlyph:` (page 24)
: Returns the nominal spacing for the given glyph—the distance the current point moves after showing the glyph—accounting for the receiver's size.

`- ascender` (page 25)
: Returns the top y-coordinate, offset from the baseline, of the receiver's longest ascender.

`- boundingRectForFont` (page 25)
: Returns the receiver's bounding rectangle, scaled to the font's size.

`- boundingRectForGlyph:` (page 26)
: Returns the bounding rectangle for the specified glyph, scaled to the receiver's size.

`- capHeight` (page 26)
: Returns the receiver's cap height.

`- descender` (page 27)
: Returns the bottom y coordinate, offset from the baseline, of the receiver's longest descender.

`- getAdvancements:forGlyphs:count:` (page 29)
: Returns an array of the advancements for the specified glyphs rendered by the receiver.

`- getAdvancements:forPackedGlyphs:length:` (page 29)
: Returns an array of the advancements for the specified packed glyphs and rendered by the receiver.

`- getBoundingRects:forGlyphs:count:` (page 30)
: Returns an array of the bounding rectangles for the specified glyphs rendered by the receiver.

`- italicAngle` (page 31)
: Returns the receiver's italic angle, the amount that the font is slanted in degrees counterclockwise from the vertical, as read from its AFM file. Because the slant is measured counterclockwise, English italic fonts typically return a negative value.

`- leading` (page 32)
: Returns the receiver's leading.

`- matrix` (page 32)
: Returns the receiver's font matrix, a standard six-element transformation matrix as used in the PostScript language, specifically with the `makefont` operator.

`- maximumAdvancement` (page 32)
: Returns the greatest advancement of any of the receiver's glyphs.

`- numberOfGlyphs` (page 33)
: Returns the number of glyphs in the receiver.

`- pointSize` (page 34)
: Returns the receiver's point size, or the effective vertical point size for a font with a nonstandard matrix.

`- textTransform` (page 37)
: Returns the current transformation matrix for the receiver.

`- underlinePosition` (page 37)
: Returns the baseline offset that should be used when drawing underlines with the receiver, as determined by the font's AFM file.

`– underlineThickness` (page 37)
Returns the thickness that should be used when drawing underlines with the receiver, as determined by the font's AFM file.

`– xHeight` (page 38)
Returns the x-height of the receiver.

Getting Font Names

`– displayName` (page 27)
Returns the name, including family and face, used to represent the font in the user interface, typically localized for the user's language.

`– familyName` (page 28)
Returns the receiver's family name—for example, "Times" or "Helvetica."

`– fontName` (page 29)
Returns the receiver's full font name, as used in PostScript language code—for example, "Times-Roman" or "Helvetica-Oblique."

Setting User Fonts

`+ setUserFont:` (page 20)
Sets the font used by default for documents and other text under the user's control to the specified font.

`+ setUserFixedPitchFont:` (page 19)
Sets the font used by default for documents and other text under the user's control, when that font should be fixed-pitch, to the specified font.

Getting Corresponding Device Fonts

`– printerFont` (page 34)
Returns the scalable PostScript font corresponding to itself.

`– screenFont` (page 35)
Returns the bitmapped screen font corresponding to itself.

`– screenFontWithRenderingMode:` (page 35)
Returns a bitmapped screen font, when sent to a font object representing a scalable PostScript font, with the specified rendering mode, matching the receiver in typeface and matrix (or size), or `nil` if such a font can't be found.

Deprecated Methods

`+ preferredFontNames` (page 47) Deprecated in Mac OS X v10.4
Returns the names of fonts that the Application Kit tries first when a character has no font specified. (Deprecated. The `NSFontDescriptor` constant `NSFontCascadeListAttribute` offers more powerful font substitution management.)

`+ setPreferredFontNames:` (page 47) Deprecated in Mac OS X v10.4
: Sets the list of preferred font names. (Deprecated. The `NSFontDescriptor` constant `NSFontCascadeListAttribute` offers more powerful font substitution management.)

`+ useFont:` (page 48) Deprecated in Mac OS X v10.4
: Records the given font name as one used in the current print operation. (Deprecated. This is now automatically handled by Quartz.)

`- afmDictionary` (page 48) Deprecated in Mac OS X v10.4
: Returns the AFM font's dictionary. (Deprecated. Use accessor functions listed in "`Keys to the AFM Dictionary`" (page 40) instead.)

`- defaultLineHeightForFont` (page 49) Deprecated in Mac OS X v10.4
: Returns the default line height for the receiver. (Deprecated. Use the `NSLayoutManager` method `defaultLineHeightForFont:` instead.)

`- encodingScheme` (page 49) Deprecated in Mac OS X v10.4
: Returns the name of the receiver's encoding scheme. (Deprecated. Use `mostCompatibleStringEncoding` (page 33) instead.)

`- glyphIsEncoded:` (page 50) Deprecated in Mac OS X v10.4
: Returns a Boolean value indicating whether the receiver encodes the given glyph. (Deprecated. The value can be deduced by `aGlyph < [NSFont numberOfGlyphs]` since only `NSNativeShortGlyphPacking` is supported.)

`- glyphPacking` (page 50) Deprecated in Mac OS X v10.4
: Returns the best way to encode the receiver's glyphs into an array of bytes. (Deprecated. Only `NSNativeShortGlyphPacking` (page 39) is supported.)

`- isBaseFont` (page 50) Deprecated in Mac OS X v10.4
: Returns a Boolean value indicating whether the receiver is a PostScript base font. (Deprecated. This information is not relevant to Mac OS X.)

`- positionOfGlyph:forCharacter:struckOverRect:` (page 51) Deprecated in Mac OS X v10.4
: Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:precededByGlyph:isNominal:` (page 51) Deprecated in Mac OS X v10.4
: Calculates and returns the location of a glyph. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:struckOverGlyph:metricsExist:` (page 52) Deprecated in Mac OS X v10.4
: Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:struckOverRect:metricsExist:` (page 52) Deprecated in Mac OS X v10.4
: Overridden by subclasses to calculate and return a suitable location for a glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:withRelation:toBaseGlyph:totalAdvancement:metricsExist:` (page 53) Deprecated in Mac OS X v10.4
: Calculates and returns a suitable location for a glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionsForCompositeSequence:numberOfGlyphs:pointArray:` (page 54) Deprecated in Mac OS X v10.4
: Calculates glyph locations. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- widthOfString:` (page 54) Deprecated in Mac OS X v10.4

Returns the x-axis offset of the current point when the specified string is drawn with a `show` operator in the receiving font. (Deprecated. Use the Application Kit string-drawing methods, as described in NSString Additions.)

