NSMatrix Class Reference

Cocoa > User Experience

2008-10-15

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

Simultaneously published in the United States and Canada.

Contents

NSMatrix Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Matrix Programming Guide for Cocoa
Declared in	NSMatrix.h
Related sample code	DatePicker OpenGLCompositorLab Quartz Composer WWDC 2005 TextEdit Sketch-112 TextEditPlus

Overview

`NSMatrix` is a class used for creating groups of `NSCell` objects that work together in various ways.

The cells in an `NSMatrix` object are numbered by row and column, each starting with 0; for example, the top left `NSCell` would be at (0, 0), and the `NSCell` that's second down and third across would be at (1, 2). The `NSMatrix` class has the notion of a single selected cell, which is the cell that was most recently clicked or that was so designated by a `selectCellAtRow:column:` (page 37) or `selectCellWithTag:` (page 38) message. The selected cell is the cell chosen for action messages except for `performClick:` (`NSCell`), which is assigned to the key cell. (The key cell is generally identical to the selected cell, but can be given click focus while leaving the selected cell unchanged.) If the user has selected multiple cells, the selected cell is the one lowest and furthest to the right in the matrix of cells.

Tasks

Initializing an NSMatrix Object

- `initWithFrame:` (page 25)
  Initializes a newly allocated matrix with the specified frame.
- `initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:` (page 25)
  Initializes and returns a newly allocated matrix of the specified size using cells of the given class.
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 26)
  Initializes and returns a newly allocated matrix of the specified size using the given cell as a prototype.

Configuring the Matrix Object

- `setMode:` (page 48)
  Sets the selection mode of the receiver.
- `mode` (page 31)
  Returns the selection mode of the matrix.
- `setAllowsEmptySelection:` (page 42)
  Sets whether a radio-mode matrix allows an empty selection.
- `allowsEmptySelection` (page 16)
  Returns a Boolean value indicating whether a radio-mode matrix supports an empty selection.
- `setSelectionByRect:` (page 49)
  Sets whether the user can select a rectangle of cells in the receiver by dragging the cursor.
- `isSelectionByRect` (page 30)
  Returns a Boolean value indicating whether the user can drag the cursor to select a rectangle of cells in the matrix.

Managing the Cell Class

- `setCellClass:` (page 44)
  Configures the receiver to use instances of the specified class when creating new cells.
- `cellClass` (page 18)
  Returns the class that the matrix uses to create new cells.
- `setPrototype:` (page 48)
  Sets the prototype cell that's copied whenever the matrix creates a new cell.
- `prototype` (page 34)
  Returns the prototype cell that's copied when a new cell is created.,

Laying Out the Cells of the Matrix

- `addColumn` (page 14)
  Adds a new column of cells to the right of the last column.

- `addColumnWithCells:` (page 14)
  Adds a new column of cells to the right of the last column, using the given cells.
- `addRow` (page 15)
  Adds a new row of cells below the last row.
- `addRowWithCells:` (page 16)
  Adds a new row of cells below the last row, using the specified cells.
- `cellFrameAtRow:column:` (page 19)
  Returns the frame rectangle of the cell that would be drawn at the specified location.
- `cellSize` (page 19)
  Returns the size of each cell in the matrix.
- `getNumberOfRows:columns:` (page 23)
  Obtains the number of rows and columns in the receiver.
- `insertColumn:` (page 27)
  Inserts a new column of cells at the specified location. .
- `insertColumn:withCells:` (page 27)
  Inserts a new column of cells before the specified column, using the given cells.
- `insertRow:` (page 28)
  Inserts a new row of cells before the specified row.
- `insertRow:withCells:` (page 28)
  Inserts a new row of cells before the specified row, using the given cells.
- `intercellSpacing` (page 29)
  Returns the spacing between cells in the matrix.
- `makeCellAtRow:column:` (page 31)
  Creates a new cell at the location specified by the given row and column in the receiver.
- `numberOfColumns` (page 33)
  Returns the number of columns in the receiver.
- `numberOfRows` (page 33)
  Returns the number of rows in the receiver.
- `putCell:atRow:column:` (page 34)
  Replaces the cell at the specified row and column with the new cell.
- `removeColumn:` (page 35)
  Removes the specified column at from the receiver.
- `removeRow:` (page 35)
  Removes the specified row from the receiver.
- `renewRows:columns:` (page 36)
  Changes the number of rows and columns in the receiver.
- `setCellSize:` (page 45)
  Sets the width and height of each of the cells in the matrix.
- `setIntercellSpacing:` (page 47)
  Sets the spacing between cells in the matrix.
- `sortUsingFunction:context:` (page 53)
  Sorts the receiver's cells in ascending order as defined by the specified comparison function.
- `sortUsingSelector:` (page 53)
  Sorts the receiver's cells in ascending order as defined by the comparison method.

