NSTypesetter Class Reference

Cocoa > Text & Fonts

2007-03-26

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

Simultaneously published in the United States and Canada.

Contents

NSTypesetter Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSTypesetter.h
Companion guides	Text System Overview Text Layout Programming Guide for Cocoa

Overview

`NSLayoutManager` uses concrete subclasses of this abstract class, `NSTypesetter`, to perform line layout, which includes word wrapping, hyphenation, and line breaking in either vertical or horizontal rectangles. By default, the text system uses the concrete subclass `NSATSTypesetter`.

Subclassing Notes

`NSTypesetter` provides concrete subclasses with default implementation interfacing with the Cocoa text system. By subclassing `NSTypesetter`, an application can override the `layoutParagraphAtPoint:` (page 26) method to integrate a custom typesetting engine into the Cocoa text system. On the other hand, an application can subclass `NSATSTypesetter` and override the glyph storage interface to integrate the concrete subclass into its own custom layout system.

`NSTypesetter` methods belong to three categories: glyph storage interface methods, layout phase interface methods, and core typesetter methods. The glyph storage interface methods map to `NSLayoutManager` methods. The typesetter itself calls these methods, and their default implementations call the Cocoa layout manager. An `NSTypesetter` subclass can override these methods to call its own glyph storage facility, in which case it should override all of them. (This does not preclude the overridden method calling its superclass implementation if appropriate.)

The layout phase interface provides control points similar to delegate methods; if implemented, the system invokes these methods to notify an `NSTypesetter` subclass of events in the layout process so it can intervene as needed.

The remainder of the `NSTypesetter` methods are primitive, core typesetter methods. The core typesetter methods correlate with typesetting state attributes; the layout manager calls these methods to store its values before starting the layout process. If you subclass `NSTypesetter` and override the glyph storage interface methods, you can call the core methods to control the typesetter directly.

Glyph Storage Interface

Override these methods to use `NSTypesetter`'s built-in concrete subclass, `NSATSTypesetter`, with a custom glyph storage and layout system other than the Cocoa layout manager and text container mechanism.

`characterRangeForGlyphRange:actualGlyphRange:` (page 18)
`glyphRangeForCharacterRange:actualCharacterRange:` (page 23)
`getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels:` (page 20)
`getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:` (page 22)
`setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 34)
`substituteGlyphsInRange:withGlyphs:` (page 38)
`insertGlyph:atGlyphIndex:characterIndex:` (page 24)
`deleteGlyphsInRange:` (page 19)
`setNotShownAttribute:forGlyphRange:` (page 35)
`setDrawsOutsideLineFragment:forGlyphRange:` (page 32)
`setLocation:withAdvancements:forStartOfGlyphRange:` (page 34)
`setAttachmentSize:forGlyphRange:` (page 30)
`setBidiLevels:forGlyphRange:` (page 31)

Layout Phase Interface

Override these methods to customize the text layout process, including modifying line fragments, controlling line breaking and hyphenation, and controlling the behavior of tabs and other control glyphs.

`willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 40)
`shouldBreakLineByWordBeforeCharacterAtIndex:` (page 37)
`shouldBreakLineByHyphenatingBeforeCharacterAtIndex:` (page 37)
`hyphenationFactorForGlyphAtIndex:` (page 23)
`hyphenCharacterForGlyphAtIndex:` (page 24)
`boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 17)

Tasks

Getting a Typesetter

`+ sharedSystemTypesetter` (page 14)
Returns a shared instance of a reentrant typesetter.

`+ sharedSystemTypesetterForBehavior:` (page 14)
Returns a shared instance of a reentrant typesetter that implements typesetting with the specified behavior.

Getting Information About a Typesetter

`+ defaultTypesetterBehavior` (page 13)
Returns the default typesetter behavior.

Getting Information About Glyphs

`+ printingAdjustmentInLayoutManager:forNominallySpacedGlyphRange:packedGlyphs:count:` (page 13)
Returns the interglyph spacing in the specified range when sent to a printer.

`- baselineOffsetInLayoutManager:glyphIndex:` (page 15)
Returns the distance from the bottom of the bounding box of a specified glyph to its baseline.

Managing the Layout Manager

`- layoutManager` (page 26)
Returns the layout manager for the text being typeset.

`- setUsesFontLeading:` (page 36)
Sets whether the typesetter uses the leading (or line gap) value specified in the font metric information.

`- usesFontLeading` (page 40)
Returns whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.

`- setTypesetterBehavior:` (page 36)
Sets the default typesetter behavior, which affects glyph spacing and line height.

`- typesetterBehavior` (page 39)
Returns the current typesetter behavior.

`- setHyphenationFactor:` (page 33)
Sets the threshold controlling when hyphenation is attempted.

`- hyphenationFactor` (page 23)
Returns the current hyphenation factor.

Managing Text Containers

`- currentTextContainer` (page 19)
Returns the text container for the text being typeset.

`- textContainers` (page 39)
Returns an array containing the text containers belonging to the current layout manager.

`- setLineFragmentPadding:` (page 33)
Sets the amount (in points) by which text is inset within line fragment rectangles.

- `lineFragmentPadding` (page 26)
  Returns the current line fragment padding, in points.

Mapping Screen and Printer Fonts

- `substituteFontForFont:` (page 38)
  Returns a screen font suitable for use in place of a given font.

Handling Control Characters

- `textTabForGlyphLocation:writingDirection:maxLocation:` (page 39)
  Returns the text tab next closest to a given glyph location within the given parameters.
- `actionForControlCharacterAtIndex:` (page 14)
  Returns the action associated with a control character.

Bidirectional Text Processing

- `setBidiProcessingEnabled:` (page 32)
  Controls whether the typesetter performs bidirectional text processing.
- `bidiProcessingEnabled` (page 17)
  Returns whether bidirectional text processing is enabled.