Class Methods

boldSystemFontOfSize:

Returns the Aqua system font used for standard interface items that are rendered in boldface type in the specified size.

```
+ (NSFont *)boldSystemFontOfSize:(CGFloat)fontSize
```

Parameters

`fontSize`

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *`fontSize`* is 0 or negative, returns the boldface system font at the default size.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ fontWithName:size:` (page 16)

Related Sample Code

GLChildWindowDemo

PDF Calendar

Worm

Declared In

`NSFont.h`

controlContentFontOfSize:

Returns the font used for the content of controls in the specified size.

```
+ (NSFont *)controlContentFontOfSize:(CGFloat)fontSize
```

Parameters

`fontSize`

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

For example, in a table, the user's input uses the control content font, and the table's header uses another font. If *fontSize* is 0 or negative, returns the control content font at the default size.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ fontWithName:size:` (page 16)

Declared In

`NSFont.h`

fontWithDescriptor:size:

Returns a font object for the specified font descriptor and font size.

```
+ (NSFont *)fontWithDescriptor:(NSFontDescriptor *)fontDescriptor
    size:(CGFloat)fontSize
```

Parameters

fontDescriptor

A font descriptor object.

fontSize

The size in points to which the font is scaled.

Return Value

A font object for the specified descriptor and size.

Discussion

In most cases, you can simply use `fontWithName:size:` (page 16) to create standard scaled fonts.

Availability

Available in Mac OS X v10.4 and later.

See Also

`+ fontWithName:size:` (page 16)

Declared In

`NSFont.h`

fontWithDescriptor:textTransform:

Returns a font object for the specified font descriptor and text transform.

```
+ (NSFont *)fontWithDescriptor:(NSFontDescriptor *)fontDescriptor
    textTransform:(NSAffineTransform *)textTransform
```

Parameters

fontDescriptor

The font descriptor object describing the font to return.

textTransform

An affine transformation applied to the font.

Return Value

A font object for the specified name and transform.

Discussion

In most cases, you can simply use `fontWithName:size:` (page 16) to create standard scaled fonts. If *textTransform* is non-nil, it has precedence over `NSFontMatrixAttribute` in *fontDescriptor*.

Availability

Available in Mac OS X v10.4 and later.

See Also

+ `fontWithName:size:` (page 16)

Declared In

`NSFont.h`

fontWithName:matrix:

Returns a font object for the specified font name and matrix.

```
+ (NSFont *)fontWithName:(NSString *)fontName matrix:(const CGFloat *)fontMatrix
```

Parameters

fontName

The fully specified family-face name of the font.

fontMatrix

A transformation matrix applied to the font.

Return Value

A font object for the specified name and transformation matrix.

Discussion

The *fontName* is a fully specified family-face name, such as Helvetica-BoldOblique or Times-Roman (not a name as shown in the Font Panel). The *fontMatrix* is a standard 6-element transformation matrix as used in the PostScript language, specifically with the `makefont` operator. In most cases, you can simply use `fontWithName:size:` (page 16) to create standard scaled fonts.

You can use the defined value `NSFontIdentityMatrix` for [1 0 0 1 0 0]. Fonts created with a matrix other than `NSFontIdentityMatrix` don't automatically flip themselves in flipped views.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isFlipped` (`NSView`)

Declared In

`NSFont.h`

fontWithName:size:

Creates a font object for the specified font name and font size.

```
+ (NSFont *)fontWithName:(NSString *)fontName size:(CGFloat)fontSize
```

Parameters

fontName

The fully specified family-face name of the font.

fontSize

The size in points to which the font is scaled.

Return Value

A font object for the specified name and size.

Discussion

The *fontName* is a fully specified family-face name, such as Helvetica-BoldOblique or Times-Roman. The *fontSize* is equivalent to using a font matrix of [*fontSize* 0 0 *fontSize* 0 0] with `fontWithDescriptor:size:` (page 14). If you use a *fontSize* of 0.0, this method uses the default User Font size.

Fonts created with this method automatically flip themselves in flipped views. This method is the preferred means for creating fonts.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CocoaVideoFrameToGWorld

CoreRecipes

JSInterpreter

NSGLImage

OpenGL Screensaver

Declared In

`NSFont.h`

labelFontOfSize:

Returns the Aqua font used for standard interface labels in the specified size.

```
+ (NSFont *)labelFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the label font with the default size.

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

Related Sample Code
CIVideoDemoGL

DockTile

SpeedometerView

TrackBall

WebKitPluginStarter

Declared In
`NSFont.h`

labelFontSize

Returns the size of the standard label font.

```
+ (CGFloat)labelFontSize
```

Return Value
The label font size in points.

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

Declared In
`NSFont.h`

menuBarFontOfSize:

Returns the font used for menu bar items, in the specified size.

```
+ (NSFont *)menuBarFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the menu bar font with the default size.

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

See Also
`+ fontWithName:size:` (page 16)

Declared In
`NSFont.h`

menuFontOfSize:

Returns the font used for menu items, in the specified size.

```
+ (NSFont *)menuFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the menu items font with the default size.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ fontWithName:size:` (page 16)

Related Sample Code

QTAudioExtractionPanel

Declared In

`NSFont.h`

messageFontOfSize:

Returns the font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

```
+ (NSFont *)messageFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns this font at the default size. This method is equivalent to `systemFontOfSize:` (page 20).

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ fontWithName:size:` (page 16)

Related Sample Code

CIAnnotation

CIVideoDemoGL

Declared In
`NSFont.h`

paletteFontOfSize:

Returns the font used for palette window title bars, in the specified size.

```
+ (NSFont *)paletteFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the palette title font at the default size.

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

See Also
`+ fontWithName:size:` (page 16)
`+ titleBarFontOfSize:` (page 22)

Declared In
`NSFont.h`

setUserFixedPitchFont:

Sets the font used by default for documents and other text under the user's control, when that font should be fixed-pitch, to the specified font.