Finding Matrix Coordinates

- `getRow:column:forPoint:` (page 24)
  Indicates whether the specified point lies within one of the cells of the matrix and returns the location of the cell within which the point lies.
- `getRow:column:ofCell:` (page 24)
  Searches the receiver for the specified cell and returns the row and column of the cell

Managing Attributes of Individual Cells

- `setState:atRow:column:` (page 50)
  Sets the state of the cell at specified location.
- `setToolTip:forCell:` (page 52)
  Sets the tooltip for the cell.
- `toolTipForCell:` (page 57)
  Returns the tooltip for the specified cell.

Selecting and Deselecting Cells

- `selectCellAtRow:column:` (page 37)
  Selects the cell at the specified row and column within the receiver.
- `selectCellWithTag:` (page 38)
  Selects the last cell with the given tag.
- `selectAll:` (page 37)
  Selects and highlights all cells in the receiver.
- `setKeyCell:` (page 48)
  Sets the cell that will be clicked when the user presses the Space bar.
- `keyCell` (page 30)
  Returns the cell that will be clicked when the user presses the Space bar.
- `setSelectionFrom:to:anchor:highlight:` (page 50)
  Programmatically selects a range of cells.
- `deselectAllCells` (page 21)
  Deselects all cells in the receiver and, if necessary, redisplays the receiver.
- `deselectSelectedCell` (page 21)
  Deselects the selected cell or cells.

Finding Cells

- `selectedCell` (page 38)
  Returns the most recently selected cell.
- `selectedCells` (page 39)
  Returns the receiver's selected and highlighted cells.

- `selectedColumn` (page 39)
  Returns the column of the selected cell.
- `selectedRow` (page 40)
  Returns the row of the selected cell.
- `cellAtRow:column:` (page 17)
  Returns the cell at the specified row and column.
- `cellWithTag:` (page 20)
  Searches the receiver and returns the last cell matching the specified tag.
- `cells` (page 19)
  Returns the cells of the matrix.

Modifying Graphics Attributes

- `backgroundColor` (page 17)
  Returns the background color of the matrix.
- `cellBackgroundColor` (page 18)
  Returns the background color of the matrix's cells.
- `drawsBackground` (page 22)
  Returns a Boolean value indicating whether the matrix draws its background.
- `drawsCellBackground` (page 23)
  Returns whether the matrix draws the background within each of its cells.
- `setBackgroundColor:` (page 43)
  Sets the background color for the receiver and redraws the receiver.
- `setCellBackgroundColor:` (page 44)
  Sets the background color for the cells in the receiver
- `setDrawsBackground:` (page 46)
  Sets whether the receiver draws its background.
- `setDrawsCellBackground:` (page 47)
  Sets whether the receiver draws the background within each of its cells.

