Class

NSCell

A mechanism for displaying text or images in a view object without the overhead of a full `NSView` subclass.

macOS

```
@MainActor
class NSCell
```

Overview

Cells are used by most of the `NSControl` classes to implement their internal workings.

Designated Initializers

When subclassing `NSCell` you must implement all of the designated initializers. Those methods include `init()`, `init(coder:)`, `init(textCell:)`, and `init(imageCell:)`.

Topics

Initializing a Cell

`init(imageCell: NSImage?)`

Returns an `NSCell` object initialized with the specified image and set to have the cell's default menu.

`init(textCell: String)`

Returns an NSCell object initialized with the specified string and set to have the cell's default menu.

Managing Cell Values

`var objectValue: Any?`

The cell's value as an Objective-C object.

`var hasValidObjectValue: Bool`

A Boolean value that indicates whether the cell has a valid object value.

`var intValue: Int32`

The cell's value as an integer.

`var integerValue: Int`

The cell's value as an integer value.

`var stringValue: String`

The cell's value as a string.

`var doubleValue: Double`

The cell's value as a double-precision floating-point number.

`var floatValue: Float`

The cell's value as a single-precision floating-point number.

Managing Cell Attributes

`func setCellAttribute(NSCell.Attribute, to: Int)`

Sets the value for the specified cell attribute.

`func cellAttribute(NSCell.Attribute) -> Int`

Returns the value for the specified cell attribute.

`var type: NSCell.CellType`

The type of the cell.

`var isEnabled: Bool`

A Boolean value indicating whether the cell is currently enabled.

`var allowsUndo: Bool`

A Boolean value indicating whether the cell assumes responsibility for undo operations.

Managing Display Attributes

`var isBezeled: Bool`

A Boolean value indicating whether the cell has a bezeled border.

`var isBordered: Bool`

A Boolean value indicating whether the cell draws itself outlined with a plain border.

`var isOpaque: Bool`

A Boolean value indicating whether the cell is completely opaque.

~~`var controlTint: NSControlTint`~~

The cell's control tint.

Deprecated

`var backgroundStyle: NSView.BackgroundStyle`

The cell's background style.

`var interiorBackgroundStyle: NSView.BackgroundStyle`

The cell's interior background style.

`enum BackgroundStyle`

Background styles to apply to a view's cell.

Managing Cell State

`var allowsMixedState: Bool`

A Boolean value indicating whether the cell supports three states instead of two.

`var nextState: Int`

The cell's next state.

`func setNextState()`

Changes cell's state to the next value in the sequence.

`var state: NSControl.StateValue`

The cell's current state.

`struct StateValue`

A constant that indicates whether a control is on, off, or in a mixed state.

Modifying Textual Attributes

`var isEditable: Bool`

A Boolean value indicating whether the cell is editable.

`var isSelectable: Bool`

A Boolean value indicating whether the cell's text can be selected.

`var isScrollable: Bool`

A Boolean value indicating whether excess text scrolls past the cell's bounds.

`var alignment: NSTextAlignment`

The alignment of the cell's text.

`var font: NSFont?`

The font that the cell uses to display text.

`var lineBreakMode: NSLineBreakMode`

The line break mode to use when drawing text in the cell.

`var truncatesLastVisibleLine: Bool`

A Boolean value indicating whether the cell truncates text that does not fit within the cell's bounds.

`var wraps: Bool`

A Boolean value indicating whether the cell wraps text whose length that exceeds the cell's frame.

`var baseWritingDirection: NSWritingDirection`

The initial writing direction used to determine the actual writing direction for text.

`var attributedStringValue: NSAttributedString`

The cell's value as an attributed string.

`var allowsEditingTextAttributes: Bool`

A Boolean value indicating whether the cell allows the editing of its content's text attributes by the user.

`var importsGraphics: Bool`

A Boolean value indicating whether the cell supports the importation of images into its text.

`func setUpFieldEditorAttributes(NSText) -> NSText`

Configures the textual and background attributes of the receiver's field editor.

`var title: String`

The cell's title text.

Managing the Target and Action

`var action: Selector?`

The action performed by the cell.

`var target: AnyObject?`

The object that receives the cell's action messages.

`var isContinuous: Bool`

A Boolean value indicating whether the cell sends its action message continuously during mouse tracking.

`func sendAction(on: NSEvent.EventTypeMask) -> Int`

Sets the conditions on which the receiver sends action messages to its target.