```
+ (void)setUserFixedPitchFont:(NSFont *)aFont
```

Discussion
Specifying *aFont* as `nil` causes the default to be removed from the application domain.

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

See Also
`+ setUserFont:` (page 20)
`+ userFixedPitchFontOfSize:` (page 23)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSFont.h`

setUserFont:

Sets the font used by default for documents and other text under the user's control to the specified font.

```
+ (void)setUserFont:(NSFont *)aFont
```

Discussion

Specifying `aFont` as `nil` causes the default to be removed from the application domain.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ setUserFixedPitchFont:` (page 19)

`+ userFontOfSize:` (page 24)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

`NSFont.h`

smallSystemFontSize

Returns the size of the standard small system font.

```
+ (CGFloat)smallSystemFontSize
```

Return Value

The small system font size in points.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Quartz Composer QCTV

WhackedTV

Declared In

`NSFont.h`

systemFontOfSize:

Returns the Aqua system font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

```
+ (NSFont *)systemFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the system font at the default size.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ boldSystemFontOfSize:` (page 13)

`+ userFontOfSize:` (page 24)

`+ userFixedPitchFontOfSize:` (page 23)

`+ fontWithName:size:` (page 16)

Related Sample Code

DatePicker

FilterDemo

IBFragmentView

Quartz Composer QCTV

WhackedTV

Declared In

`NSFont.h`

systemFontSize

Returns the size of the standard system font.

```
+ (CGFloat)systemFontSize
```

Return Value

The standard system font size in points.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Mountains

Declared In

`NSFont.h`

systemFontSizeForControlSize:

Returns the font size used for the specified control size.

```
+ (CGFloat)systemFontSizeForControlSize:(NSControlSize)controlSize
```

Parameters

controlSize

The control size constant.

Return Value

The font size in points for the specified control size.

Discussion

If *`controlSize`* does not correspond to a valid `NSControlSize`, returns the size of the standard system font.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

IBFragmentView

ObjectPath

Declared In

`NSFont.h`

titleBarFontOfSize:

Returns the font used for window title bars, in the specified size.

```
+ (NSFont *)titleBarFontOfSize:(CGFloat)fontSize
```

Parameters

`fontSize`

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *`fontSize`* is 0 or negative, returns the title bar font at the default size. This method is equivalent to `boldSystemFontOfSize:` (page 13).

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ paletteFontOfSize:` (page 19)

Declared In

`NSFont.h`

toolTipsFontOfSize:

Returns the font used for tool tips labels, in the specified size.

```
+ (NSFont *)toolTipsFontOfSize:(CGFloat)fontSize
```

Parameters

`fontSize`

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the tool tips font at the default size.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ fontWithName:size:` (page 16)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

TipWrapper

Declared In

`NSFont.h`

userFixedPitchFontOfSize:

Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), when that font should be fixed-pitch, in the specified size.

```
+ (NSFont *)userFixedPitchFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the fixed-pitch font at the default size.

The system does not guarantee that all the glyphs in a fixed-pitch font are the same width. For example, certain Japanese fonts are dual-pitch, and other fonts may have nonspacing marks that can affect the display of other glyphs.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ userFontOfSize:` (page 24)

`+ fontWithName:size:` (page 16)

`+ setUserFixedPitchFont:` (page 19)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In
`NSFont.h`

userFontOfSize:

Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), in the specified size.

```
+ (NSFont *)userFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the user font at the default size.

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

See Also
`+ userFixedPitchFontOfSize:` (page 23)
`+ fontWithName:size:` (page 16)
`+ setUserFont:` (page 20)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus
UIElementInspector

Declared In
`NSFont.h`

Instance Methods

advancementForGlyph:

Returns the nominal spacing for the given glyph—the distance the current point moves after showing the glyph—accounting for the receiver's size.

```
- (NSSize)advancementForGlyph:(NSGlyph)aGlyph
```

Parameters

aGlyph
The glyph whose advancement is returned.

Return Value
The advancement spacing in points.

Discussion

This spacing is given according to the glyph's movement direction, which is either strictly horizontal or strictly vertical.

Availability

Available in Mac OS X v10.0 and later.

See Also

`– boundingRectForGlyph:` (page 26)

`– maximumAdvancement` (page 32)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

`NSFont.h`

ascender

Returns the top y-coordinate, offset from the baseline, of the receiver's longest ascender.

```
- (CGFloat)ascender
```

Return Value

The distance of the longest ascender's top y-coordinate from the baseline in points.

Availability

Available in Mac OS X v10.0 and later.

See Also

`– descender` (page 27)

`– capHeight` (page 26)

`– xHeight` (page 38)

Related Sample Code

NSFontAttributeExplorer

QTKitTimeCode

Declared In

`NSFont.h`

boundingRectForFont

Returns the receiver's bounding rectangle, scaled to the font's size.

```
- (NSRect)boundingRectForFont
```

Discussion

The bounding rectangle is the union of the bounding rectangles of every glyph in the font.

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

See Also
- `boundingRectForGlyph:` (page 26)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

boundingRectForGlyph:

Returns the bounding rectangle for the specified glyph, scaled to the receiver's size.

```
- (NSRect)boundingRectForGlyph:(NSGlyph)aGlyph
```

Discussion
Japanese fonts encoded with the scheme "EUC12-NJE-CFEncoding" do not have individual metrics or bounding boxes available for the glyphs above 127. For those glyphs, this method returns the bounding rectangle for the font instead.

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

See Also
- `boundingRectForFont` (page 25)
- `getBoundingRects:forGlyphs:count:` (page 30)
- `getAdvancements:forGlyphs:count:` (page 29)
- `getAdvancements:forPackedGlyphs:length:` (page 29)

Declared In
`NSFont.h`

capHeight

Returns the receiver's cap height.

```
- (CGFloat)capHeight
```

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

See Also
- `ascender` (page 25)
- `descender` (page 27)
- `xHeight` (page 38)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

coveredCharacterSet

Returns an `NSCharacterSet` object containing all of the nominal characters renderable by the receiver, which is all of the entries mapped in the receiver's '`cmap`' table.

```
- (NSCharacterSet *)coveredCharacterSet
```

Return Value
An `NSCharacterSet` object containing all of the nominal characters renderable by the receiver.