Editing Text in Cells

- `selectText:` (page 40)
  Selects text in the currently selected cell or in the key cell.
- `selectTextAtRow:column:` (page 40)
  Selects the text in the cell at the specified location and returns the cell.
- `textShouldBeginEditing:` (page 56)
  Requests permission to begin editing text.
- `textDidBeginEditing:` (page 54)
  Invoked when there's a change in the text after the receiver gains first responder status.
- `textDidChange:` (page 54)
  Invoked when a key-down event or paste operation occurs that changes the receiver's contents.

- `textShouldEndEditing:` (page 56)
  Requests permission to end editing.
- `textDidEndEditing:` (page 55)
  Invoked when text editing ends.

Setting Tab Key Behavior

- `setTabKeyTraversesCells:` (page 51)
  Sets whether pressing the Tab key advances the key cell to the next selectable cell.
- `tabKeyTraversesCells` (page 54)
  Returns a Boolean value indicating whether pressing the Tab key advances the key cell to the next selectable cell.

Managing the Delegate

- `delegate` (page 20)
  Returns the delegate for messages from the field editor.
- `setDelegate:` (page 45)
  Sets the delegate for messages from the field editor.

Resizing the Matrix and Its Cells

- `setAutosizesCells:` (page 43)
  Sets whether the cell sizes change when the receiver is resized.
- `autosizesCells` (page 16)
  Returns a Boolean value indicating whether the matrix automatically resizes it cells.
- `setValidateSize:` (page 52)
  Specifies whether the receiver's size information is validated.
- `sizeToCells` (page 52)
  Changes the width and the height of the receiver’s frame so it exactly contains the cells.

Scrolling Cells in the Matrix

- `setAutoscroll:` (page 43)
  Sets whether the receiver is automatically scrolled.
- `isAutoscroll` (page 29)
  Returns a Boolean value indicating whether the receiver is automatically scrolled.
- `setScrollable:` (page 49)
  Specifies whether the cells in the matrix are scrollable.
- `scrollCellToVisibleAtRow:column:` (page 37)
  Scrolls the receiver so the specified cell is visible.

Displaying and Highlighting Cells

- `drawCellAtRow:column:` (page 22)
  Displays the cell at the specified row and column.
- `highlightCell:atRow:column:` (page 25)
  Highlights or unhighlights the cell at the specified row and column location.

Managing and Sending Action Messages

- `sendAction` (page 41)
  If the selected cell has both an action and a target, sends its action to its target.
- `sendAction:to:forAllCells:` (page 41)
  Iterates through the cells in the receiver, sending the specified selector to an object for each cell.
- `setDoubleAction:` (page 46)
  Sets the action sent to the target of the receiver when the user double-clicks a cell.
- `doubleAction` (page 21)
  Returns the matrix's double-click action method.
- `sendDoubleAction` (page 42)
  Sends the double-click action message to the target of the receiver.

Handling Event and Action Messages

- `acceptsFirstMouse:` (page 13)
  Returns a Boolean value indicating whether the receiver accepts the first mouse.
- `mouseDown:` (page 32)
  Responds to a mouse-down event.
- `mouseDownFlags` (page 32)
  Returns the flags in effect at the mouse-down event that started the current tracking session.
- `performKeyEquivalent:` (page 33)
  Looks for a cell that has the given key equivalent and, if found, makes that cell respond as if clicked.

Managing the Cursor

- `resetCursorRects` (page 36)
  Resets cursor rectangles so the cursor becomes an I-beam over text cells.

Instance Methods

acceptsFirstMouse:

Returns a Boolean value indicating whether the receiver accepts the first mouse.

```
- (BOOL)acceptsFirstMouse:(NSEvent *)theEvent
```

Parameters

theEvent

This parameter is ignored.