Managing the Image

`var image: NSImage?`

The image displayed by the cell, if any.

Managing the Tag

`var tag: Int`

A tag for identifying the cell.

Formatting and Validating Data

`var formatter: Formatter?`

The cell's formatter object.

Managing Menus

`class var defaultMenu: NSMenu?`

Returns the default menu for instances of the cell.

`var menu: NSMenu?`

The cell's contextual menu.

`func menu(for: NSEvent, in: NSRect, of: NSView) -> NSMenu?`

Returns the menu associated with the cell and related to the specified event and frame.

Comparing Cells

`func compare(Any) -> ComparisonResult`

Compares the string values of the receiver another cell, disregarding case.

Respond to Keyboard Events

`var acceptsFirstResponder: Bool`

A Boolean value indicating whether the cell accepts first responder status.

`var showsFirstResponder: Bool`

A Boolean value indicating whether the cell provides a visual indication that it is the first responder.

`var refusesFirstResponder: Bool`

A Boolean value indicating whether the cell refuses the first responder status.

`func performClick(Any?)`

Simulates a single mouse click on the receiver.

Deriving Values

`func takeObjectValueFrom(Any?)`

Sets the value of the receiver's cell to the object value obtained from the specified object.

`func takeIntegerValueFrom(Any?)`

Sets the value of the receiver's cell to an integer value obtained from the specified object.

`func takeIntValueFrom(Any?)`

Sets the value of the receiver's cell to an integer value obtained from the specified object.

`func takeStringValueFrom(Any?)`

Sets the value of the receiver's cell to the string value obtained from the specified object.

`func takeDoubleValueFrom(Any?)`

Sets the value of the receiver's cell to a double-precision floating-point value obtained from the specified object.

`func takeFloatValueFrom(Any?)`

Sets the value of the receiver's cell to a single-precision floating-point value obtained from the specified object.

Representing an Object

`var representedObject: Any?`

The object represented by the cell.

Tracking the Mouse

`func trackMouse(with: NSEvent, in: NSRect, of: NSView, untilMouseUp: Bool) -> Bool`

Initiates the mouse tracking behavior in a cell.

`func startTracking(at: NSPoint, in: NSView) -> Bool`

Begins tracking mouse events within the receiver.

`func continueTracking(last: NSPoint, current: NSPoint, in: NSView) -> Bool`

Returns a Boolean value that indicates whether mouse tracking should continue in the receiving cell.

`func stopTracking(last: NSPoint, current: NSPoint, in: NSView, mouseIsUp: Bool)`

Stops tracking mouse events within the receiver.

`var mouseDownFlags: Int`

The modifier flags for the last (left) mouse-down event.

`class var prefersTrackingUntilMouseUp: Bool`

Returns a Boolean value that indicates whether tracking stops when the cursor leaves the cell.

```
func getPeriodicDelay(UnsafeMutablePointer<Float>, interval: Unsafe
MutablePointer<Float>)
```

Returns the initial delay and repeat values for continuous sending of action messages to target objects.

Hit Testing

```
func hitTest(for: NSEvent, in: NSRect, of: NSView) -> NSCell.HitResult
```

Returns hit testing information for the receiver.

Managing the Cursor

```
func resetCursorRect(NSRect, in: NSView)
```

Sets the receiver to show the I-beam cursor while it tracks the mouse.

Handling Keyboard Alternatives

```
var keyEquivalent: String
```

The key equivalent associated with clicking the cell.

Dragging Cells

```
func draggingImageComponents(withFrame: NSRect, in: NSView) -> [
NSDraggingImageComponent]
```

Generates dragging image components with the specified frame in the view.

Managing Focus Rings

```
func drawFocusRingMask(withFrame: NSRect, in: NSView)
```

Draws the focus ring for the control.

```
func focusRingMaskBounds(forFrame: NSRect, in: NSView) -> NSRect
```

Returns the bounds of the focus ring mask.

```
class var defaultFocusRingType: NSFocusRingType
```

Returns the default type of focus ring for the receiver.

```
var focusRingType: NSFocusRingType
```

The type of focus ring to use with the associated view.

Determining Cell Size

`func calcDrawInfo(NSRect)`

Recalculates the cell geometry.

`var cellSize: NSSize`

The minimum size needed to display the cell.