Discussion
The number of glyphs supported by a given font is often larger than the number of characters contained in the character set returned by this method.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSFont.h`

descender

Returns the bottom y coordinate, offset from the baseline, of the receiver's longest descender.

```
- (CGFloat)descender
```

Discussion
Thus, if the longest descender extends 2 points below the baseline, `descender` will return –2.

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

Related Sample Code
DockTile

NSFontAttributeExplorer

SpeedometerView

WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In
`NSFont.h`

displayName

Returns the name, including family and face, used to represent the font in the user interface, typically localized for the user's language.

```
- (NSString *)displayName
```

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

Related Sample Code
NSFontAttributeExplorer

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In
`NSFont.h`

familyName

Returns the receiver's family name—for example, "Times" or "Helvetica."

```
- (NSString *)familyName
```

Discussion
This name is the one that `NSFontManager` uses and may differ slightly from the AFM name.

The value returned by this method is intended for an application's internal usage and not for display. Use `displayName` (page 27) instead.

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

See Also
`- fontName` (page 29)

Declared In
`NSFont.h`

fontDescriptor

Returns the receiver's font descriptor.

```
- (NSFontDescriptor *)fontDescriptor
```

Return Value
A font descriptor object that describes the receiver.

Discussion
The font descriptor contains a mutable dictionary of optional attributes for creating an `NSFont` object. See documentation on NSFontDescriptor for more information.

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

Declared In
`NSFont.h`

fontName

Returns the receiver's full font name, as used in PostScript language code—for example, "Times-Roman" or "Helvetica-Oblique."

```
- (NSString *)fontName
```

Discussion
The value returned by this method is intended for an application's internal usage and not for display. Use `displayName` (page 27) instead.

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

See Also
- `familyName` (page 28)

Declared In
`NSFont.h`

getAdvancements:forGlyphs:count:

Returns an array of the advancements for the specified glyphs rendered by the receiver.

```
- (void)getAdvancements:(NSSizeArray)advancements forGlyphs:(const NSGlyph *)glyphs
     count:(NSUInteger)glyphCount
```

Discussion
Returns in *advancements* an array of the advancements for the glyphs specified by *glyphs* and rendered by the receiver. The *glyphCount* must specify the count of glyphs passed in *glyphs*.

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

See Also
- `boundingRectForFont` (page 25)
- `boundingRectForGlyph:` (page 26)
- `getAdvancements:forPackedGlyphs:length:` (page 29)
- `getBoundingRects:forGlyphs:count:` (page 30)

Declared In
`NSFont.h`

getAdvancements:forPackedGlyphs:length:

Returns an array of the advancements for the specified packed glyphs and rendered by the receiver.

```
- (void)getAdvancements:(NSSizeArray)advancements forPackedGlyphs:(const void
    *)packedGlyphs length:(NSUInteger)length- (void)getAdvancements
```

Discussion
Returns in *advancements* an array of the advancements for the packed glyphs specified by *packedGlyphs* and rendered by the receiver. The *glyphCount* must specify the count of glyphs passed in *packedGlyphs*.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `– boundingRectForFont` (page 25)
- `– boundingRectForGlyph:` (page 26)
- `– getBoundingRects:forGlyphs:count:` (page 30)
- `– getAdvancements:forGlyphs:count:` (page 29)

Declared In

`NSFont.h`

getBoundingRects:forGlyphs:count:

Returns an array of the bounding rectangles for the specified glyphs rendered by the receiver.

```
- (void)getBoundingRects:(NSRectArray)bounds forGlyphs:(const NSGlyph *)glyphs
    count:(NSUInteger)glyphCount
```

Discussion

Returns in *bounds* an array of the bounding rectangles for the glyphs specified by *glyphs* and rendered by the receiver. The *glyphCount* must specify the count of glyphs passed in *glyphs*.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `– boundingRectForFont` (page 25)
- `– boundingRectForGlyph:` (page 26)
- `– getAdvancements:forGlyphs:count:` (page 29)
- `– getAdvancements:forPackedGlyphs:length:` (page 29)

Declared In

`NSFont.h`

glyphWithName:

Returns the named encoded glyph, or –1 if the receiver contains no such glyph.

```
- (NSGlyph)glyphWithName:(NSString *)glyphName
```

Parameters

glyphName

The name of the glyph.

Return Value

The named encoded glyph.

Discussion

Returns –1 if the glyph named *glyphName* isn’t encoded.

Glyph names in fonts do not always accurately identify the glyph. If possible, look up the appropriate glyph on your own.

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

Declared In
`NSFont.h`

isFixedPitch

Returns a Boolean value indicating whether all glyphs in the receiver have the same advancement.

```
- (BOOL)isFixedPitch
```

Return Value
`YES` if all glyphs in the receiver have the same advancement; `NO` if any advancements differ.

Discussion
Some Japanese fonts encoded with the scheme "EUC12-NJE-CFEncoding" return that they have the same advancement, but actually encode glyphs with one of two advancements, for historical compatibility. You may need to handle such fonts specially for some applications.

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

See Also
`- advancementForGlyph:` (page 24)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

italicAngle

Returns the receiver's italic angle, the amount that the font is slanted in degrees counterclockwise from the vertical, as read from its AFM file. Because the slant is measured counterclockwise, English italic fonts typically return a negative value.

```
- (CGFloat)italicAngle
```

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

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

leading

Returns the receiver's leading.

```
- (CGFloat)leading
```

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

See Also
+ fontWithDescriptor:size: (page 14)

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

matrix

Returns the receiver's font matrix, a standard six-element transformation matrix as used in the PostScript language, specifically with the makefont operator.

```
- (const CGFloat *)matrix
```

Discussion
In most cases, with a font of fontSize, this matrix is [fontSize 0 0 fontSize 0 0].

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

See Also
+ fontWithDescriptor:size: (page 14)

Declared In
NSFont.h

maximumAdvancement

Returns the greatest advancement of any of the receiver's glyphs.

```
- (NSSize)maximumAdvancement
```

Discussion
This advancement is always either strictly horizontal or strictly vertical.