Return Value

`NO` if the selection mode of the receiver is `NSListModeMatrix`, `YES` if the receiver is in any other selection mode. The receiver does not accept first mouse in `NSListModeMatrix` to prevent the loss of multiple selections.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `mode` (page 31)

Declared In

`NSMatrix.h`

addColumn

Adds a new column of cells to the right of the last column.

```
- (void)addColumn
```

Discussion

This method raises an `NSRangeException` if there are 0 rows or 0 columns. This method creates new cells as needed with `makeCellAtRow:column:` (page 31). Use `renewRows:columns:` (page 36) to add new cells to an empty matrix.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 36), new cells are created only if they are needed. This fact allows you to grow and shrink an `NSMatrix` without repeatedly creating and freeing the cells.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `cellClass` (page 18)
- `insertColumn:` (page 27)
- `prototype` (page 34)
- `addRow` (page 15)

Declared In

`NSMatrix.h`

addColumnWithCells:

Adds a new column of cells to the right of the last column, using the given cells.

```
- (void)addColumnWithCells:(NSArray *)newCells
```

Parameters

newCells

An array of objects to use when filling the new column starting with the object at index 0. Each object in should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`). The array should have a sufficient number of cells to fill the entire column. Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one column and enough rows for all the elements of *newCells*.

Discussion

This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertColumn:withCells:` (page 27)
- `addRowWithCells:` (page 16)

Declared In

`NSMatrix.h`

addRow

Adds a new row of cells below the last row.

```
- (void)addRow
```

Discussion

New cells are created as needed with `makeCellAtRow:column:` (page 31). This method raises an `NSRangeException` if there are 0 rows or 0 columns. Use `renewRows:columns:` (page 36) to add new cells to an empty matrix.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 36), then new cells are created only if they are needed. This fact allows you to grow and shrink an NSMatrix without repeatedly creating and freeing the cells.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `cellClass` (page 18)
- `insertRow:` (page 28)
- `prototype` (page 34)
- `addColumn` (page 14)

Declared In

`NSMatrix.h`

addRowWithCells:

Adds a new row of cells below the last row, using the specified cells.

```
- (void)addRowWithCells:(NSArray *)newCells
```

Parameters

newCells

An array of objects to use to fill the new row, starting with the object at index 0. Each object should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`). The array should contain a sufficient number of cells to fill the entire row. Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one row and enough columns for all the elements of *newCells*.

Discussion

This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertRow:withCells:` (page 28)
- `addColumnWithCells:` (page 14)

Declared In

`NSMatrix.h`

allowsEmptySelection

Returns a Boolean value indicating whether a radio-mode matrix supports an empty selection.

```
- (BOOL)allowsEmptySelection
```

Return Value

`YES` if it is possible to have no cells selected in a radio-mode matrix; otherwise `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `mode` (page 31)
- `setAllowsEmptySelection:` (page 42)

Declared In

`NSMatrix.h`

autosizesCells

Returns a Boolean value indicating whether the matrix automatically resizes it cells.

```
- (BOOL)autosizesCells
```

Return Value
`YES` if cells are resized proportionally to the receiver when its size changes (and intercell spacing is kept constant). `NO` if the cell size and intercell spacing remain constant.

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

See Also
`- setAutosizesCells:` (page 43)

Declared In
`NSMatrix.h`

backgroundColor

Returns the background color of the matrix.

```
- (NSColor *)backgroundColor
```

Return Value
The color used to draw the background of the receiver (the space between the cells).

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

See Also
`- cellBackgroundColor` (page 18)
`- drawsBackground` (page 22)
`- setBackgroundColor:` (page 43)

Declared In
`NSMatrix.h`

cellAtRow:column:

Returns the cell at the specified row and column.

```
- (id)cellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters
row
The number of the row containing the cell to return.
column
The number of the column containing the cell to return.

Return Value
The `NSCell` object at the specified row and column location specified, or `nil` if either *row* or *column* is outside the bounds of the receiver.

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

See Also
- `getRow:column:ofCell:` (page 24)

Related Sample Code
NewsReader

Declared In
`NSMatrix.h`

cellBackgroundColor

Returns the background color of the matrix's cells.