`func cellSize(forBounds: NSRect) -> NSSize`

Returns the minimum size needed to display the receiver, constraining it to the specified rectangle.

`func drawingRect(forBounds: NSRect) -> NSRect`

Returns the rectangle within which the receiver draws itself

`func imageRect(forBounds: NSRect) -> NSRect`

Returns the rectangle in which the receiver draws its image.

`func titleRect(forBounds: NSRect) -> NSRect`

Returns the rectangle in which the receiver draws its title text.

`var controlSize: NSControl.ControlSize`

The size of the cell.

Drawing and Highlighting

`func draw(withFrame: NSRect, in: NSView)`

Draws the receiver's border and then draws the interior of the cell.

`func highlightColor(withFrame: NSRect, in: NSView) -> NSColor?`

Returns the color the receiver uses when drawing the selection highlight.

`func drawInterior(withFrame: NSRect, in: NSView)`

Draws the interior portion of the receiver, which includes the image or text portion but does not include the border.

`var controlView: NSView?`

The view associated with the cell.

`func highlight(Bool, withFrame: NSRect, in: NSView)`

Redraws the receiver with the specified highlight setting.

`var isHighlighted: Bool`

A Boolean value indicating whether the cell has a highlighted appearance.

Editing and Selecting Text

`func edit(withFrame: NSRect, in: NSView, editor: NSText, delegate: Any?, event: NSEvent?)`

Begins editing of the receiver's text using the specified field editor.

`func select(withFrame: NSRect, in: NSView, editor: NSText, delegate: Any?, start: Int, length: Int)`

Selects the specified text range in the cell's field editor.

`var sendsActionOnEndEditing: Bool`

A Boolean value indicating whether the cell's control object sends its action message when the user finishes editing the cell's text.

`func endEditing(NSText)`

Ends the editing of text in the receiver using the specified field editor.

`var wantsNotificationForMarkedText: Bool`

A Boolean value indicating whether the cell's field editor should post text change notifications.

`func fieldEditor(for: NSView) -> NSTextView?`

Returns a custom field editor for editing in the view.

`var usesSingleLineMode: Bool`

A Boolean value indicating whether the cell restricts layout and rendering of text to a single line.

Managing Expansion Frames

`func expansionFrame(withFrame: NSRect, in: NSView) -> NSRect`

Returns the expansion cell frame for the receiver.

`func draw(withExpansionFrame: NSRect, in: NSView)`

Instructs the receiver to draw in an expansion frame.

User Interface Layout Direction

`var userInterfaceLayoutDirection: NSUserInterfaceLayoutDirection`

The layout direction of the user interface.

Constants

`enum CellType`

Constants for specifying how a cell represents its data (as text or as an image).

`enum Attribute`

Constants for specifying how a button behaves when pressed and how it displays its state.

`enum ImagePosition`

A constant for specifying the position of a button's image relative to its title.

`enum NSImageScaling`

Constants that specify a cell's image scaling behavior.

~~`typealias StateValue`~~

Constants for specifying a cell's state and are used mostly for buttons.

Deprecated

`struct StyleMask`

Constants for specifying what happens when a button is pressed or is displaying its alternate state.

`enum NSControlTint`

Constants for specifying a cell's tint color.

`enum ControlSize`

A constant for specifying a cell's size.

`struct HitResult`

Constants used by the `hitTest(for:in:of:)` method to determine the effect of an event.

`enum BackgroundStyle`

Background styles to apply to a view's cell.

Deprecated Scaling Constants

These are deprecated scaling constants.

Data Entry Types

These constants specify how a cell formats numeric data.

Notifications

~~`class let currentControlTintDidChangeNotification: NSNotification.Name`~~

Sent after the user changes control tint preference.

Deprecated

Initializers

`init()`

`init(coder: NSCoder)`

Relationships

Inherits From

`NSObject`

Inherited By

`NSActionCell`
`NSBrowserCell`
`NSImageCell`
`NSTextAttachmentCell`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSAccessibilityElementProtocol`

NSAccessibilityProtocol
NSCoding
NSCopying
NSObjectProtocol
NSUserInterfaceItemIdentification
Sendable

See Also

View fundamentals

`class` NSView

The infrastructure for drawing, printing, and handling events in an app.

`class` NSControl

A specialized view, such as a button or text field, that notifies your app of relevant events using the target-action design pattern.

`class` NSActionCell

An active area inside a control.