

---

# NSRulerView Class Reference

[Cocoa](#) > [Text & Fonts](#)



2006-05-23



Apple Inc.  
© 2006 Apple Computer, Inc.  
All rights reserved.

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

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

Numbers is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

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

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR**

**CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

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

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

# Contents

## NSRulerView Class Reference 5

---

|  |    |
|--|----|
| Class at a Glance  | 5  |
| Overview   | 6  |
| Tasks  | 6  |
| Creating Instances   | 6  |
| Altering Measurement Units   | 6  |
| Setting the Client View  | 6  |
| Setting an Accessory View  | 7  |
| Setting the Zero Mark Position   | 7  |
| Adding and Removing Markers  | 7  |
| Drawing Temporary Ruler Lines  | 7  |
| Drawing  | 7  |
| Ruler Layout   | 8  |
| Adding markers   | 8  |
| Moving markers   | 9  |
| Removing markers   | 9  |
| Handling mouse events  | 9  |
| Changing client view   | 9  |
| Class Methods  | 9  |
| registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle: stepDownCycle: | 9  |
| Instance Methods   | 10 |
| accessoryView  | 10 |
| addMarker:   | 10 |
| baselineLocation   | 11 |
| clientView   | 11 |
| drawHashMarksAndLabelsInRect:  | 11 |
| drawMarkersInRect:   | 12 |
| initWithScrollView:orientation:  | 12 |
| invalidateHashMarks  | 13 |
| isFlipped  | 13 |
| markers  | 13 |
| measurementUnits   | 14 |
| moveRulerlineFromLocation:toLocation:  | 14 |
| orientation  | 15 |
| originOffset   | 15 |
| removeMarker:  | 15 |
| requiredThickness  | 16 |
| reservedThicknessForAccessoryView  | 16 |
| reservedThicknessForMarkers  | 16 |
| ruleThickness  | 17 |

- scrollView 17
- setAccessoryView: 17
- setClientView: 18
- setMarkers: 18
- setMeasurementUnits: 19
- setOrientation: 19
- setOriginOffset: 19
- setReservedThicknessForAccessoryView: 20
- setReservedThicknessForMarkers: 20
- setRuleThickness: 21
- setScrollView: 21
- trackMarker:withMouseEvent: 22
- Delegate Methods 22
  - rulerView:didAddMarker: 22
  - rulerView:didMoveMarker: 23
  - rulerView:didRemoveMarker: 23
  - rulerView:handleMouseDown: 23
  - rulerView:shouldAddMarker: 24
  - rulerView:shouldMoveMarker: 24
  - rulerView:shouldRemoveMarker: 24
  - rulerView:willAddMarker:atLocation: 25
  - rulerView:willMoveMarker:toLocation: 25
  - rulerView:willSetClientView: 26
- Constants 26
  - NSRulerOrientation 26

**Document Revision History 29**

---

**Index 31**

---

# NSRulerView Class Reference

---

|                            |   |
|----------------------------|---|
| <b>Inherits from</b>       | NSView : NSResponder : NSObject   |
| <b>Conforms to</b>         | NSAnimatablePropertyContainer (NSView)<br>NSCoding (NSResponder)<br>NSObject (NSObject) |
| <b>Framework</b>           | /System/Library/Frameworks/AppKit.framework   |
| <b>Availability</b>        | Available in Mac OS X v10.0 and later.  |
| <b>Companion guide</b>     | Rulers and Paragraph Styles   |
| <b>Declared in</b>         | NSRulerView.h   |
| <b>Related sample code</b> | Sketch-112<br>WhackedTV   |

## Class at a Glance

An `NSRulerView` displays a ruler and markers above or to the side of an `NSScrollView`'s document view. Views within the `NSScrollView` can become clients of the ruler view, having it display markers for their elements, and receiving messages from the ruler view when the user manipulates the markers.

## Principal Attributes

---

- Displays markers that represent elements of the client view.
- Displays in arbitrary units.
- Provides for an accessory view containing extra controls.

```
setHasHorizontalRuler: (NSScrollView)  
setHasVerticalRuler: (NSScrollView)  
initWithScrollView:orientation: (page 12) Designated initializer.
```

## Commonly Used Methods

---

[setClientView:](#) (page 18)  
Changes the ruler's client view.

- [setMarkers:](#) (page 18)  
Sets the markers displayed by the ruler view.
- [setAccessoryView:](#) (page 17)  
Sets the accessory view.
- [trackMarker:withMouseEvent:](#) (page 22)  
Allows the user to add a new marker.