```
- (NSColor *)cellBackgroundColor
```

Return Value
The color used to fill the background of the receiver's cells.

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

See Also
- `backgroundColor` (page 17)
- `drawsCellBackground` (page 23)
- `setCellBackgroundColor:` (page 44)

Declared In
`NSMatrix.h`

cellClass

Returns the class that the matrix uses to create new cells.

```
- (Class)cellClass
```

Return Value
The subclass of `NSCell` that the receiver uses when creating new (empty) cells.

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

See Also
- `prototype` (page 34)
- `makeCellAtRow:column:` (page 31)
- `setCellClass:` (page 44)

Declared In
`NSMatrix.h`

cellFrameAtRow:column:

Returns the frame rectangle of the cell that would be drawn at the specified location.

```
- (NSRect)cellFrameAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row

The row of the cell.

column

The column of the cell.

Return Value
The frame rectangle of the cell (whether or not the specified cell actually exists).

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

See Also
- cellSize (page 19)

Declared In
NSMatrix.h

cells

Returns the cells of the matrix.

```
- (NSArray *)cells
```

Return Value
An array containing the cells of the receiver.

Discussion
The cells in the array are row-ordered; that is, the first row of cells appears first in the array, followed by the second row, and so forth.

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

See Also
- cellAtRow:column: (page 17)

Declared In
NSMatrix.h

cellSize

Returns the size of each cell in the matrix.

```
- (NSSize)cellSize
```

Return Value
The width and height of each cell in the receiver (all cells in an NSMatrix are the same size).

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

See Also
- `cellFrameAtRow:column:` (page 19)
- `intercellSpacing` (page 29)
- `setCellSize:` (page 45)

Declared In
`NSMatrix.h`

cellWithTag:

Searches the receiver and returns the last cell matching the specified tag.

```
- (id)cellWithTag:(NSInteger)anInt
```

Parameters

anInt
The tag of the cell to return.

Return Value
The last (when viewing the matrix as a row-ordered array) `NSCell` object that has a tag matching *anInt*, or `nil` if no such cell exists

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

See Also
- `selectCellWithTag:` (page 38)
- `setTag:` (NSActionCell)

Declared In
`NSMatrix.h`

delegate

Returns the delegate for messages from the field editor.

```
- (id)delegate
```

Return Value
The delegate.

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

See Also
- `textShouldBeginEditing:` (page 56)
- `textShouldEndEditing:` (page 56)
- `setDelegate:` (page 45)

Declared In
NSMatrix.h

deselectAllCells

Deselects all cells in the receiver and, if necessary, redisplays the receiver.

```
- (void)deselectAllCells
```

Discussion
If the selection mode is `NSRadioModeMatrix` and empty selection is not allowed, this method does nothing.

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

See Also
- `allowsEmptySelection` (page 16)
- `mode` (page 31)
- `selectAll:` (page 37)

Declared In
NSMatrix.h

deselectSelectedCell

Deselects the selected cell or cells.

```
- (void)deselectSelectedCell
```

Discussion
If the selection mode is `NSRadioModeMatrix` and empty selection is not allowed, or if nothing is currently selected, this method does nothing. This method doesn't redisplay the receiver.

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

See Also
- `allowsEmptySelection` (page 16)
- `mode` (page 31)
- `selectCellAtRow:column:` (page 37)

Declared In
NSMatrix.h

doubleAction

Returns the matrix's double-click action method.

```
- (SEL)doubleAction
```

Return Value
The action method sent by the receiver to its target when the user double-clicks an entry or `NULL` if there's no double-click action.

Discussion
The double-click action of an `NSMatrix` is sent after the appropriate single-click action (for the `NSCell` clicked or for the `NSMatrix` if the `NSCell` doesn't have its own action). If there is no double-click action and the `NSMatrix` doesn't ignore multiple clicks, the single-click action is sent twice.