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

See Also
- advancementForGlyph: (page 24)

Related Sample Code
NSFontAttributeExplorer

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In
`NSFont.h`

mostCompatibleStringEncoding

Returns the string encoding that works best with the receiver, where there are the fewest possible unmatched characters in the string encoding and glyphs in the font.

```
- (NSStringEncoding)mostCompatibleStringEncoding
```

Return Value
The string encoding that works best with the receiver.

Discussion
You can use `NSString`'s `dataUsingEncoding:` or `dataUsingEncoding:allowLossyConversion:` method to convert the string to this encoding.

If this method returns `NSASCIIStringEncoding`, it could not determine the correct encoding and assumed that the font can render only ASCII characters.

This method works heuristically using well-known font encodings, so for nonstandard encodings it may not in fact return the optimal string encoding.

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

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

numberOfGlyphs

Returns the number of glyphs in the receiver.

```
- (NSUInteger)numberOfGlyphs
```

Discussion
Glyphs are numbered starting at 0.

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

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

pointSize

Returns the receiver's point size, or the effective vertical point size for a font with a nonstandard matrix.

```
- (CGFloat)pointSize
```

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

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In
`NSFont.h`

printerFont

Returns the scalable PostScript font corresponding to itself.

```
- (NSFont *)printerFont
```

Discussion
When sent to a font object representing a scalable PostScript font, returns `self`. When sent to a font object representing a bitmapped screen font, returns its corresponding scalable PostScript font.

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

See Also
`- screenFont` (page 35)

Related Sample Code
DockTile

SpeedometerView

WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In
`NSFont.h`

renderingMode

Returns the rendering mode of the receiver.

```
- (NSFontRenderingMode)renderingMode
```

Return Value
The rendering mode of the receiver.

Discussion
For valid rendering modes, see "Constants" (page 38).

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

See Also
`- screenFontWithRenderingMode:` (page 35)

Declared In
`NSFont.h`

screenFont

Returns the bitmapped screen font corresponding to itself.

```
- (NSFont *)screenFont
```

Discussion
When sent to a font object representing a scalable PostScript font, returns a bitmapped screen font matching the receiver in typeface and matrix (or size), or `nil` if such a font can't be found. When sent to a font object representing a bitmapped screen font, returns `nil`.

Screen fonts are for direct use with the window server only. Never use them with Application Kit objects, such as in `setFont:` methods. Internally, the Application Kit automatically uses the corresponding screen font for a font object as long as the view is not rotated or scaled.

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

See Also
`- printerFont` (page 34)
`- screenFontWithRenderingMode:` (page 35)

Declared In
`NSFont.h`

screenFontWithRenderingMode:

Returns a bitmapped screen font, when sent to a font object representing a scalable PostScript font, with the specified rendering mode, matching the receiver in typeface and matrix (or size), or `nil` if such a font can't be found.

```
- (NSFont *)screenFontWithRenderingMode:(NSFontRenderingMode)renderingMode
```

Discussion
For valid rendering modes, see `NSFontRenderingMode` (page 38).

Screen fonts are for direct use with the window server only. Never use them with Application Kit objects, such as in `setFont:` methods. Internally, the Application Kit automatically uses the corresponding screen font for a font object as long as the view is not rotated or scaled.

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

See Also

- `printerFont` (page 34)
- `screenFont` (page 35)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

`NSFont.h`

set

Establishes the receiver as the current font for PostScript `show` and other text-drawing operators.

```
- (void)set
```

Discussion

During a print operation, also records the font as used in the PostScript code emitted.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ `useFont:` (page 48)

- `setInContext:` (page 36)

Declared In

`NSFont.h`

setInContext:

Establishes the receiver as the current font for the specified graphics context.

```
- (void)setInContext:(NSGraphicsContext *)graphicsContext
```

Parameters

graphicsContext

The graphics context for which the font is set.

Availability

Available in Mac OS X v10.4 and later.

See Also

+ `useFont:` (page 48)

- `set` (page 36)

Declared In

`NSFont.h`

textTransform

Returns the current transformation matrix for the receiver.

```
- (NSAffineTransform *)textTransform
```

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

See Also
`+ useFont:` (page 48)
`- set` (page 36)

Declared In
`NSFont.h`

underlinePosition

Returns the baseline offset that should be used when drawing underlines with the receiver, as determined by the font's AFM file.

```
- (CGFloat)underlinePosition
```

Discussion
This value is usually negative, which must be considered when drawing in a flipped coordinate system.

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

See Also
`- underlineThickness` (page 37)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

underlineThickness

Returns the thickness that should be used when drawing underlines with the receiver, as determined by the font's AFM file.

```
- (CGFloat)underlineThickness
```

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

See Also
`- underlinePosition` (page 37)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

xHeight

Returns the x-height of the receiver.

```
- (CGFloat)xHeight
```

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

See Also
- `– ascender` (page 25)
- `– descender` (page 27)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

Constants

NSFontRenderingMode

These constants specify the font rendering mode.

```
typedef enum {
    NSFontDefaultRenderingMode = 0,
    NSFontAntialiasedRenderingMode = 1,
    NSFontIntegerAdvancementsRenderingMode = 2,
    NSFontAntialiasedIntegerAdvancementsRenderingMode = 3
} NSFontRenderingMode;
```

Constants

`NSFontDefaultRenderingMode`

Determines the actual mode based on the user preference settings.

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

`NSFontAntialiasedRenderingMode`

Specifies antialiased, floating-point advancements rendering mode (synonymous with printerFont).

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

`NSFontIntegerAdvancementsRenderingMode`

Specifies integer advancements rendering mode.

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

`NSFontAntialiasedIntegerAdvancementsRenderingMode`

Specifies antialiased, integer advancements rendering mode.

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

Declared In

`NSFont.h`

PostScript Transformation Matrix

The identity matrix.

```
const float *NSFontIdentityMatrix;
```

Constants

`NSFontIdentityMatrix`

A transformation matrix useful as a parameter to fontWithDescriptor:size: (page 14).

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Declared In

`NSFont.h`

NSMultibyteGlyphPacking

A constant for glyph packing.

```
typedef enum {
    NSNativeShortGlyphPacking = 5
} NSMultibyteGlyphPacking;
```

Constants

`NSNativeShortGlyphPacking`

The native format for Mac OS X.