Accessing Paragraph Typesetting Information

- `currentParagraphStyle` (page 18)
  Returns the paragraph style object for the text being typeset.
- `setAttributedString:` (page 31)
  Sets the text backing store on which this typesetter operates.
- `attributedString` (page 15)
  Returns the text backing store, usually an instance of `NSTextStorage`.
- `setParagraphGlyphRange:separatorGlyphRange:` (page 35)
  Sets the current glyph range being processed.
- `paragraphGlyphRange` (page 28)
  Returns the glyph range currently being processed.
- `paragraphSeparatorGlyphRange` (page 29)
  Returns the current paragraph separator range.
- `paragraphCharacterRange` (page 27)
  Returns the character range currently being processed.
- `paragraphSeparatorCharacterRange` (page 28)
  Returns the current paragraph separator character range.
- `attributesForExtraLineFragment` (page 15)
  Returns the attributes used to lay out the extra line fragment.

Paragraph Layout

- `layoutParagraphAtPoint:` (page 26)
  Lays out glyphs in the current glyph range until the next paragraph separator is reached.
- `beginParagraph` (page 16)
  Sets up layout parameters at the beginning of a paragraph.
- `endParagraph` (page 20)
  Sets up layout parameters at the end of a paragraph.
- `beginLineWithGlyphAtIndex:` (page 16)
  Sets up layout parameters at the beginning of a line during typesetting.
- `endLineWithGlyphRange:` (page 20)
  Sets up layout parameters at the end of a line during typesetting.

Line and Paragraph Spacing

- `lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 27)
  Returns the line spacing in effect following the specified glyph.
- `paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 29)
  Returns the paragraph spacing that is in effect after the specified glyph.
- `paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:` (page 30)
  Returns the number of points of space—added before a paragraph—that is in effect before the specified glyph.

Glyph Caching

- `setHardInvalidation:forGlyphRange:` (page 32)
  Sets whether to force the layout manager to invalidate the specified portion of the glyph cache when invalidating layout.

Laying out Glyphs

- `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 25)
  Lays out glyphs in the specified layout manager starting at a specified glyph.
- `boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 17)
  Returns the bounding rectangle for the specified control glyph with the specified parameters.
- `getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:` (page 21)
  Calculates the line fragment rectangle and line fragment used rectangle for blank lines.
- `getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:` (page 22)
  Calculates line fragment rectangle, line fragment used rectangle, and remaining rectangle for a line fragment.

- `hyphenCharacterForGlyphAtIndex:` (page 24)
  Returns the hyphen character to be inserted after the specified glyph.
- `hyphenationFactorForGlyphAtIndex:` (page 23)
  Returns the hyphenation factor in effect at a specified location.
- `shouldBreakLineByHyphenatingBeforeCharacterAtIndex:` (page 37)
  Returns whether the line being laid out should be broken by hyphenating at the specified character.
- `shouldBreakLineByWordBeforeCharacterAtIndex:` (page 37)
  Returns whether the line being laid out should be broken by a word break at the specified character.
- `willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 40)
  Called by the typesetter just prior to storing the actual line fragment rectangle location in the layout manager.

Interfacing with Glyph Storage

- `characterRangeForGlyphRange:actualGlyphRange:` (page 18)
  Returns the range for the characters in the receiver's text store that are mapped to the specified glyphs.
- `deleteGlyphsInRange:` (page 19)
  Deletes the specified glyphs from the glyph cache maintained by the layout manager.
- `substituteGlyphsInRange:withGlyphs:` (page 38)
  Replaces the specified glyphs with specified replacement glyphs.
- `getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels:` (page 20)
  Extracts the information needed to lay out the provided glyphs from the provided range.
- `glyphRangeForCharacterRange:actualCharacterRange:` (page 23)
  Returns the range for the glyphs mapped to the characters of the text store in the specified range.
- `insertGlyph:atGlyphIndex:characterIndex:` (page 24)
  Enables the typesetter to insert a new glyph into the stream.
- `setAttachmentSize:forGlyphRange:` (page 30)
  Sets the size the specified glyphs (assumed to be attachments) will be asked to draw themselves at.
- `setBidiLevels:forGlyphRange:` (page 31)
  Sets the direction of the specified glyphs for bidirectional text.
- `setDrawsOutsideLineFragment:forGlyphRange:` (page 32)
  Sets whether the specified glyphs exceed the bounds of the line fragment in which they are laid out.
- `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 34)
  Sets the line fragment rectangle where the specified glyphs are laid out.
- `setLocation:withAdvancements:forStartOfGlyphRange:` (page 34)
  Sets the location where the specified glyphs are laid out.
- `setNotShownAttribute:forGlyphRange:` (page 35)
  Sets whether the specified glyphs are not shown.

Class Methods

defaultTypesetterBehavior

Returns the default typesetter behavior.

```
+ (NSTypesetterBehavior)defaultTypesetterBehavior
```

Return Value
The default typesetter behavior.

Discussion
Possible return values are described in the "Constants" section for `NSLayoutManager`.

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

Declared In
`NSTypesetter.h`

printingAdjustmentInLayoutManager:forNominallySpacedGlyphRange:packedGlyphs:count:

Returns the interglyph spacing in the specified range when sent to a printer.

```
+ (NSSize)printingAdjustmentInLayoutManager:(NSLayoutManager *)layoutMgr
    forNominallySpacedGlyphRange:(NSRange)nominallySpacedGlyphsRange
    packedGlyphs:(const unsigned char *)packedGlyphs
    count:(NSUInteger)packedGlyphsCount
```

Parameters

layoutMgr
The layout manager that will do the drawing.

nominallySpacedGlyphsRange
The range of the glyphs whose spacing is desired.

packedGlyphs
The glyphs as they are packed for sending to be drawn in *layoutMgr*.

packedGlyphsCount
The number of glyphs in *packedGlyphs*.