## Overview

An NSRulerView resides in an NSScrollView, displaying a labeled ruler and markers for its client, the document view of the NSScrollView, or a subview of the document view.

## Tasks

### Creating Instances

- [initWithScrollView:orientation:](#) (page 12)  
Initializes a newly allocated NSRulerView to have *orientation* (NSHorizontalRuler or NSVerticalRuler) within *aScrollView*.

### Altering Measurement Units

- + [registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:](#) (page 9)  
Registers a new unit of measurement with the NSRulerView class, making it available to all instances of NSRulerView.
- [setMeasurementUnits:](#) (page 19)  
Sets the measurement units used by the ruler to *unitName*.
- [measurementUnits](#) (page 14)  
Returns the full name of the measurement units in effect for the receiver.

### Setting the Client View

- [setClientView:](#) (page 18)  
Sets the receiver's client view to *aView*, without retaining it, and removes its ruler markers, after informing the prior client of the change using [rulerView:willSetClientView:](#) (page 26).
- [clientView](#) (page 11)  
Returns the receiver's client view, if it has one.

## Setting an Accessory View

- [setAccessoryView:](#) (page 17)  
Sets the receiver's accessory view to *aView*.
- [accessoryView](#) (page 10)  
Returns the receiver's accessory view, if it has one.

## Setting the Zero Mark Position

- [setOriginOffset:](#) (page 19)  
Sets the distance to the zero hash mark from the bounds origin of the NSScrollView's document view (not of the receiver's client view), in the document view's coordinate system.
- [originOffset](#) (page 15)  
Returns the distance from the receiver's zero hash mark to the bounds origin of the NSScrollView's document view (not the receiver's client view), in the document view's coordinate system.

## Adding and Removing Markers

- [setMarkers:](#) (page 18)  
Sets the receiver's ruler markers to *markers*, removing any existing ruler markers and not consulting with the client view about the new markers.
- [markers](#) (page 13)  
Returns the receiver's NSRulerMarkers.
- [addMarker:](#) (page 10)  
Adds *aMarker* to the receiver, without consulting the client view for approval.
- [removeMarker:](#) (page 15)  
Removes *aMarker* from the receiver, without consulting the client view for approval.
- [trackMarker:withMouseEvent:](#) (page 22)  
Tracks the mouse to add *aMarker* based on the initial mouse-down or mouse-dragged event *theEvent*.

## Drawing Temporary Ruler Lines

- [moveRulerlineFromLocation:toLocation:](#) (page 14)  
Draws temporary lines in the ruler area.

## Drawing

- [drawHashMarksAndLabelsInRect:](#) (page 11)  
Draws the receiver's hash marks and labels in *aRect*, which is expressed in the receiver's coordinate system.
- [drawMarkersInRect:](#) (page 12)  
Draws the receiver's markers in *aRect*, which is expressed in the receiver's coordinate system.

- [invalidateHashMarks](#) (page 13)  
Forces recalculation of the hash mark spacing for the next time the receiver is displayed.

## Ruler Layout

- [scrollView](#): (page 21)  
Sets the NSScrollView that owns the receiver to *scrollView*, without retaining it.
- [scrollView](#) (page 17)  
Returns the NSScrollView object that contains the receiver.
- [setOrientation:](#) (page 19)  
Sets the orientation of the receiver to *orientation*.
- [orientation](#) (page 15)  
Returns the orientation of the receiver.
- [setReservedThicknessForAccessoryView:](#) (page 20)  
Sets the room available for the receiver's accessory view to *thickness*.
- [reservedThicknessForAccessoryView](#) (page 16)  
Returns the thickness reserved to contain the receiver's accessory view, its height or width depending on the receiver's orientation.
- [setReservedThicknessForMarkers:](#) (page 20)  
Sets the room available for ruler markers to *thickness*.
- [reservedThicknessForMarkers](#) (page 16)  
Returns the thickness reserved to contain the images of the receiver's ruler markers, the height or width depending on the receiver's orientation.
- [setRuleThickness:](#) (page 21)  
Sets to *thickness* the thickness of the area where ruler hash marks and labels are drawn.
- [ruleThickness](#) (page 17)  
Returns the thickness of the receiver's ruler area (the area where hash marks and labels are drawn), its height or width depending on the receiver's orientation.
- [requiredThickness](#) (page 16)  
Returns the thickness needed for proper tiling of the receiver within an NSScrollView.
- [baselineLocation](#) (page 11)  
Returns the location of the receiver's baseline, in its own coordinate system.
- [isFlipped](#) (page 13)  
Returns YES if the NSRulerView's coordinate system is flipped, NO otherwise.