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Discussion

Cocoa stores all text data as Unicode. The text system converts Unicode into glyph IDs and places them in 1-, 2-, or 4-byte storage depending on the context. To render text, you must convert the storage into a format the text engine understands. The following constants describe the glyph packing schemes the text rendering engine can use. They are used to extract glyphs from a font for making a multibyte (or single-byte) array of glyphs for passing to an interpreter, such as the window server, which expects a big-endian multibyte stream (that is, “packed glyphs”) instead of a pure `NSGlyph` stream. They’re used by `glyphPacking` (page 50). With Quartz, the engine always expects the format to be in 2-byte short array, so `NSNativeShortGlyphPacking` is the only format currently in use.

Declared In

`NSFont.h`

Reserved Glyph Codes

These constants define reserved glyph codes.

```
enum {
   NSControlGlyph = 0x00FFFFFF,
   NSNullGlyph = 0x0
};
```

Constants

`NSControlGlyph`

`NSGlyphGenerator` generates `NSControlGlyph` for all characters in the Unicode General Category C* and U200B (ZERO WIDTH SPACE).

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

`NSNullGlyph`

A null glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Declared In

`NSFont.h`

Keys to the AFM Dictionary

These constants are used as keys retrieve information from an AFM dictionary. (**Deprecated.** The AFM dictionary is no longer used in Mac OS X. Use the font metrics accessor methods listed with the individual constants instead.)

```
NSString *NSAFMFamilyName;
NSString *NSAFMFontName;
NSString *NSAFMFormatVersion;
NSString *NSAFMFullName;
NSString *NSAFMNotice;
NSString *NSAFMVersion;
NSString *NSAFMWeight;
NSString *NSAFMEncodingScheme;
NSString *NSAFMCharacterSet;
NSString *NSAFMCapHeight;
NSString *NSAFMXHeight;
NSString *NSAFMAscender;
NSString *NSAFMDescender;
NSString *NSAFMUnderlinePosition;
NSString *NSAFMUnderlineThickness;
NSString *NSAFMItalicAngle;
NSString *NSAFMMappingScheme;
```

Constants

`NSAFMFamilyName`

Font family name key. (**Deprecated.** Use `familyName` (page 28) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMFontName`

Font name key. (**Deprecated.** Use `displayName` (page 27) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMFormatVersion`

Format version name key. (**Deprecated.** This information is not relevant to Mac OS X.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMFullName`

Full font name key. (**Deprecated.** Use `fontName` (page 29) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMNotice`

Font notice key. (**Deprecated.** Use Apple Type Services instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMVersion`

Font version key. (**Deprecated.** Use Apple Type Services instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMWeight`

Font weight key. (**Deprecated.** Use the `NSFontManager` method `weightOfFont:` instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMEncodingScheme`

Font encoding scheme key. (**Deprecated.** Use `mostCompatibleStringEncoding` (page 33) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMCharacterSet`

Font character set key. (**Deprecated.** Use `coveredCharacterSet` (page 27) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMCapHeight`

Font cap-height key. (**Deprecated.** Use `capHeight` (page 26) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMXHeight`

Font x-height key. (**Deprecated.** Use `xHeight` (page 38) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMAscender`

Font ascender height key. (**Deprecated.** Use `ascender` (page 25) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMDescender`

Font descender depth key. (**Deprecated.** Use `descender` (page 27) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMUnderlinePosition`

Font underline rule position key. (**Deprecated.** Use `underlinePosition` (page 37) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMUnderlineThickness`

Font underline rule thickness key. (**Deprecated.** Use `underlineThickness` (page 37) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMItalicAngle`

Font italic angle key. (**Deprecated.** Use `italicAngle` (page 31) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMMappingScheme`

Font mapping scheme key. (**Deprecated.** This information is irrelevant to Mac OS X.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

Declared In
`NSFont.h`

NSGlyph

This type is used to specify glyphs in such methods as `glyphWithName:`.

```
typedef unsigned int NSGlyph;
```

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

Declared In
`NSFont.h`

NSGlyphRelation

These constants are used for calculating the layout of stacked glyphs. (**Deprecated.** Context-sensitive interglyph spacing is now performed at the typesetting stage)

```
typedef enum _NSGlyphRelation {
    NSGlyphBelow = 1,
    NSGlyphAbove = 2
} NSGlyphRelation;
```

Constants

`NSGlyphBelow`

The glyph is located below the base glyph. (**Deprecated.** Context-sensitive interglyph spacing is now performed at the typesetting stage.)

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

`NSGlyphAbove`

The glyph is located above the base glyph. (**Deprecated.** Context-sensitive interglyph spacing is now performed at the typesetting stage.)

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In

`NSFont.h`

NSMultibyteGlyphPacking

Glyph packing constants are used to extract glyphs from a font for making a multibyte (or single-byte) array of glyphs for passing to an interpreter, such as the window server. With Quartz, the engine always expects the format to be in 2-byte short array, so `NSNativeShortGlyphPacking` (page 39) is the only format currently in use. (**Deprecated.** Use `NSNativeShortGlyphPacking` (page 39) instead.)

```
enum {
    NSOneByteGlyphPacking,
    NSJapaneseEUCGlyphPacking,
    NSAsciiWithDoubleByteEUCGlyphPacking,
    NSTwoByteGlyphPacking,
    NSFourByteGlyphPacking,
}
```

Constants

`NSOneByteGlyphPacking`

One-byte storage format. (**Deprecated.** Use `NSNativeShortGlyphPacking` (page 39) instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

`NSJapaneseEUCGlyphPacking`

Extended Unix Code for Japanese format. (**Deprecated.** Use `NSNativeShortGlyphPacking` (page 39) instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

`NSAsciiWithDoubleByteEUCGlyphPacking`

Two-byte Extended Unix Code format. (**Deprecated.** Use `NSNativeShortGlyphPacking` (page 39) instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

`NSTwoByteGlyphPacking`

Two-byte storage format. (**Deprecated.** Use `NSNativeShortGlyphPacking` (page 39) instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

`NSFourByteGlyphPacking`

Four-byte storage format. (**Deprecated.** Use `NSNativeShortGlyphPacking` (page 39) instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Declared In

`NSFont.h`

Notifications

NSAntialiasThresholdChangedNotification

Posted after the threshold for anti-aliasing changes.

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

Declared In
`NSFont.h`

NSFontSetChangedNotification

Posted after the the currently-set font changes.