Return Value
The interglyph spacing in the specified range when sent to a printer. If the font metrics of the font used for displaying text on the screen is different from the font metrics of the font used in printing, then this interglyph spacing may need to be adjusted slightly to match that used on the screen.

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

Declared In
`NSTypesetter.h`

sharedSystemTypesetter

Returns a shared instance of a reentrant typesetter.

```
+ (id)sharedSystemTypesetter
```

Return Value
The shared system typesetter. This typesetter is reentrant.

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

Declared In
`NSTypesetter.h`

sharedSystemTypesetterForBehavior:

Returns a shared instance of a reentrant typesetter that implements typesetting with the specified behavior.

```
+ (id)sharedSystemTypesetterForBehavior:(NSTypesetterBehavior)theBehavior
```

Parameters
theBehavior
The desired behavior.

Return Value
A shared instance of a reentrant typesetter that implements typesetting with the specified behavior.

Discussion
Possible return values are described in the "Constants" section for `NSLayoutManager`.

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

See Also
- `setTypesetterBehavior:` (page 36)
- `typesetterBehavior` (page 39)

Declared In
`NSTypesetter.h`

Instance Methods

actionForControlCharacterAtIndex:

Returns the action associated with a control character.

```
- (NSTypesetterControlCharacterAction)actionForControlCharacterAtIndex:(NSUInteger)charIndex
```

Parameters

charIndex

The index of the control character.

Return Value

The action associated with the control character at *charIndex*.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTypesetter.h

attributedString

Returns the text backing store, usually an instance of NSTextStorage.

```
- (NSAttributedString *)attributedString
```

Return Value

The text backing store.

Availability

Available in Mac OS X v10.4 and later.

See Also

- setAttributedString: (page 31)

Declared In

NSTypesetter.h

attributesForExtraLineFragment

Returns the attributes used to lay out the extra line fragment.

```
- (NSDictionary *)attributesForExtraLineFragment
```

Return Value

A dictionary of attributes used to lay out the extra line fragment.

Discussion

The default implementation tries to use the NSTextView method typingAttributes if possible; otherwise, it uses the attributes for the last character.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTypesetter.h

baselineOffsetInLayoutManager:glyphIndex:

Returns the distance from the bottom of the bounding box of a specified glyph to its baseline.

```
- (CGFloat)baselineOffsetInLayoutManager:(NSLayoutManager *)layoutMgr
    glyphIndex:(NSUInteger)glyphIndex
```

Parameters

layoutMgr

The layout manager used for the drawing.

glyphIndex

The index of the glyph in question.

Return Value

The distance from the bottom of the bounding box of the glyph in *layoutMgr* specified by *glyphIndex* to its baseline.

Discussion

The text system uses this value to calculate the vertical position of underlines.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSTypesetter.h

beginLineWithGlyphAtIndex:

Sets up layout parameters at the beginning of a line during typesetting.

```
- (void)beginLineWithGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of the first glyph to be laid out in the line.

Discussion

Concrete subclass implementations of `layoutParagraphAtPoint:` (page 26) should invoke this method at the beginning of each line.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- endLineWithGlyphRange:` (page 20)

Declared In

NSTypesetter.h

beginParagraph

Sets up layout parameters at the beginning of a paragraph.

```
- (void)beginParagraph
```

Discussion

Concrete subclasses should invoke this method at the beginning of their `layoutParagraphAtPoint:` (page 26) implementation.

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

See Also
- endParagraph (page 20)

Declared In
NSTypesetter.h

bidiProcessingEnabled

Returns whether bidirectional text processing is enabled.

```
- (BOOL)bidiProcessingEnabled
```

Return Value
YES if bidirectional text processing is enabled, NO otherwise.

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

See Also
- setBidiProcessingEnabled: (page 32)

Declared In
NSTypesetter.h

boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:

Returns the bounding rectangle for the specified control glyph with the specified parameters.

```
- (NSRect)boundingBoxForControlGlyphAtIndex:(NSUInteger)glyphIndex
    forTextContainer:(NSTextContainer *)textContainer
    proposedLineFragment:(NSRect)proposedRect glyphPosition:(NSPoint)glyphPosition
     characterIndex:(NSUInteger)charIndex
```

Parameters

glyphIndex
The index of the control glyph in question.

textContainer
The text container to use to calculate the position.

proposedRect
The proposed line fragment rectangle.

glyphPosition
The position of the glyph in *textContainer*.

charIndex
The character index in *textContainer*.

Return Value
The bounding rectangle of the control glyph at *glyphIndex*, at the given *glyphPosition* and character index *charIndex*, in *textContainer*.

Discussion
The typesetter calls this method when it encounters a control glyph. The default behavior is to return zero width for control glyphs. A subclass can override this method to do something different, such as implement a way to display control characters.

`NSGlyphGenerator` can choose whether or not to map control characters to `NSControlGlyph`. Tab characters, for example, do not use this facility.

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

Declared In
`NSTypesetter.h`

characterRangeForGlyphRange:actualGlyphRange:

Returns the range for the characters in the receiver's text store that are mapped to the specified glyphs.

```
- (NSRange)characterRangeForGlyphRange:(NSRange)glyphRange
    actualGlyphRange:(NSRangePointer)actualGlyphRange
```

Parameters

glyphRange
The range of glyphs.

actualGlyphRange
On return, the range of all glyphs mapped to the characters in the receiver's text store. May be `NULL`.

Return Value
The range for the characters in the receiver's text store that are mapped to the glyphs in *glyphRange*.

Discussion
A subclass can override this method to interact with custom glyph storage.

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

See Also
`- glyphRangeForCharacterRange:actualCharacterRange:` (page 23)

Declared In
`NSTypesetter.h`

currentParagraphStyle

Returns the paragraph style object for the text being typeset.