## Adding markers

- [rulerView:shouldAddMarker:](#) (page 24) *delegate method*  
Requests permission for *aRulerView* to add *aMarker*, an NSRulerMarker being dragged onto the ruler by the user.
- [rulerView:willAddMarker:atLocation:](#) (page 25) *delegate method*  
Informs the client that *aRulerView* will add the new NSRulerMarker, *aMarker*.
- [rulerView:didAddMarker:](#) (page 22) *delegate method*  
Informs the client that *aRulerView* allowed the user to add *aMarker*.



## Moving markers

- `rulerView:shouldMoveMarker:` (page 24) *delegate method*  
Requests permission for *aRulerView* to move *aMarker*.
- `rulerView:willMoveMarker:toLocation:` (page 25) *delegate method*  
Informs the client that *aRulerView* will move *aMarker*, an `NSRulerMarker` already on the ruler view.
- `rulerView:didMoveMarker:` (page 23) *delegate method*  
Informs the client that *aRulerView* allowed the user to move *aMarker*.

## Removing markers

- `rulerView:shouldRemoveMarker:` (page 24) *delegate method*  
Requests permission for *aRulerView* to remove *aMarker*.
- `rulerView:didRemoveMarker:` (page 23) *delegate method*  
Informs the client that *aRulerView* allowed the user to remove *aMarker*.

## Handling mouse events

- `rulerView:handleMouseDown:` (page 23) *delegate method*  
Informs the client that the user has pressed the mouse button while the cursor is in the ruler area of *aRulerView*.

## Changing client view

- `rulerView:willSetClientView:` (page 26) *delegate method*  
Informs the client view that *aRulerView* is about to be appropriated by *newClient*.

## Class Methods

### **registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:**

Registers a new unit of measurement with the `NSRulerView` class, making it available to all instances of `NSRulerView`.

```
+ (void)registerUnitWithName:(NSString *)unitName abbreviation:(NSString *)abbreviation unitToPointsConversionFactor:(CGFloat)conversionFactor stepUpCycle:(NSArray *)stepUpCycle stepDownCycle:(NSArray *)stepDownCycle
```

#### **Discussion**

*unitName* is the name of the unit in English, in plural form and capitalized by convention—“Inches”; for example. The unit name is used as a key to identify the measurement units and so shouldn’t be localized. *abbreviation* is a localized short form of the unit name, such as “in” for Inches. *conversionFactor* is the number of PostScript points in the specified unit; there are 72.0 points per inch, for example. *stepUpCycle*

and *stepDownCycle* are arrays of NSNumbers that specify how hash marks are calculated, as explained in “Setting Up a Ruler View”. All numbers in *stepUpCycle* should be greater than 1.0, those in *stepDownCycle* should be less than 1.0.

NSRulerView supports these units by default:

| Unit Name   | Abbreviation | Points/Unit | Step-Up Cycle | Step-Down Cycle |
|-------------|--------------|-------------|---------------|-----------------|
| Inches      | in           | 72.0        | 2.0           | 0.5             |
| Centimeters | cm           | 28.35       | 2.0           | 0.5, 0.2        |
| Points      | pt           | 1.0         | 10.0          | 0.5             |
| Picas       | pc           | 12.0        | 10.0          | 0.5             |

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [setMeasurementUnits:](#) (page 19)

#### Declared In

NSRulerView.h

## Instance Methods

### accessoryView

Returns the receiver’s accessory view, if it has one.

- (NSView \*)accessoryView

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [setAccessoryView:](#) (page 17)  
 - [reservedThicknessForAccessoryView](#) (page 16)

#### Declared In

NSRulerView.h

### addMarker:

Adds *aMarker* to the receiver, without consulting the client view for approval.

- (void)addMarker:(NSRulerMarker \*)aMarker

### Discussion

Raises an `NSInternalInconsistencyException` if the receiver has no client view.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setMarkers:](#) (page 18)
- [removeMarker:](#) (page 15)
- [markers](#) (page 13)
- [trackMarker:withMouseEvent:](#) (page 22)

### Declared In

NSRulerView.h

## baselineLocation

Returns the location of the receiver's baseline, in its own coordinate system.

- (CGFloat)baselineLocation

### Discussion

This is a y position for horizontal rulers and an x position for vertical ones.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [ruleThickness](#) (page 17)

### Declared In

NSRulerView.h

## clientView

Returns the receiver's client view, if it has one.