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

See Also
- `action` (NSControl)
- `target` (NSControl)
- `ignoresMultiClick` (NSControl)
- `sendDoubleAction` (page 42)
- `setDoubleAction:` (page 46)

Declared In
`NSMatrix.h`

drawCellAtRow:column:

Displays the cell at the specified row and column.

```
- (void)drawCellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row
The row containing the cell to draw.

column
The column containing the cell to draw.

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

See Also
- `drawCell:` (NSControl)
- `drawCellInside:` (NSControl)

Declared In
`NSMatrix.h`

drawsBackground

Returns a Boolean value indicating whether the matrix draws its background.

```
- (BOOL)drawsBackground
```

Return Value
`YES` if the receiver draws its background (the space between the cells); otherwise `NO`.

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

See Also
- `backgroundColor` (page 17)
- `drawsCellBackground` (page 23)
- `setDrawsBackground:` (page 46)

Declared In
`NSMatrix.h`

drawsCellBackground

Returns whether the matrix draws the background within each of its cells.

```
- (BOOL)drawsCellBackground
```

Return Value
`YES` if the receiver draws the cell background; otherwise `NO`.

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

See Also
- `cellBackgroundColor` (page 18)
- `drawsBackground` (page 22)
- `setDrawsCellBackground:` (page 47)

Declared In
`NSMatrix.h`

getNumberOfRows:columns:

Obtains the number of rows and columns in the receiver.

```
- (void)getNumberOfRows:(NSInteger *)rowCount columns:(NSInteger *)columnCount
```

Parameters
rowCount
On return, the number of rows in the matrix.

columnCount
On return, the number of columns in the matrix.

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

See Also
- `numberOfColumns` (page 33)
- `numberOfRows` (page 33)

Declared In
`NSMatrix.h`

getRow:column:forPoint:

Indicates whether the specified point lies within one of the cells of the matrix and returns the location of the cell within which the point lies.

```
- (BOOL)getRow:(NSInteger *)row column:(NSInteger *)column forPoint:(NSPoint)aPoint
```

Parameters

row

On return, the row of the cell containing the specified point.

column

On return, the column of the cell containing the specified point.

aPoint

The point to locate; this point should be in the coordinate system of the receiver.

Return Value

`YES` if the point lies within one of the cells in the receiver; `NO` if the point falls outside the bounds of the receiver or lies within an intercell spacing.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- getRow:column:ofCell:` (page 24)

Declared In

`NSMatrix.h`

getRow:column:ofCell:

Searches the receiver for the specified cell and returns the row and column of the cell

```
- (BOOL)getRow:(NSInteger *)row column:(NSInteger *)column ofCell:(NSCell *)aCell
```

Parameters

row

On return, the row in which the cell is located.

column

On return, the column in which the cell is located.

aCell

The cell to locate within the matrix.

Return Value

`YES` if the cell is one of the cells in the receiver, `NO` otherwise.

Discussion

.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- getRow:column:forPoint:` (page 24)

Declared In
NSMatrix.h

highlightCell:atRow:column:

Highlights or unhighlights the cell at the specified row and column location.

```
- (void)highlightCell:(BOOL)flag atRow:(NSInteger)row column:(NSInteger)column
```

Parameters

flag
YES to highlight the cell; NO to unhighlight the cell.

row
The row containing the cell.

column
The column containing the cell.

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

Declared In
NSMatrix.h

initWithFrame:

Initializes a newly allocated matrix with the specified frame.

```
- (id)initWithFrame:(NSRect)frameRect
```

Parameters

frameRect
The frame with which to initialize the matrix.

Return Value
The NSMatrix, initialized with default parameters. The new NSMatrix contains no rows or columns. The default mode is NSRadioModeMatrix. The default cell class is NSActionCell.

Discussion
.

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

Declared In
NSMatrix.h

initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:

Initializes and returns a newly allocated matrix of the specified size using cells of the given class.