```
- (NSParagraphStyle *)currentParagraphStyle
```

Return Value
The paragraph style object for the text being typeset. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 25).

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

Declared In
`NSTypesetter.h`

currentTextContainer

Returns the text container for the text being typeset.

```
- (NSTextContainer *)currentTextContainer
```

Return Value
The text container for the text being typeset. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 25).

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

See Also
`- textContainers` (page 39)

Declared In
`NSTypesetter.h`

deleteGlyphsInRange:

Deletes the specified glyphs from the glyph cache maintained by the layout manager.

```
- (void)deleteGlyphsInRange:(NSRange)glyphRange
```

Parameters
glyphRange
The range of glyphs to be deleted.

Discussion
A subclass can override this method to interact with custom glyph storage.

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

See Also
`insertGlyph:atGlyphIndex:characterIndex:` (page 24)

Declared In
NSTypesetter.h

endLineWithGlyphRange:

Sets up layout parameters at the end of a line during typesetting.

```
- (void)endLineWithGlyphRange:(NSRange)lineGlyphRange
```

Parameters

lineGlyphRange
The range of glyphs laid out in the line.

Discussion
Concrete subclass implementations of `layoutParagraphAtPoint:` (page 26) should invoke this method at the end of each line.

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

See Also
- `beginLineWithGlyphAtIndex:` (page 16)

Declared In
NSTypesetter.h

endParagraph

Sets up layout parameters at the end of a paragraph.

```
- (void)endParagraph
```

Discussion
Concrete subclasses should invoke this method at the end of their `layoutParagraphAtPoint:` (page 26) implementation.

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

See Also
- `beginParagraph` (page 16)

Declared In
NSTypesetter.h

getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels:

Extracts the information needed to lay out the provided glyphs from the provided range.

```
- (NSUInteger)getGlyphsInRange:(NSRange)glyphsRange glyphs:(NSGlyph *)glyphBuffer
    characterIndexes:(NSUInteger *)charIndexBuffer
   glyphInscriptions:(NSGlyphInscription *)inscribeBuffer elasticBits:(BOOL
   *)elasticBuffer bidiLevels:(unsigned char *)bidiLevelBuffer
```

Parameters

glyphsRange

The range of glyphs.

glyphBuffer

The glyphs to lay out.

charIndexBuffer

The original characters for the glyphs. Note that a glyph at index 1 is not necessarily mapped to the character at index 1, because a glyph may be for a ligature or accent.

inscribeBuffer

The inscription attributes for each glyph, which are used to layout characters that are combined together.

elasticBuffer

Contains a Boolean value indicating whether a glyph is elastic for each glyph. An elastic glyph can be made longer at the end of a line or when needed for justification.

bidiLevelBuffer

Contains the bidirectional level value generated by `NSGlyphGenerator`, in case a subclass chooses to use this value.

Discussion

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSTypesetter.h`

getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:

Calculates the line fragment rectangle and line fragment used rectangle for blank lines.

```
- (void)getLineFragmentRect:(NSRectPointer)lineFragmentRect
    usedRect:(NSRectPointer)lineFragmentUsedRect
    forParagraphSeparatorGlyphRange:(NSRange)paragraphSeparatorGlyphRange
    atProposedOrigin:(NSPoint)lineOrigin
```

Parameters

lineFragmentRect

On return, the calculated line fragment rectangle.

lineFragmentUsedRect

On return, the used rectangle (the portion of the line fragment rectangle that actually contains marks).

paragraphSeparatorGlyphRange

The range of glyphs under consideration. A *paragraphSeparatorGlyphRange* with length 0 indicates an extra line fragment (which occurs if the last character in the paragraph is a line separator).

lineOrigin

The origin point of the line fragment rectangle.

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

Declared In
NSTypesetter.h

getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:

Calculates line fragment rectangle, line fragment used rectangle, and remaining rectangle for a line fragment.

```
- (void)getLineFragmentRect:(NSRectPointer)lineFragmentRect
    usedRect:(NSRectPointer)lineFragmentUsedRect
    remainingRect:(NSRectPointer)remainingRect
    forStartingGlyphAtIndex:(NSUInteger)startingGlyphIndex
    proposedRect:(NSRect)proposedRect lineSpacing:(CGFloat)lineSpacing
    paragraphSpacingBefore:(CGFloat)paragraphSpacingBefore
    paragraphSpacingAfter:(CGFloat)paragraphSpacingAfter
```

Parameters

lineFragmentRect

On return, the calculated line fragment rectangle.

lineFragmentUsedRect

On return, the used rectangle (the portion of the line fragment rectangle that actually contains marks).

remainingRect

On return, the remaining rectangle of *proposedRect*.

startingGlyphIndex

The glyph index where the line fragment starts.

proposedRect

The proposed rectangle of the line fragment.

lineSpacing

The line spacing.

paragraphSpacingBefore

The spacing before the paragraph.

paragraphSpacingAfter

The spacing after the paragraph.

Discussion
The height of the line fragment is determined using *lineSpacing*, *paragraphSpacingBefore*, and *paragraphSpacingAfter* as well as *proposedRect*. The width for *lineFragmentUsedRect* is set to the *lineFragmentRect* width. In the standard implementation, paragraph spacing is included in the line fragment rectangle but not the line fragment used rectangle; line spacing is included in both.

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

Declared In
NSTypesetter.h

glyphRangeForCharacterRange:actualCharacterRange:

Returns the range for the glyphs mapped to the characters of the text store in the specified range.

```
- (NSRange)glyphRangeForCharacterRange:(NSRange)charRange
    actualCharacterRange:(NSRangePointer)actualCharRange
```

Parameters

charRange

The range of the characters whose glyph range is desired.

actualCharRange

On return, all characters mapped to those glyphs; may be `NULL`.

Return Value

The range for the glyphs mapped to the characters of the text store in *charRange*.

Discussion

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `characterRangeForGlyphRange:actualGlyphRange:` (page 18)

Declared In

`NSTypesetter.h`

hyphenationFactor

Returns the current hyphenation factor.