- (NSView \*)clientView

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setClientView:](#) (page 18)

### Declared In

NSRulerView.h

## drawHashMarksAndLabelsInRect:

Draws the receiver's hash marks and labels in *aRect*, which is expressed in the receiver's coordinate system.

- (void)drawHashMarksAndLabelsInRect:(NSRect)aRect

#### Discussion

This method is invoked by `drawRect:`—you should never need to invoke it directly. You can define custom measurement units using the class method

`registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:` (page 9). Override this method if you want to customize the appearance of the hash marks themselves.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [invalidateHashMarks](#) (page 13)
- [drawMarkersInRect:](#) (page 12)

#### Declared In

NSRulerView.h

## drawMarkersInRect:

Draws the receiver's markers in *aRect*, which is expressed in the receiver's coordinate system.

- (void)drawMarkersInRect:(NSRect)aRect

#### Discussion

This method is invoked by `drawRect:`; you should never need to invoke it directly, but you might want to override it if you want to do something different when drawing markers.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [reservedThicknessForMarkers](#) (page 16)
- [drawHashMarksAndLabelsInRect:](#) (page 11)

#### Declared In

NSRulerView.h

## initWithScrollView:orientation:

Initializes a newly allocated NSRulerView to have *orientation* (NSHorizontalRuler or NSVerticalRuler) within *aScrollView*.

```
- (id)initWithScrollView:(NSScrollView *)aScrollView
    orientation:(NSRulerOrientation)orientation
```

#### Discussion

The new ruler view displays the user's preferred measurement units and has no client, markers, or accessory view. Unlike most subclasses of NSView, no initial frame rectangle is given for NSRulerView; its containing NSScrollView adjusts its frame rectangle as needed.

This method is the designated initializer for the NSRulerView class. Returns an initialized object.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSRulerView.h

**invalidateHashMarks**

Forces recalculation of the hash mark spacing for the next time the receiver is displayed.

- (void)invalidateHashMarks

**Discussion**

You should never need to invoke this method directly, but might need to override it if you override [drawHashMarksAndLabelsInRect:](#) (page 11).

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [drawHashMarksAndLabelsInRect:](#) (page 11)

**Declared In**

NSRulerView.h

**isFlipped**

Returns YES if the NSRulerView's coordinate system is flipped, NO otherwise.

- (BOOL)isFlipped

**Discussion**

A vertical ruler takes into account whether the coordinate system of the NSScrollView's document view—not the receiver's client view—is flipped. A horizontal ruler is always flipped.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

WhackedTV

**Declared In**

NSRulerView.h

**markers**

Returns the receiver's NSRulerMarkers.

- (NSArray \*)markers

**Discussion**

The markers aren't guaranteed to be sorted in any particular order.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setMarkers:](#) (page 18)
- [addMarker:](#) (page 10)
- [removeMarker:](#) (page 15)
- `markerLocation` (NSRulerMarker)

**Declared In**

NSRulerView.h

**measurementUnits**

Returns the full name of the measurement units in effect for the receiver.

- (NSString \*)measurementUnits

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setMeasurementUnits:](#) (page 19)
- + [registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:](#) (page 9)

**Declared In**

NSRulerView.h

**moveRulerlineFromLocation:toLocation:**

Draws temporary lines in the ruler area.

- (void)moveRulerlineFromLocation:(CGFloat)*oldLoc* toLocation:(CGFloat)*newLoc*

**Discussion**

If *oldLoc* is 0 or greater, erases the ruler line at that location; if *newLoc* is 0 or greater, draws a new rulerline at that location. *oldLoc* and *newLoc* are expressed in the coordinate system of the NSRulerView, not the client or document view, and are x coordinates for horizontal rulers and y coordinates for vertical rulers. Use NSView's `convert...` methods to convert coordinates from the client or document view's coordinate system to that of the NSRulerView.

This method is useful for drawing highlight lines in the ruler to show the position or extent of an object while it's being dragged in the client view. The sender is responsible for keeping track of the number and positions of temporary lines—the NSRulerView only does the drawing.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

Sketch-112

**Declared In**

NSRulerView.h

**orientation**

Returns the orientation of the receiver.

- (NSRulerOrientation)orientation

**Discussion**Possible values are described in [“Constants”](#) (page 26).**Availability**

Available in Mac OS X v10.0 and later.

**See Also**- [setOrientation:](#) (page 19)**Declared In**

NSRulerView.h

**originOffset**

Returns the distance from the receiver’s zero hash mark to the bounds origin of the NSScrollView’s document view (not the receiver’s client view), in the document view’s coordinate system.