```
- (id)initWithFrame:(NSRect)frameRect mode:(NSInteger)aMode cellClass:(Class)classId
    numberOfRows:(NSInteger)numRows numberOfColumns:(NSInteger)numColumns
```

Parameters

frameRect

The matrix's frame.

aMode

The tracking mode for the matrix; this can be one of the modes described in `NSMatrixMode` (page 57).

classId

The class to use for any cells that the matrix creates and uses. This can be obtained by sending a `class` message to the desired subclass of `NSCell`.

numRows

The number of rows in the matrix.

numColumns

The number of columns in the matrix.

Return Value

The initialized instance of `NSMatrix`.

Discussion

This method is the designated initializer for matrices that add cells by creating instances of an `NSCell` subclass.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSMatrix.h`

initWithFrame:mode:prototype:numberOfRows:numberOfColumns:

Initializes and returns a newly allocated matrix of the specified size using the given cell as a prototype.

```
- (id)initWithFrame:(NSRect)frameRect mode:(NSInteger)aMode prototype:(NSCell
    *)aCell numberOfRows:(NSInteger)numRows numberOfColumns:(NSInteger)numColumns
```

Parameters

frameRect

The matrix's frame.

aMode

The tracking mode for the matrix; this can be one of the modes described in `NSMatrixMode` (page 57).

aCell

An instance of a subclass of `NSCell`, which the new matrix copies when it creates new cells.

numRows

The number of rows in the matrix.

numColumns

The number of columns in the matrix.

Discussion

This method is the designated initializer for matrices that add cells by copying an instance of an `NSCell` subclass.

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

Declared In
`NSMatrix.h`

insertColumn:

Inserts a new column of cells at the specified location. .

```
- (void)insertColumn:(NSInteger)column
```

Parameters

column

The number of the column before which the new column is inserted. If *column* is greater than the number of columns in the receiver, enough columns are created to expand the receiver to be *column* columns wide.

Discussion
New cells are created if needed with `makeCellAtRow:column:` (page 31). This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 36), new cells are created only if they're needed. This fact allows you to grow and shrink an `NSMatrix` without repeatedly creating and freeing the cells.

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

See Also
- `addColumn` (page 14)
- `insertRow:` (page 28)

Declared In
`NSMatrix.h`

insertColumn:withCells:

Inserts a new column of cells before the specified column, using the given cells.

```
- (void)insertColumn:(NSInteger)column withCells:(NSArray *)newCells
```

Parameters

column

The column at which to insert the new cells.

newCells

An array of objects to use to fill the new column, starting with the object at index 0. Each object should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`).

Discussion
If *column* is greater than the number of columns in the receiver, enough columns are created to expand the receiver to be *column* columns wide. *newCells* should either be empty or contain a sufficient number of cells to fill each new column. If *newCells* is `nil` or an array with no elements, the call is equivalent to calling `insertColumn:` (page 27). Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one column and enough rows for all the elements of *newCells*.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

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

See Also
- `addColumnWithCells:` (page 14)
- `insertRow:withCells:` (page 28)

Declared In
`NSMatrix.h`

insertRow:

Inserts a new row of cells before the specified row.

```
- (void)insertRow:(NSInteger)row
```

Parameters

row

The location at which to insert the new row. If this is greater than the number of rows in the receiver, enough rows are created to expand the receiver to be *row* rows high.

Discussion
New cells are created if needed with `makeCellAtRow:column:` (page 31). This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 36), then new cells are created only if they're needed. This fact allows you to grow and shrink an `NSMatrix` without repeatedly creating and freeing the cells.

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

See Also
- `addRow` (page 15)
- `insertColumn:` (page 27)

Declared In
`NSMatrix.h`

insertRow:withCells:

Inserts a new row of cells before the specified row, using the given cells.