```
- (float)hyphenationFactor
```

Return Value

The hyphenation factor, a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `setHyphenationFactor:` (page 33)

Declared In

`NSTypesetter.h`

hyphenationFactorForGlyphAtIndex:

Returns the hyphenation factor in effect at a specified location.

```
- (float)hyphenationFactorForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of the glyph position to examine.

Return Value

The hyphenation factor in effect at *glyphIndex*. The hyphenation factor is a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

Discussion

The typesetter calls this method with a proposed hyphenation point for a line break to find the hyphenation factor in effect at that time. A subclass can override this method to customize the text layout process.

Availability

Available in Mac OS X v10.4 and later.

See Also

- hyphenCharacterForGlyphAtIndex: (page 24)

Declared In

NSTypesetter.h

hyphenCharacterForGlyphAtIndex:

Returns the hyphen character to be inserted after the specified glyph.

```
- (UTF32Char)hyphenCharacterForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of the glyph in question.

Return Value

The hyphen character to be inserted after the glyph at *glyphIndex*.

Discussion

The typesetter calls this method before hyphenating. A subclass can override this method to return a different hyphen glyph.

Availability

Available in Mac OS X v10.4 and later.

See Also

- hyphenationFactorForGlyphAtIndex: (page 23)

Declared In

NSTypesetter.h

insertGlyph:atGlyphIndex:characterIndex:

Enables the typesetter to insert a new glyph into the stream.

```
- (void)insertGlyph:(NSGlyph)glyph atGlyphIndex:(NSUInteger)glyphIndex
    characterIndex:(NSUInteger)charIndex
```

Parameters

glyph

The glyph to insert into the glyph cache.

glyphIndex

The index at which to insert *glyph*.

charIndex

The index of the character that *glyph* maps to. If the glyph is mapped to several characters, *charIndex* should indicate the first character to which it's mapped.

Discussion

The standard typesetter uses this method for inserting hyphenation glyphs. Because this method keeps the glyph caches synchronized, subclasses should always use this method to insert glyphs instead of calling `layoutManager` (page 26) directly.

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSTypesetter.h`

layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:

Lays out glyphs in the specified layout manager starting at a specified glyph.

```
- (void)layoutGlyphsInLayoutManager:(NSLayoutManager *)layoutMgr
    startingAtGlyphIndex:(NSUInteger)startGlyphIndex
    maxNumberOfLineFragments:(NSUInteger)maxNumLines nextGlyphIndex:(NSUInteger
    *)nextGlyph
```

Parameters

layoutMgr

The layout manager in which to lay out glyphs.

startGlyphIndex

The index of the starting glyph.

maxNumLines

The maximum number of lines to generate. Fewer lines may be laid out if the glyph storage runs out of glyphs.

nextGlyph

On return, set to the index of the next glyph that needs to be laid out.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTypesetter.h`

layoutManager

Returns the layout manager for the text being typeset.

```
- (NSLayoutManager *)layoutManager
```

Return Value
The layout manager for the text being typeset. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 25).

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

Declared In
`NSTypesetter.h`

layoutParagraphAtPoint:

Lays out glyphs in the current glyph range until the next paragraph separator is reached.

```
- (NSUInteger)layoutParagraphAtPoint:(NSPointPointer)lineFragmentOrigin
```

Parameters

lineFragmentOrigin

The upper-left corner of line fragment rectangle. On return, *lineFragmentOrigin* contains the next origin.

Return Value
The next glyph index; usually the index right after the paragraph separator, but it can be inside the paragraph range if, for example, the end of the text container is reached before the paragraph separator.

Discussion
Concrete subclasses must implement this method. A concrete implementation must invoke `beginParagraph` (page 16), `beginLineWithGlyphAtIndex:` (page 16), `endLineWithGlyphRange:` (page 20), and `endParagraph` (page 20).

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

Declared In
`NSTypesetter.h`

lineFragmentPadding

Returns the current line fragment padding, in points.

```
- (CGFloat)lineFragmentPadding
```

Return Value
The current line fragment padding, in points; that is, the portion on each end of the line fragment rectangle left blank.

Discussion
Text is inset within the line fragment rectangle by this amount.

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

See Also
- `setLineFragmentPadding:` (page 33)

Declared In
`NSTypesetter.h`

lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:

Returns the line spacing in effect following the specified glyph.

```
- (CGFloat)lineSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Parameters

glyphIndex
The index of the glyph in question.

rect
The proposed line fragment rectangle.

Return Value
the line spacing in effect following the glyph at *glyphIndex*.

Discussion
The `NSATSTypesetter` calls this method to determine the number of points of space to include below the descenders in the used rectangle for the proposed line fragment rectangle *rect*.

Line spacing, also called leading, is an attribute of `NSParagraphStyle`, which you can set on an `NSMutableParagraphStyle` object. A font typically includes a default minimum line spacing metric used if none is set in the paragraph style.

If the typesetter behavior specified in the layout manager is `NSTypesetterOriginalBehavior`, the text system uses the original, private typesetter `NSSimpleHorizontalTypesetter`, which adds the line spacing above the ascender. Similarly, `NSATSTypesetter` adds the line spacing above the ascender if the value is negative.

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

Declared In
`NSTypesetter.h`

paragraphCharacterRange

Returns the character range currently being processed.

```
- (NSRange)paragraphCharacterRange
```

Return Value
The character range currently being processed.

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

See Also
- `paragraphSeparatorCharacterRange` (page 28)
- `paragraphSeparatorGlyphRange` (page 29)
- `paragraphGlyphRange` (page 28)

Declared In
`NSTypesetter.h`

paragraphGlyphRange

Returns the glyph range currently being processed.

```
- (NSRange)paragraphGlyphRange
```

Return Value
The glyph range currently being processed.