- (CGFloat)originOffset

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**- [setOriginOffset:](#) (page 19)**Declared In**

NSRulerView.h

**removeMarker:**Removes *aMarker* from the receiver, without consulting the client view for approval.

- (void)removeMarker:(NSRulerMarker \*)aMarker

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**- [setMarkers:](#) (page 18)- [addMarker:](#) (page 10)**Declared In**

NSRulerView.h

## requiredThickness

Returns the thickness needed for proper tiling of the receiver within an NSScrollView.

- (CGFloat)requiredThickness

### Discussion

This thickness is the height of a horizontal ruler and the width of a vertical ruler. The required thickness is the sum of the thicknesses of the ruler area, the marker area, and the accessory view.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [ruleThickness](#) (page 17)
- [reservedThicknessForMarkers](#) (page 16)
- [reservedThicknessForAccessoryView](#) (page 16)

### Declared In

NSRulerView.h

## reservedThicknessForAccessoryView

Returns the thickness reserved to contain the receiver's accessory view, its height or width depending on the receiver's orientation.

- (CGFloat)reservedThicknessForAccessoryView

### Discussion

This thickness is automatically enlarged as necessary to the accessory view's thickness (but never automatically reduced). To prevent retiling of a ruler view's scroll view, you should set its maximal thickness upon creating using [setReservedThicknessForAccessoryView:](#) (page 20).

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

NSRulerView.h

## reservedThicknessForMarkers

Returns the thickness reserved to contain the images of the receiver's ruler markers, the height or width depending on the receiver's orientation.

- (CGFloat)reservedThicknessForMarkers

### Discussion

This thickness is automatically enlarged as necessary to accommodate the thickest ruler marker image (but never automatically reduced). To prevent retiling of a ruler view's scroll view, you should set its maximal thickness upon creating using [setReservedThicknessForMarkers:](#) (page 20).

### Availability

Available in Mac OS X v10.0 and later.



**See Also**

- `thicknessRequiredInRuler` (NSRulerMarker)

**Declared In**

NSRulerView.h

**ruleThickness**

Returns the thickness of the receiver's ruler area (the area where hash marks and labels are drawn), its height or width depending on the receiver's orientation.

- (CGFloat)ruleThickness

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setRuleThickness:](#) (page 21)

**Declared In**

NSRulerView.h

**scrollView**

Returns the NSScrollView object that contains the receiver.

- (NSScrollView \*)scrollView

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [scrollView:](#) (page 21)  
 - `setHorizontalRulerView:` (NSScrollView)  
 - `setVerticalRulerView:` (NSScrollView)

**Declared In**

NSRulerView.h

**setAccessoryView:**

Sets the receiver's accessory view to *aView*.

- (void)setAccessoryView:(NSView \*)aView

**Discussion**

Raises an NSInternalInconsistencyException if *aView* is not nil and the receiver has no client view.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [accessoryView](#) (page 10)
- [reservedThicknessForAccessoryView](#) (page 16)

**Declared In**

NSRulerView.h

**setClientView:**

Sets the receiver's client view to *aView*, without retaining it, and removes its ruler markers, after informing the prior client of the change using [rulerView:willSetClientView:](#) (page 26).

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

**Discussion**

*aView* must be either the document view of the NSScrollView that contains the receiver or a subview of the document view.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [clientView](#) (page 11)

**Declared In**

NSRulerView.h

**setMarkers:**

Sets the receiver's ruler markers to *markers*, removing any existing ruler markers and not consulting with the client view about the new markers.

```
- (void)setMarkers:(NSArray *)markers
```

**Discussion**

*markers* can be nil or empty to remove all ruler markers. Raises an `NSInternalInconsistencyException` if *markers* is not nil and the receiver has no client view.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [addMarker:](#) (page 10)
- [removeMarker:](#) (page 15)

**Related Sample Code**

Sketch-112

**Declared In**

NSRulerView.h

## setMeasurementUnits:

Sets the measurement units used by the ruler to *unitName*.

```
- (void)setMeasurementUnits:(NSString *)unitName
```

### Discussion

*unitName* must have been registered with the NSRulerView class object prior to invoking this method. See the description of the class method

[registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:](#) (page 9) for a list of predefined units.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [measurementUnits](#) (page 14)

### Declared In

NSRulerView.h

## setOrientation:

Sets the orientation of the receiver to *orientation*.

```
- (void)setOrientation:(NSRulerOrientation)orientation
```

### Discussion

Possible values for *orientation* are described in “[Constants](#)” (page 26). You should never need to invoke this method directly—it’s automatically invoked by the containing NSScrollView.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [orientation](#) (page 15)

### Declared In

NSRulerView.h

## setOriginOffset:

Sets the distance to the zero hash mark from the bounds origin of the NSScrollView’s document view (not of the receiver’s client view), in the document view’s coordinate system.

```
- (void)setOriginOffset:(CGFloat)offset
```

### Discussion

The default offset is 0.0, meaning that the ruler origin coincides with the bounds origin of the document view.

