# NSRulerMarker

| | |
|---|---|
| **Inherits From:** | NSObject |
| **Conforms To:** | NSCopying<br>NSObject (NSObject) |
| **Declared In:** | AppKit/NSRulerMarker.h |

## Class Description

An NSRulerMarker displays a symbol on an NSRulerView, indicating a location for whatever graphic element it represents in the client of the NSRulerView (for example, a margin or tab setting, or the edges of a graphic on the page). A ruler marker comprises three primary attributes: the image it displays on the NSRulerView, the location of that image, and the object it represents. The **setImage:**, **setMarkerLocation:** and **setRepresentedObject:** methods set each of these attributes, respectively. In addition, a ruler marker records an offset for the image, allowing it to be placed relative to the marker location much in the way a cursor's hot spot relates a cursor image to the mouse location; the **setImageOrigin:** method establishes this offset.

Most of these attributes are set upon initialization by the **initWithRulerView:markerLocation:image:imageOrigin:** method. New ruler markers don't have represented objects; the client typically establishes the represented object in its **rulerView:didAddMarker:** method. A new NSRulerMarker can be moved around in its NSRulerView, but not removed from it. The **setMovable:** and **setRemovable:** methods alter these default settings.

Represented objects allow the NSRulerView's client to distinguish among different attributes of the selection. In the NSRulerView client methods, the client can retrieve the marker's represented object to determine what attribute to alter. Generic attributes can be represented by string or other value objects, such as the edge names "Left", "Right", "Top", and "Bottom". Attributes already implemented as objects can be represented by those objects. For example, the OPENSTEP text system records tab stops as NSTextTab objects, which include the tab location and its alignment. When an NSTextView is the client view of a ruler, it simply makes the NSTextTabs the represented objects of the ruler markers.

## Adopted Protocols

| | |
|---|---|
| NSCopying | – copyWithZone: |

## Method Types

| | |
|---|---|
| Creating instances | – initWithRulerView:markerLocation:image:imageOrigin: |
| Getting the ruler view | – ruler |
| Setting the image | – setImage:<br>– image<br>– setImageOrigin:<br>– imageOrigin<br>– imageRectInRuler<br>– thicknessRequiredInRuler |
| Setting movability | – setMovable:<br>– isMovable<br>– setRemovable:<br>– isRemovable |
| Setting the location | – setMarkerLocation:<br>– markerLocation |
| Setting the represented object | – setRepresentedObject:<br>– representedObject |
| Drawing and event handling | – drawRect:<br>– isDragging<br>– trackMouse:adding: |

## Instance Methods

### drawRect:

– (void)**drawRect:**(NSRect)*aRect*

Draws the part of the receiver's image that intersects *aRect* in the NSRulerView's coordinate system.

**See also:** – **imageRectInRuler**

### image

– (NSImage *)**image**

Returns the NSImage object displayed by the receiver.

**See also:** – **setImage:**

## imageOrigin

– (NSPoint)**imageOrigin**

Returns the point in the receiver's image that's positioned at the receiver's location on the NSRulerView, expressed in the image's coordinate system.

For a horizontal ruler, the *x* coordinate of the image origin is aligned with the location of the marker, and the *y* coordinate lies on the baseline of the ruler. For vertical rulers, the *y* coordinate of the image origin is the location, and the *x* coordinate lies on the baseline.

**See also:** – **setImageOrigin:**, – **imageRectInRuler**

## imageRectInRuler

– (NSRect)**imageRectInRuler**

Returns the rectangle occupied by the marker's image, in the NSRulerView's coordinate system, accounting for whether the NSRulerView's coordinate system is flipped.

**See also:** – **drawRect:**, – **thicknessRequiredInRuler**

## initWithRulerView:markerLocation:image:imageOrigin:

– (id)**initWithRulerView:**(NSRulerView *)*aRulerView*
    **markerLocation:**(float)*location*
    **image:**(NSImage *)*anImage*
    **imageOrigin:**(NSPoint)*imageOrigin*

Initializes a newly allocated NSRulerMarker object, associating it with (but not adding it to) *aRulerView* and assigning the attributes provided. *location* is the *x* or *y* position of the marker in the client view's coordinate system, depending on whether the NSRulerView is horizontal or vertical. *anImage* is the image displayed at the marker location, and *imageOrigin* is the point within the image that's positioned at the marker location, expressed in pixels relative to the lower-left corner of the image. This method raises an NSInvalidArgumentException if *aRulerView* or *anImage* is **nil**.

**Note:** The image used to draw the marker must be appropriate for the orientation of the ruler. Markers may need to look different on a horizontal ruler than on a vertical ruler, and the NSRulerView neither scales nor rotates the images.

To add the new ruler marker to *aRulerView*, use either of NSRulerView's **addMarker:** or **trackMarker:withMouseEvent:** methods. **addMarker:** immediately puts the marker on the ruler, while **trackMarker:withMousEvent:** allows the client view to intercede in the addition and placement of the marker.

A new ruler marker is can be moved on its NSRulerView, but not removed. Use **setMovable:** and **setRemovable:** to change these attributes. The new ruler marker also has no represented object; use **setRepresentedObject:** to provide or change it.

This method retains *anImage,* since it's an essential part of the ruler marker, but doesn't retain *aRulerView* (the NSRulerView instead retains the new marker when it's added). This method is the designated initializer for the NSRulerMarker class. Returns **self**.

**See also:** – **setMarkerLocation:**, – **setImage:**, – **setImageOrigin:**

## isDragging

– (BOOL)**isDragging**

Returns YES if the receiver is being dragged, NO otherwise.