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

See Also
- `setParagraphGlyphRange:separatorGlyphRange:` (page 35)
- `paragraphSeparatorGlyphRange` (page 29)
- `paragraphCharacterRange` (page 27)
- `paragraphSeparatorCharacterRange` (page 28)

Declared In
`NSTypesetter.h`

paragraphSeparatorCharacterRange

Returns the current paragraph separator character range.

```
- (NSRange)paragraphSeparatorCharacterRange
```

Return Value
The current paragraph separator character range, which is the full range that contains the current character range and that extends from one paragraph separator character to the next.

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

See Also
- `paragraphGlyphRange` (page 28)
- `paragraphSeparatorGlyphRange` (page 29)
- `paragraphCharacterRange` (page 27)

Declared In
NSTypesetter.h

paragraphSeparatorGlyphRange

Returns the current paragraph separator range.

```
- (NSRange)paragraphSeparatorGlyphRange
```

Return Value
The current paragraph separator range, which is the full range that contains the current glyph range and that extends from one paragraph separator character to the next.

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

See Also
- setParagraphGlyphRange:separatorGlyphRange: (page 35)
- paragraphGlyphRange (page 28)
- paragraphSeparatorCharacterRange (page 28)
- paragraphCharacterRange (page 27)

Declared In
NSTypesetter.h

paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:

Returns the paragraph spacing that is in effect after the specified glyph.

```
- (CGFloat)paragraphSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Parameters

glyphIndex
The index of the glyph in question.

rect
The line fragment rectangle of the last line in the paragraph.

Return Value
The paragraph spacing—that is, the number of points of space added following a paragraph—that is in effect after the glyph specified by *glyphIndex*.

Discussion
The typesetter adds the number of points specified in the return value to the bottom of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added after a paragraph correlates to the value returned by the paragraphSpacing method of NSParagraphStyle, which you can set using the setParagraphSpacing: method of NSMutableParagraphStyle.

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

See Also
- paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect: (page 30)

Declared In
NSTypesetter.h

paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:

Returns the number of points of space—added before a paragraph—that is in effect before the specified glyph.

```
- (CGFloat)paragraphSpacingBeforeGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Parameters

glyphIndex
The index of the glyph in question.

rect
The line fragment rectangle of the first line in the paragraph.

Return Value
The number of points of space—added before a paragraph—that is in effect before the glyph specified by *glyphIndex*.

Discussion
The typesetter adds the number of points specified in the return value to the top of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added before a paragraph correlates to the value returned by the paragraphSpacingBefore method of NSParagraphStyle, which you can set using the setParagraphSpacingBefore: method of NSMutableParagraphStyle.

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

See Also
- paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect: (page 29)

Declared In
NSTypesetter.h

setAttachmentSize:forGlyphRange:

Sets the size the specified glyphs (assumed to be attachments) will be asked to draw themselves at.

```
- (void)setAttachmentSize:(NSSize)attachmentSize forGlyphRange:(NSRange)glyphRange
```

Parameters

attachmentSize
The size the glyphs in *glyphRange* (assumed to be attachments) will be asked to draw themselves at.

glyphRange
The range of glyphs the attachment size applies to.

Discussion
A subclass can override this method to interact with custom glyph storage.

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

Declared In
`NSTypesetter.h`

setAttributedString:

Sets the text backing store on which this typesetter operates.

```
- (void)setAttributedString:(NSAttributedString *)attrString
```

Parameters

attrString
The text backing store on which the typesetter should operate.

Special Considerations

Typesetters do not retain the text backing store on which they are operating.

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

See Also
`- attributedString` (page 15)

Declared In
`NSTypesetter.h`

setBidiLevels:forGlyphRange:

Sets the direction of the specified glyphs for bidirectional text.

```
- (void)setBidiLevels:(const uint8_t *)levels forGlyphRange:(NSRange)glyphRange
```

Parameters

levels
Values in *levels* can range from 0 to 61 as defined by Unicode Standard Annex #9.

glyphRange
The range of glyphs for which the bidirectional text levels are desired.

Discussion
A subclass can override this method to interact with custom glyph storage.

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

Declared In
`NSTypesetter.h`

setBidiProcessingEnabled:

Controls whether the typesetter performs bidirectional text processing.

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

Parameters

flag

YES to enable bidirectional text processing, NO to disable it.

Discussion

You can use this method to disable the bidirectional layout stage if you know the paragraph does not need this stage; that is, if the characters in the backing store are in display order.

Availability

Available in Mac OS X v10.4 and later.

See Also

- bidiProcessingEnabled (page 17)

Declared In

NSTypesetter.h

setDrawsOutsideLineFragment:forGlyphRange:

Sets whether the specified glyphs exceed the bounds of the line fragment in which they are laid out.

```
- (void)setDrawsOutsideLineFragment:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Parameters

flag

YES if the glyphs in *glyphRange* exceed the bounds of the line fragment in which they are laid out, NO otherwise.

glyphRange

The range of the glyphs in question.

Discussion

This can happen when text is set at a fixed line height. For example, if the user specifies a fixed line height of 12 points and sets the font size to 24 points, the glyphs will exceed their layout rectangles.

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTypesetter.h

setHardInvalidation:forGlyphRange:

Sets whether to force the layout manager to invalidate the specified portion of the glyph cache when invalidating layout.

```
- (void)setHardInvalidation:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Parameters

flag

YES if the layout manager should invalidate the specified portion of the glyph cache, NO otherwise.

glyphRange

The range of glyphs in the cache to be marked for hard invalidation.

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

Declared In
NSTypesetter.h

setHyphenationFactor:

Sets the threshold controlling when hyphenation is attempted.

```
- (void)setHyphenationFactor:(float)factor
```

Parameters

factor

A frequency factor in the range of 0.0 to 1.0. By default, the value is 0.0, meaning hyphenation is off. A *factor* of 1.0 causes hyphenation to be attempted always.