### Availability

Available in Mac OS X v10.0 and later.

**See Also**

- [originOffset](#) (page 15)

**Declared In**

NSRulerView.h

**setReservedThicknessForAccessoryView:**

Sets the room available for the receiver's accessory view to *thickness*.

```
- (void)setReservedThicknessForAccessoryView:(CGFloat)thickness
```

**Discussion**

If the ruler is horizontal, *thickness* is the height of the accessory view; otherwise, it's the width. NSRulerViews by default reserve no space for an accessory view.

An NSRulerView automatically increases the reserved thickness as necessary to that of the accessory view. When the accessory view is thinner than the reserved space, it's centered in that space. If you plan to use several accessory views of different sizes, you should set the reserved thickness beforehand to that of the thickest accessory view, in order to avoid retiling of the NSScrollView.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [reservedThicknessForAccessoryView](#) (page 16)
- [setAccessoryView:](#) (page 17)
- [setReservedThicknessForMarkers:](#) (page 20)

**Related Sample Code**

Sketch-112

**Declared In**

NSRulerView.h

**setReservedThicknessForMarkers:**

Sets the room available for ruler markers to *thickness*.

```
- (void)setReservedThicknessForMarkers:(CGFloat)thickness
```

**Discussion**

The default thickness reserved for markers is 15.0 PostScript units for a horizontal ruler and 0.0 PostScript units for a vertical ruler (under the assumption that vertical rulers rarely contain markers). If you don't expect to have any markers on the ruler, you can set the reserved thickness to 0.0.

An NSRulerView automatically increases the reserved thickness as necessary to that of its thickest marker. If you plan to use markers of varying sizes, you should set the reserved thickness beforehand to that of the thickest one in order to avoid retiling of the NSScrollView.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [reservedThicknessForMarkers](#) (page 16)
- [setMarkers:](#) (page 18)
- [setReservedThicknessForAccessoryView:](#) (page 20)
- [thicknessRequiredInRuler](#) (NSRulerMarker)

**Related Sample Code**

Sketch-112

**Declared In**

NSRulerView.h

**setRuleThickness:**

Sets to *thickness* the thickness of the area where ruler hash marks and labels are drawn.

```
- (void)setRuleThickness:(CGFloat)thickness
```

**Discussion**

This value is the height of the ruler area for a horizontal ruler or the width of the ruler area for a vertical ruler. Rulers are by default 16.0 PostScript units thick. You should rarely need to change this layout attribute, but subclasses might do so to accommodate custom drawing.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [ruleThickness](#) (page 17)

**Declared In**

NSRulerView.h

**setScrollView:**

Sets the NSScrollView that owns the receiver to *scrollView*, without retaining it.

```
- (void)setScrollView:(NSScrollView *)scrollView
```

**Discussion**

This method is generally invoked only by the ruler's scroll view; you should rarely need to invoke it directly.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [scrollView](#) (page 17)
- [setHorizontalRulerView:](#) (NSScrollView)
- [setVerticalRulerView:](#) (NSScrollView)

**Declared In**

NSRulerView.h

**trackMarker:withMouseEvent:**

Tracks the mouse to add *aMarker* based on the initial mouse-down or mouse-dragged event *theEvent*.

- (BOOL)trackMarker:(NSRulerMarker \*)*aMarker* withMouseEvent:(NSEvent \*)*theEvent*

**Discussion**

Returns YES if the receiver adds *aMarker*, NO if it doesn't. This method works by sending trackMouse:adding: to *aMarker* with *theEvent* and YES as arguments.

An application typically invokes this method in one of two cases. In the simpler case, the client view can implement `rulerView:handleMouseDown:` (page 23) to invoke this method when the user presses the mouse button while the cursor is in the NSRulerView's ruler area. This technique is appropriate when it's clear what kind of marker will be added by clicking the ruler area. The second, more general, case involves the application providing a palette of different kinds of markers that can be dragged onto the ruler, from the ruler's accessory view or from some other place. With this technique the palette tracks the cursor until it enters the ruler view, at which time it hands over control to the ruler view by invoking `trackMarker:withMouseEvent:` (page 22).