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

Declared In
`NSFont.h`

Deprecated NSFont Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.4

preferredFontNames

Returns the names of fonts that the Application Kit tries first when a character has no font specified. (Deprecated in Mac OS X v10.4. The `NSFontDescriptor` constant `NSFontCascadeListAttribute` offers more powerful font substitution management.)

```
+ (NSArray *)preferredFontNames
```

Discussion
Returns the names of fonts that the Application Kit tries first when a character has no font specified or when the font specified doesn't have a glyph for that character. If none of these fonts provides a glyph, the remaining fonts on the system are searched for a glyph.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

See Also
`+ setPreferredFontNames:` (page 47)

Declared In
`NSFont.h`

setPreferredFontNames:

Sets the list of preferred font names. (Deprecated in Mac OS X v10.4. The `NSFontDescriptor` constant `NSFontCascadeListAttribute` offers more powerful font substitution management.)

```
+ (void)setPreferredFontNames:(NSArray *)fontNames
```

Discussion
Sets the list of preferred font names to *fontNames* and records them in the user defaults database for all applications. The Application Kit tries these fonts first when a character has no font specified or when the font specified doesn't have a glyph for that character. If none of these fonts provides a glyph, the remaining fonts on the system are searched for a glyph.

This method is useful for optimizing glyph rendering for uncommon scripts, by guaranteeing that appropriate fonts are searched first. For example, suppose you have three hundred Latin alphabet fonts and one Cyrillic alphabet font. When you read a document in Russian, you want it to find the Cyrillic font quickly. Ordinarily, the Application Kit will search for the Cyrillic font among all 301 fonts. But if it is in the list of preferred fonts, the Cyrillic font will be one of the first searched.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

See Also
`+ preferredFontNames` (page 47)

Declared In
`NSFont.h`

useFont:

Records the given font name as one used in the current print operation. (Deprecated in Mac OS X v10.4. This is now automatically handled by Quartz.)

```
+ (void)useFont:(NSString *)fontName
```

Discussion
Records *fontName* as one used in the current print operation.

The `NSFont` class object keeps track of the fonts used in an `NSView` by recording each one that receives a `set` (page 36) message. When the view is called upon to generate conforming PostScript language output (such as during printing), the `NSFont` class provides the list of fonts required for the `%%DocumentFonts` comment, as required by Adobe's document structuring conventions.

The `useFont:` argument augments this system by providing a way to register fonts that are included in the document but not set using `NSFont`'s `set` (page 36) method. For example, you might set a font by executing the `setfont` operator within a function created by the `pswrap` utility. In such a case, be sure to pair the use of the font with a `useFont:` message to register the font for listing in the document comments.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`NSFont.h`

afmDictionary

Returns the AFM font's dictionary. (Deprecated in Mac OS X v10.4. Use accessor functions listed in "`Keys to the AFM Dictionary`" (page 40) instead.)

```
- (NSDictionary *)afmDictionary
```

Discussion

Always returns `nil`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`NSFont.h`

defaultLineHeightForFont

Returns the default line height for the receiver. (Deprecated in Mac OS X v10.4. Use the `NSLayoutManager` method `defaultLineHeightForFont:` instead.)

```
- (CGFloat)defaultLineHeightForFont
```

Discussion

Equivalent to ascent plus descent plus linegap.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

See Also

`– ascender` (page 25)

`– descender` (page 27)

Declared In

`NSFont.h`

encodingScheme

Returns the name of the receiver's encoding scheme. (Deprecated in Mac OS X v10.4. Use `mostCompatibleStringEncoding` (page 33) instead.)

```
- (NSString *)encodingScheme
```

Discussion

Returns the name of the receiver's encoding scheme, such as "AdobeStandardEncoding," "ISOLatin1Encoding," "FontSpecific," and so on.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`NSFont.h`

glyphIsEncoded:

Returns a Boolean value indicating whether the receiver encodes the given glyph. (Deprecated in Mac OS X v10.4. The value can be deduced by `aGlyph < [NSFont numberOfGlyphs]` since only `NSNativeShortGlyphPacking` is supported.)

```
- (BOOL)glyphIsEncoded:(NSGlyph)aGlyph
```

Discussion
Returns `YES` if the receiver encodes *aGlyph*, `NO` if it doesn't contain it.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`NSFont.h`

glyphPacking

Returns the best way to encode the receiver's glyphs into an array of bytes. (Deprecated in Mac OS X v10.4. Only `NSNativeShortGlyphPacking` (page 39) is supported.)

```
- (NSMultibyteGlyphPacking)glyphPacking
```

Discussion
Returns the best way to encode the receiver's glyphs into an array of bytes. The return value is one of values described in "Constants" (page 38).

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`NSFont.h`

isBaseFont

Returns a Boolean value indicating whether the receiver is a PostScript base font. (Deprecated in Mac OS X v10.4. This information is not relevant to Mac OS X.)

```
- (BOOL)isBaseFont
```

Discussion
Returns `YES` if the receiver is a PostScript base font, `NO` if it's a PostScript composite font composed of other base fonts.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`NSFont.h`

positionOfGlyph:forCharacter:struckOverRect:

Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph forCharacter:(unichar)aChar
    struckOverRect:(NSRect)aRect
```

Discussion
Calculates and returns a suitable location for *`aGlyph`* to be drawn as a diacritic or nonspacing mark relative to *`aRect`*, assuming that *`aGlyph`* represents *`aChar`*. Returns `NSZeroPoint` if the location can't be calculated. The nature of *`aChar`* as one appearing above or below its base character determines the location returned. For example, in the first figure below, the gray tilde and box represent *`aGlyph`* and *`aRect`*, and the black dot is the point returned (defined relative to the origin of the *`aRect`*).

To place multiple glyphs with respect to a rectangle, work from the innermost glyphs to the outermost. As you calculate the position of each glyph, enlarge the rectangle to include the bounding rectangle of the glyph in preparation for the next glyph. The second figure shows a tilde, acute accent, and cedilla all placed in their appropriate positions with respect to a rectangle, with the acute accent placed relative to the expanded bounding box of the base rectangle and the tilde.

This method is the last fallback mechanism for performing minimally legible typography when metrics aren't available. Use it when `positionOfGlyph:struckOverGlyph:metricsExist:` (page 52) indicates that metrics don't exist for the base glyph specified, or when you are combining glyphs from different fonts (for example, the base glyph is in a different font than the accent). It can account for the layout and placement of most Latin, Greek, and Cyrillic nonspacing marks. You should draw the glyph at the returned location, even if it's `NSZeroRect`.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`NSFont.h`

positionOfGlyph:precededByGlyph:isNominal:

Calculates and returns the location of a glyph. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph precededByGlyph:(NSGlyph)prevGlyph
    isNominal:(BOOL *)flag
```