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

See Also
- hyphenationFactor (page 23)

Declared In
NSTypesetter.h

setLineFragmentPadding:

Sets the amount (in points) by which text is inset within line fragment rectangles.

```
- (void)setLineFragmentPadding:(CGFloat)padding
```

Parameters

padding

The amount (in points) by which text is inset within line fragment rectangles.

Special Considerations

Line fragment padding isn't a suitable means for expressing margins; you should set the text view's position and size for document margins or the paragraph margin attributes for text margins.

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

See Also
- lineFragmentPadding (page 26)

Declared In
NSTypesetter.h

setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:

Sets the line fragment rectangle where the specified glyphs are laid out.

```
- (void)setLineFragmentRect:(NSRect)fragmentRect forGlyphRange:(NSRange)glyphRange
    usedRect:(NSRect)usedRect baselineOffset:(CGFloat)baselineOffset
```

Parameters

fragmentRect
The line fragment rectangle where the glyphs in *glyphRange* are laid out.

glyphRange
The range of the specified glyphs.

usedRect
The portion of *fragmentRect*, in the NSTextContainer object's coordinate system, that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding). The *usedRect* must be equal to or contained within *fragmentRect*.

baselineOffset
The vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

Discussion
The exact positions of the glyphs must be set after the line fragment rectangle with `setLocation:forStartOfGlyphRange:`.

A subclass can override this method to interact with custom glyph storage.

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

Declared In
NSTypesetter.h

setLocation:withAdvancements:forStartOfGlyphRange:

Sets the location where the specified glyphs are laid out.

```
- (void)setLocation:(NSPoint)location withAdvancements:(const CGFloat *)advancements
    forStartOfGlyphRange:(NSRange)glyphRange
```

Parameters

location
The location where the glyphs in *glyphRange* are laid out. The x-coordinate of *location* is expressed relative to the line fragment rectangle origin, and the y-coordinate is expressed relative to the baseline previously specified by `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 34).

advancements
The nominal glyph advance width specified in the font metric information.

glyphRange

The range of glyphs whose layout location is being set. This series of glyphs can be displayed with a single PostScript `show` operation (a nominal range).

Discussion
Setting the location for a series of glyphs implies that the glyphs preceding it can't be included in a single `show` operation.

Before setting the location for a glyph range, you must specify line fragment rectangle with `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:`.

A subclass can override this method to interact with custom glyph storage.

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

Declared In
`NSTypesetter.h`

setNotShownAttribute:forGlyphRange:

Sets whether the specified glyphs are not shown.

```
- (void)setNotShownAttribute:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Parameters

flag

`YES` if the glyphs in *glyphRange* are not shown, `NO` if they are shown.

glyphRange

The range of glyphs in question.

Discussion
For example, a tab or newline character doesn't leave any marks; it just indicates where following glyphs are laid out.

A subclass can override this method to interact with custom glyph storage.

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

Declared In
`NSTypesetter.h`

setParagraphGlyphRange:separatorGlyphRange:

Sets the current glyph range being processed.

```
- (void)setParagraphGlyphRange:(NSRange)paragraphRange
    separatorGlyphRange:(NSRange)paragraphSeparatorRange
```

Parameters

paragraphRange

The current glyph range being processed.

paragraphSeparatorRange
The range of the paragraph separator character or characters.

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

See Also
- `paragraphGlyphRange` (page 28)
- `paragraphSeparatorGlyphRange` (page 29)

Declared In
`NSTypesetter.h`

setTypesetterBehavior:

Sets the default typesetter behavior, which affects glyph spacing and line height.

```
- (void)setTypesetterBehavior:(NSTypesetterBehavior)behavior
```

Parameters
behavior
The new behavior.

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

See Also
- `typesetterBehavior` (page 39)

Declared In
`NSTypesetter.h`

setUsesFontLeading:

Sets whether the typesetter uses the leading (or line gap) value specified in the font metric information.

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

Parameters
flag
`YES` to use the information in the font metrics, `NO` to ignore it.

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

See Also
- `usesFontLeading` (page 40)

Declared In
`NSTypesetter.h`

shouldBreakLineByHyphenatingBeforeCharacterAtIndex:

Returns whether the line being laid out should be broken by hyphenating at the specified character.

```
- (BOOL)shouldBreakLineByHyphenatingBeforeCharacterAtIndex:(NSUInteger)charIndex
```

Parameters

charIndex

The index of the character just after the proposed hyphenation would occur.

Return Value

`YES` if the line should be broken by hyphenating, `NO` otherwise.

Discussion

The typesetter calls this method, if implemented by a subclass, before breaking a line by hyphenating before the character at *charIndex*, enabling the subclass to control line breaking.

A subclass can override this method to customize the text layout process. If the method returns `NO`, the typesetter continues looking for a break point.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- shouldBreakLineByWordBeforeCharacterAtIndex:` (page 37)

Declared In

`NSTypesetter.h`

shouldBreakLineByWordBeforeCharacterAtIndex:

Returns whether the line being laid out should be broken by a word break at the specified character.

```
- (BOOL)shouldBreakLineByWordBeforeCharacterAtIndex:(NSUInteger)charIndex
```

Parameters

charIndex

The index of the character just after the proposed word break would occur.

Return Value

`YES` if the line should be broken by a word break, `NO` otherwise.

Discussion

The typesetter calls this method, if implemented by a subclass, before breaking a line by word wrapping before the character at *charIndex*, enabling the subclass to control line breaking.

A subclass can override this method to customize the text layout process. If the method returns `NO`, the typesetter continues looking for a break point.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- shouldBreakLineByHyphenatingBeforeCharacterAtIndex:` (page 37)

Declared In
NSTypesetter.h

substituteFontForFont:

Returns a screen font suitable for use in place of a given font.