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- `addMarker:` (page 10)
- `setMarkers:` (page 18)

**Declared In**

NSRulerView.h

## Delegate Methods

**rulerView:didAddMarker:**

Informs the client that *aRulerView* allowed the user to add *aMarker*.

- (void)rulerView:(NSRulerView \*)*aRulerView* didAddMarker:(NSRulerMarker \*)*aMarker*

**Discussion**

The client can take whatever action it needs based on this message, such as adding a new tab stop to the selected paragraph or creating a layout guideline.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- `representedObject` (NSRulerMarker)
- `markerLocation` (NSRulerMarker)

**Declared In**

NSRulerView.h

**rulerView:didMoveMarker:**

Informs the client that *aRulerView* allowed the user to move *aMarker*.

```
- (void)rulerView:(NSRulerView *)aRulerView didMoveMarker:(NSRulerMarker *)aMarker
```

**Discussion**

The client can take whatever action it needs based on this message, such as updating the location of a tab stop in the selected paragraph, moving a layout guideline, or resizing a graphics element.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- `representedObject` (NSRulerMarker)
- `markerLocation` (NSRulerMarker)

**Declared In**

NSRulerView.h

**rulerView:didRemoveMarker:**

Informs the client that *aRulerView* allowed the user to remove *aMarker*.

```
- (void)rulerView:(NSRulerView *)aRulerView didRemoveMarker:(NSRulerMarker *)aMarker
```

**Discussion**

The client can take whatever action it needs based on this message, such as deleting a tab stop from the paragraph style or removing a layout guideline.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- `representedObject` (NSRulerMarker)

**Declared In**

NSRulerView.h

**rulerView:handleMouseDown:**

Informs the client that the user has pressed the mouse button while the cursor is in the ruler area of *aRulerView*.

```
- (void)rulerView:(NSRulerView *)aRulerView handleMouseDown:(NSEvent *)theEvent
```

**Discussion**

*theEvent* is the mouse-down event that triggered the message. The client view can implement this method to perform an action such as adding a new marker using `trackMarker:withMouseEvent:` (page 22) or adding layout guidelines.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSRulerView.h

**rulerView:shouldAddMarker:**

Requests permission for *aRulerView* to add *aMarker*, an NSRulerMarker being dragged onto the ruler by the user.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldAddMarker:(NSRulerMarker *)aMarker
```

**Discussion**

If the client returns YES the ruler view accepts the new marker and begins tracking its movement; if the client returns NO the ruler view refuses the new marker.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [rulerView:willAddMarker:atLocation:](#) (page 25)

**Declared In**

NSRulerView.h

**rulerView:shouldMoveMarker:**

Requests permission for *aRulerView* to move *aMarker*.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldMoveMarker:(NSRulerMarker *)aMarker
```

**Discussion**

If the client returns YES the ruler view allows the user to move the marker; if the client returns NO the marker doesn't move.

The user's ability to move a marker is typically set on the marker itself, using NSRulerMarker's `setMovable:` method. You should use this client view method only when the marker's movability can vary depending on a variable condition (for example, if graphic items can be locked down to prevent them from being inadvertently moved).

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [rulerView:willMoveMarker:toLocation:](#) (page 25)

**Declared In**

NSRulerView.h

**rulerView:shouldRemoveMarker:**

Requests permission for *aRulerView* to remove *aMarker*.



```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldRemoveMarker:(NSRulerMarker *)aMarker
```

#### Discussion

If the client returns YES the ruler view allows the user to remove the marker; if the client returns NO the marker is kept pinned to the ruler's baseline. This message is sent as many times as needed while the user drags the marker.

The user's ability to remove a marker is typically set on the marker itself, using NSRulerMarker's `setRemovable:` method. You should use this client view method only when the marker's removability can vary while the user drags it (for example, if the user must press the Shift key to remove a marker).