**See also:** – **trackMouse:adding:**

## isMovable

– (BOOL)**isMovable**

Returns YES if the user can move the receiver on its NSRulerView, NO otherwise. NSRulerMarkers are by default movable.

**See also:** – **setMovable:**, – **isRemovable**

## isRemovable

– (BOOL)**isRemovable**

Returns YES if the user can remove the receiver from its NSRulerView, NO otherwise. NSRulerMarkers cannot by default be removed from their NSRulerViews.

**See also:** – **setRemovable:**, – **isMovable**

## markerLocation

– (float)**makerLocation**

Returns the location of the receiver in the coordinate system of the NSRulerView's client view. This is an *x* position for a horizontal ruler, a *y* position for a vertical ruler.

**See also:** – **setMarkerLocation:**

## representedObject

– (id <NSCopying>)**representedObject**

Returns the object that the receiver represents, as explained in the class description.

**See also:** – **setRepresentedObject:**

## ruler

– (NSRulerView *)**ruler**

Returns the NSRulerView that the receiver belongs to.

**See also:** – **addMarker:** (NSRulerView)

## setImage:

– (void)**setImage:**(NSImage *)*anImage*

Sets the receiver's image to *anImage*.

**See also:** – **image**, – **setImageOrigin:**

## setImageOrigin:

– (void)**setImageOrigin:**(NSPoint)*aPoint*

Sets the point in the receiver's image that's positioned at the receiver's location on the NSRulerView to that at *aPoint*. This point is always expressed in pixels relative to the lower-left corner of the image.

For a horizontal ruler, the *x* coordinate of the image origin is aligned with the location of the marker, and the *y* coordinate lies on the baseline of the ruler. For vertical rulers, the *y* coordinate of the image origin is the location, and the *x* coordinate lies on the baseline.

**See also:** – **imageOrigin**, – **setImage:**, – **setMarkerLocation:**

## setMarkerLocation:

– (void)**setMarkerLocation:**(float)*location*

Sets the location of the receiver in the coordinate system of the NSRulerView's client view to *location*. This is an *x* position for a horizontal ruler, a *y* position for a vertical ruler.

**See also:** – **markerLocation**, – **setImageOrigin:**

### setMovable:

– (void)**setMovable:**(BOOL)*flag*

Controls whether the user can move the receiver in its NSRulerView. If *flag* is YES, the user can drag the marker image in the ruler. If *flag* is NO, the receiver is immovable. NSRulerMarkers are by default movable.

**See also:** – **isMovable**, – **setRemovable:**

### setRemovable:

– (void)**setRemovable:**(BOOL)*flag*

Controls whether the user can remove the receiver from its NSRulerView. If *flag* is YES, the user can drag the marker image off of the ruler. If *flag* is NO, the receiver can't be removed. NSRulerMarkers are by default not removable.

**See also:** – **isRemovable**, – **setMovable:**

### setRepresentedObject:

– (void)**setRepresentedObject:**(id <NSCopying>)*anObject*

Sets the object that the receiver represents to *anObject.* See the class description for more information on the represented object.

**See also:** – **representedObject**

### thicknessRequiredInRuler

– (float)**thicknessRequiredInRuler**

Returns the amount of the receiver's image that's displayed above or to the left of the NSRulerView's baseline, the height for a horizontal ruler or width for a vertical ruler.

**See also:** – **imageOrigin**

### trackMouse:adding:

– (BOOL)**trackMouse:**(NSEvent *)*theEvent* **adding:**(BOOL)*flag*

Handles user manipulation of the receiver in its NSRulerView. NSRulerView invokes this method automatically to add a new marker or to move or remove an existing marker. You should never need to invoke it directly.

If *flag* is YES, the receiver is a new marker being added to its NSRulerView. Before the receiver actually adds itself to the NSRulerView, it queries the NSRulerView’s client view using **rulerView:shouldAddMarker:**. If the client view responds to this method and returns NO, this method immediately returns NO and the new marker isn’t added.

If *flag* is NO, this method attempts to move or remove an existing marker, once again based on responses from the NSRulerView’s client view. If the receiver is neither movable nor removable, this method immediately returns NO. Further, if the NSRulerView’s client responds to **rulerView:shouldMoveMarker:** and returns NO, this method returns NO, indicating that the receiver can’t be moved.

If the receiver is being added or moved, this method queries the client view using **rulerView:willAddMarker:atLocation:** or **rulerView:willMoveMarker:toLocation:**, respectively. If the client responds to the method, the return value is used as the receiver’s location. These methods are invoked repeatedly as the receiver is dragged within the NSRulerView.

If the receiver is an existing marker being removed (dragged off the ruler), this method queries the client view using **rulerView:shouldRemoveMarker:**. If the client responds to this method and returns NO, the marker is pinned to the NSRulerView’s baseline (following the mouse on the baseline if it’s movable).

When the user releases the mouse, this method informs the client view of the marker’s new status using **rulerView:didAddMarker:**, **rulerView:didMoveMarker:**, or **rulerView:didRemoveMarker:** as appropriate. The client view can use this notification to set the marker’s represented object, modify its state and redisplay (for example, adjusting text layout around a new tab stop), or take whatever other action it might need. If *flag* is YES and the user dragged the new marker away from the ruler, the marker isn’t added, no message is sent, and this method returns NO.

See the NSRulerView class description for more information on these client methods.

**See also:** – **isMovable**, – **isRemovable**