```
- (NSFont *)substituteFontForFont:(NSFont *)originalFont
```

Parameters

originalFont
The original font.

Return Value
A screen font suitable for use in place of *originalFont*. This method returns *originalFont* if a screen font can't be used or isn't available.

Discussion
A screen font can only be substituted if the receiver is set to use screen fonts and if no text view associated with the receiver is scaled or rotated.

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

Declared In
NSTypesetter.h

substituteGlyphsInRange:withGlyphs:

Replaces the specified glyphs with specified replacement glyphs.

```
- (void)substituteGlyphsInRange:(NSRange)glyphRange withGlyphs:(NSGlyph *)glyphs
```

Parameters

glyphRange
The range of glyphs to be substituted.

glyphs
The glyphs to substitute for the glyphs in `glyphRange`.

Discussion
This method does not alter the glyph-to-character mapping or invalidate layout information.

A subclass can override this method to interact with custom glyph storage.

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

Declared In
NSTypesetter.h

textContainers

Returns an array containing the text containers belonging to the current layout manager.

```
- (NSArray *)textContainers
```

Return Value
An array containing the text containers belonging to the current layout manager. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 25).

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

See Also
- `layoutManager` (page 26)
- `currentTextContainer` (page 19)

Declared In
`NSTypesetter.h`

textTabForGlyphLocation:writingDirection:maxLocation:

Returns the text tab next closest to a given glyph location within the given parameters.

```
- (NSTextTab *)textTabForGlyphLocation:(CGFloat)glyphLocation
    writingDirection:(NSWritingDirection)direction maxLocation:(CGFloat)maxLocation
```

Parameters

glyphLocation
The location at which to start searching.

direction
The direction in which to search.

maxLocation
The maximum location for the search.

Return Value
The text tab next closest to *glyphLocation*, indexing in *direction* but not beyond *maxLocation*.

Discussion
The typesetter calls this method whenever it finds a tab character. To determine the width to advance the next glyph, the typesetter examines the `NSParagraphStyle` object's tab array and the default tab interval.

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

Declared In
`NSTypesetter.h`

typesetterBehavior

Returns the current typesetter behavior.

```
- (NSTypesetterBehavior)typesetterBehavior
```

Return Value
The current typesetter behavior.

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

See Also
- `setTypesetterBehavior:` (page 36)

Declared In
`NSTypesetter.h`

usesFontLeading

Returns whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.

```
- (BOOL)usesFontLeading
```

Return Value
`YES` if it uses the information in the font metrics, `NO` otherwise.

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

See Also
- `setUsesFontLeading:` (page 36)

Declared In
`NSTypesetter.h`

willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:

Called by the typesetter just prior to storing the actual line fragment rectangle location in the layout manager.

```
- (void)willSetLineFragmentRect:(NSRectPointer)lineRect
    forGlyphRange:(NSRange)glyphRange usedRect:(NSRectPointer)usedRect
    baselineOffset:(CGFloat *)baselineOffset
```

Parameters

lineRect
The rectangle in which the glyphs in *glyphRange* are laid out.

glyphRange
The range of the glyphs to lay out.

usedRect
The portion of *lineRect*, in the NSTextContainer object's coordinate system, that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding). The *usedRect* must be equal to or contained within *lineRect*.

baselineOffset
The vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

Discussion
Called by the typesetter just prior to calling `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 34) which stores the actual line fragment rectangle location in the layout manager.

A subclass can override this method to customize the text layout process. For example, it could change the shape of the line fragment rectangle. The subclass is responsible for ensuring that the modified rectangle remains valid (for example, that it lies within the text container).

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

Declared In
`NSTypesetter.h`

Constants

NSTypesetterControlCharacterAction

The following constants are possible values returned by the `actionForControlCharacterAtIndex:` (page 14) method to determine the action associated with a control character.

```
enum {
    NSTypesetterZeroAdvancementAction = (1 << 0),
    NSTypesetterWhitespaceAction = (1 << 1),
    NSTypesetterHorizontalTabAction = (1 << 2),
    NSTypesetterLineBreakAction = (1 << 3),
    NSTypesetterParagraphBreakAction = (1 << 4),
    NSTypesetterContainerBreakAction = (1 << 5)
};
typedef NSUInteger NSTypesetterControlCharacterAction;
```

Constants

`NSTypesetterZeroAdvancementAction`
Glyphs with this action are flitered out from layout `(notShownAttribute == YES)`.
Available in Mac OS X v10.4 and later.
Declared in `NSTypesetter.h`.

`NSTypesetterWhitespaceAction`
The width for glyphs with this action are determined by `boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 17), if the method is implemented; otherwise, same as `NSTypesetterZeroAdvancementAction`.
Available in Mac OS X v10.4 and later.
Declared in `NSTypesetter.h`.

`NSTypesetterHorizontalTabAction`
Treated as tab character.
Available in Mac OS X v10.4 and later.
Declared in `NSTypesetter.h`.

`NSTypesetterLineBreakAction`
: Causes line break.

  Available in Mac OS X v10.4 and later.

  Declared in `NSTypesetter.h`.

`NSTypesetterParagraphBreakAction`
: Causes paragraph break; the value returned by `firstLineHeadIndent` is the advancement used for the following glyph.

  Available in Mac OS X v10.4 and later.

  Declared in `NSTypesetter.h`.

`NSTypesetterContainerBreakAction`
: Causes container break.

  Available in Mac OS X v10.4 and later.

  Declared in `NSTypesetter.h`.

Declared In
`NSTypesetter.h`

Document Revision History

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

Date	Notes
2007-03-26	Removed description of deprecated lineFragmentRectForProposedRect:remainingRect: method which no longer appears in the public header file. Corrected description of NSTypesetterControlCharacterAction enumeration constant.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index

N

P

S

T

U

W