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

NSRulerView.h

## rulerView:willAddMarker:atLocation:

Informs the client that *aRulerView* will add the new NSRulerMarker, *aMarker*.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willAddMarker:(NSRulerMarker *)aMarker atLocation:(CGFloat)location
```

#### Discussion

*location* is the marker's tentative new location, expressed in the client view's coordinate system. The value returned by the client view is actually used; the client can simply return *location* unchanged or adjust it as needed. For example, it may snap the location to a grid. This message is sent repeatedly to the client as the user drags the marker.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [rulerView:willMoveMarker:toLocation:](#) (page 25)

#### Declared In

NSRulerView.h

## rulerView:willMoveMarker:toLocation:

Informs the client that *aRulerView* will move *aMarker*, an NSRulerMarker already on the ruler view.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willMoveMarker:(NSRulerMarker *)aMarker toLocation:(CGFloat)location
```

#### Discussion

*location* is the marker's tentative new location, expressed in the client view's coordinate system. The value returned by the client view is actually used; the client can simply return *location* unchanged or adjust it as needed. For example, it may snap the location to a grid. This message is sent repeatedly to the client as the user drags the marker.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [rulerView:willAddMarker:atLocation:](#) (page 25)

**Declared In**

NSRulerView.h

**rulerView:willSetClientView:**

Informs the client view that *aRulerView* is about to be appropriated by *newClient*.

```
- (void)rulerView:(NSRulerView *)aRulerView willSetClientView:(NSView *)newClient
```

**Discussion**

The client view can use this opportunity to clear any cached information related to the ruler.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSRulerView.h

## Constants

**NSRulerOrientation**

These constants are defined to specify a ruler's orientation and are used by [orientation](#) (page 15) and [setOrientation:](#) (page 19).

```
typedef enum {
    NSHorizontalRuler,
    NSVerticalRuler
} NSRulerOrientation;
```

**Constants**

NSHorizontalRuler

Ruler is oriented horizontally.

Available in Mac OS X v10.0 and later.

Declared in NSRulerView.h.

NSVerticalRuler

Ruler is oriented vertically.

Available in Mac OS X v10.0 and later.

Declared in NSRulerView.h.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSRulerView.h



# Document Revision History

---

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

| Date       | Notes   |
|------------|---|
| 2006-05-23 | First publication of this content as a separate document. |

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

accessoryView **instance method** [10](#)  
addMarker: **instance method** [10](#)

## B

---

baselineLocation **instance method** [11](#)

## C

---

clientView **instance method** [11](#)

## D

---

drawHashMarksAndLabelsInRect: **instance method** [11](#)  
drawMarkersInRect: **instance method** [12](#)

## I

---

initWithScrollView:orientation: **instance method** [12](#)  
invalidateHashMarks **instance method** [13](#)  
isFlipped **instance method** [13](#)

## M

---

markers **instance method** [13](#)  
measurementUnits **instance method** [14](#)  
moveRulerlineFromLocation:toLocation: **instance method** [14](#)

## N

---

NSHorizontalRuler **constant** [26](#)  
NSRulerOrientation **data type** [26](#)  
NSVerticalRuler **constant** [26](#)

## O

---

orientation **instance method** [15](#)  
originOffset **instance method** [15](#)

## R

---

registerUnitWithName:abbreviation:  
    unitToPointsConversionFactor:stepUpCycle:  
    stepDownCycle: **class method** [9](#)  
removeMarker: **instance method** [15](#)  
requiredThickness **instance method** [16](#)  
reservedThicknessForAccessoryView **instance method** [16](#)  
reservedThicknessForMarkers **instance method** [16](#)  
rulerView:didAddMarker: <NSView> **delegate method** [22](#)  
rulerView:didMoveMarker: <NSView> **delegate method** [23](#)  
rulerView:didRemoveMarker: <NSView> **delegate method** [23](#)  
rulerView:handleMouseDown: <NSView> **delegate method** [23](#)  
rulerView:shouldAddMarker: <NSView> **delegate method** [24](#)  
rulerView:shouldMoveMarker: <NSView> **delegate method** [24](#)  
rulerView:shouldRemoveMarker: <NSView> **delegate method** [24](#)  
rulerView:willAddMarker:atLocation: <NSView> **delegate method** [25](#)  
rulerView:willMoveMarker:toLocation: <NSView> **delegate method** [25](#)

rulerView:willSetClientView: <NSView> delegate  
method 26  
ruleThickness instance method 17

## S

---

scrollView instance method 17  
setAccessoryView: instance method 17  
setClientView: instance method 18  
setMarkers: instance method 18  
setMeasurementUnits: instance method 19  
setOrientation: instance method 19  
setOriginOffset: instance method 19  
setReservedThicknessForAccessoryView: instance  
method 20  
setReservedThicknessForMarkers: instance method  
20  
setRuleThickness: instance method 21  
setScrollView: instance method 21

## T

---

trackMarker:withMouseEvent: instance method 22