Discussion
Calculates and returns the location of *`aGlyph`* relative to *`prevGlyph`*, assuming that *`prevGlyph`* precedes it in the layout (not necessarily in the character stream). The point returned should be used relative to whatever location is used for *`prevGlyph`*. If *`flag`* is non-`nil`, it's filled with `NO` if kerning tables are available and were used in the calculation; it is filled with `YES` if the default spacing is used.

Returns `NSZeroPoint` if either *aGlyph* or *prevGlyph* is `NSControlGlyph` or is invalid. Returns the nominal advancement of *prevGlyph* if *aGlyph* is `NSNullGlyph`.

This method is useful for sequential glyph placement when glyphs aren't drawn with a single PostScript operation.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`NSFont.h`

positionOfGlyph:struckOverGlyph:metricsExist:

Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph struckOverGlyph:(NSGlyph)baseGlyph
    metricsExist:(BOOL *)flag
```

Discussion
Calculates and returns a suitable location for *aGlyph* to be drawn as a diacritic or nonspacing mark relative to *baseGlyph*. The point returned should be used relative to whatever location is used for *baseGlyph*. If *flag* is non-`nil` it's filled with `YES` if font metrics are available, `NO` if they're not. If flag is returned as `NO`, the result isn't valid and shouldn't be used. In that case, use `positionOfGlyph:struckOverRect:metricsExist:` (page 52) or `positionOfGlyph:forCharacter:struckOverRect:` (page 51) to calculate a reasonable offset.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

See Also
`– positionsForCompositeSequence:numberOfGlyphs:pointArray:` (page 54)

`– positionOfGlyph:struckOverRect:metricsExist:` (page 52)

Declared In
`NSFont.h`

positionOfGlyph:struckOverRect:metricsExist:

Overridden by subclasses to calculate and return a suitable location for a glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph struckOverRect:(NSRect)aRect
    metricsExist:(BOOL *)flag
```

Discussion

Overridden by subclasses to calculate and return a suitable location for *aGlyph* to be drawn as a diacritic or nonspacing mark relative to *aRect*, provided metrics exist. Returns `NSZeroRect` if the location can't be determined. If *flag* is non-`nil` it's filled with `YES` if font metrics are available, `NO` if they're not. If *flag* is returned as `NO`, the result isn't valid and shouldn't be used. In that case, use `positionOfGlyph:forCharacter:struckOverRect:` (page 51) to calculate a reasonable offset.

Because current PostScript font metrics don't include support for generic placement relative to rectangles, `NSFont`'s implementation of this method always returns `NSZeroPoint` and returns *flag* as `NO`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`NSFont.h`

positionOfGlyph:withRelation:toBaseGlyph:totalAdvancement:metricsExist:

Calculates and returns a suitable location for a glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph withRelation:(NSGlyphRelation)relation
     toBaseGlyph:(NSGlyph)baseGlyph totalAdvancement:(NSSizePointer)offset
    metricsExist:(BOOL *)flag
```

Discussion

Calculates and returns a suitable location for *aGlyph* to be drawn relative to *baseGlyph*, where *relation* is `NSGlyphBelow` or `NSGlyphAbove`. The point returned should be used relative to whatever location is used for *baseGlyph*. This method is useful for calculating the layout of stacked glyphs, found in some non-Western scripts.

If *offset* is non-`NULL`, this method sets it to the larger of the two glyphs' advancements, allowing for reasonable layout of following glyphs.

If *flag* is non-`nil`, this method sets it to whether font metrics are available: `YES` if they are, `NO` if they're not. If metrics aren't available, the location is calculated as a simple stacking with no gap between *baseGlyph* and *aGlyph*. Current Postscript fonts do not contain appropriate font metrics, so this method always sets *flag* to `NO`. If you subclass `NSFont` to handle fonts that do contain metrics, override this method.

This method supports only horizontally laid out base glyphs.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`NSFont.h`

positionsForCompositeSequence:numberOfGlyphs:pointArray:

Calculates glyph locations. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSInteger)positionsForCompositeSequence:(NSGlyph *)glyphs
    numberOfGlyphs:(NSInteger)numGlyphs pointArray:(NSPointArray)points
```

Discussion

Calculates and fills `points` with the locations for `glyphs`, assuming the first glyph is a base character and those following are nonspacing marks. These points should all be interpreted as relative to the location of the first glyph in `glyphs`. The storage block `points` points to should be large enough for at least `numGlyphs` points. Returns the number of points that could be calculated.

If the number of points calculated is less than `numGlyphs`, the number of glyphs provided, you can use `positionOfGlyph:struckOverRect:metricsExist:` (page 52) to determine the positions for the remaining glyphs. When using that method, calculate the base rectangle for each glyph from the bounding rectangles and positions of all preceding glyphs.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`NSFont.h`

widthOfString:

Returns the x-axis offset of the current point when the specified string is drawn with a `show` operator in the receiving font. (Deprecated in Mac OS X v10.4. Use the Application Kit string-drawing methods, as described in NSString Additions.)

```
- (CGFloat)widthOfString:(NSString *)aString
```

Discussion

This method is for backward compatibility only. This method performs lossy conversion of `aString` to the most compatible encoding for the receiving font. Use this method only when you're sure all of `aString` can be rendered with the receiving font.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

See Also

`- mostCompatibleStringEncoding` (page 33)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

`NSFont.h`

Document Revision History

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

Date	Notes
2007-06-08	Removed superfluous colon from title of textTransform method and size: parameter from fontWithDescriptor:textTransform:.
2006-07-24	Corrected information for deprecated constants, reformatted parameter descriptions, and made minor revisions throughout.
2006-05-23	First publication of this content as a separate document.

Index