```
- (void)insertRow:(NSInteger)row withCells:(NSArray *)newCells
```

Parameters

row

The location at which to insert the new row.

newCells

An array of objects to use when filling the new row, starting with the object at index 0. Each object in *newCells* should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`).

Discussion

If *row* is greater than the number of rows in the receiver, enough rows are created to expand the receiver to be *row* rows high. *newCells* should either be empty or contain a sufficient number of cells to fill each new row. If *newCells* is `nil` or an array with no elements, the call is equivalent to calling `insertRow:` (page 28). Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one row and enough columns for all the elements of *newCells*.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 52) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addRowWithCells:` (page 16)
- `insertColumn:withCells:` (page 27)

Declared In

`NSMatrix.h`

intercellSpacing

Returns the spacing between cells in the matrix.

```
- (NSSize)intercellSpacing
```

Return Value

The vertical and horizontal spacing between cells in the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `cellSize` (page 19)
- `setIntercellSpacing:` (page 47)

Declared In

`NSMatrix.h`

isAutoscroll

Returns a Boolean value indicating whether the receiver is automatically scrolled.

```
- (BOOL)isAutoscroll
```

Return Value
`YES` if the receiver will be automatically scrolled whenever the cursor is dragged outside the receiver after a mouse-down event within its bounds; otherwise `NO`.

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

See Also
- `scrollCellToVisibleAtRow:column:` (page 37)
- `setScrollable:` (page 49)

Declared In
`NSMatrix.h`

isSelectionByRect

Returns a Boolean value indicating whether the user can drag the cursor to select a rectangle of cells in the matrix.

```
- (BOOL)isSelectionByRect
```

Return Value
`YES` if the user can select a rectangle of cells in the receiver by dragging the cursor, `NO` otherwise.

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

See Also
- `setSelectionFrom:to:anchor:highlight:` (page 50)

Declared In
`NSMatrix.h`

keyCell

Returns the cell that will be clicked when the user presses the Space bar.

```
- (id)keyCell
```

Return Value
The cell that will be clicked when the user presses the Space bar.

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

See Also
- `tabKeyTraversesCells` (page 54)
- `setKeyCell:` (page 48)

Declared In
`NSMatrix.h`

makeCellAtRow:column:

Creates a new cell at the location specified by the given row and column in the receiver.

```
- (NSCell *)makeCellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row

The row in which to create the new cell.

column

The column in which to create the new cell.

Return Value
The newly created cell.

Discussion
If the receiver has a prototype cell, it's copied to create the new cell. If not, and if the receiver has a cell class set, it allocates and initializes (with `init`) an instance of that class. If the receiver hasn't had either a prototype cell or a cell class set, `makeCellAtRow:column:` creates an `NSActionCell`.

Your code should never invoke this method directly; it's used by `addRow` (page 15) and other methods when a cell must be created. It may be overridden to provide more specific initialization of cells.

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

See Also
- `addColumn` (page 14)
- `addRow` (page 15)
- `insertColumn:` (page 27)
- `insertRow:` (page 28)
- `setCellClass:` (page 44)
- `setPrototype:` (page 48)

Declared In
`NSMatrix.h`

mode

Returns the selection mode of the matrix.

```
- (NSMatrixMode)mode
```

Return Value
The selection mode of the receiver. Possible return values are defined in `NSMatrixMode` (page 57).

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

See Also
- `initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:` (page 25)
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 26)
- `setMode:` (page 48)

Declared In
NSMatrix.h

mouseDown:

Responds to a mouse-down event.

```
- (void)mouseDown:(NSEvent *)theEvent
```

Parameters

theEvent
The mouse-down event.

Discussion
A mouse-down event in a text cell initiates editing mode. A double click in any cell type except a text cell sends the double-click action of the receiver (if there is one) in addition to the single-click action.

Your code should never invoke this method, but you may override it to implement different mouse tracking than NSMatrix does. The response of the receiver depends on its selection mode, as explained in the class description.

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

See Also
- sendAction (page 41)
- sendDoubleAction (page 42)

Declared In
NSMatrix.h

mouseDownFlags

Returns the flags in effect at the mouse-down event that started the current tracking session.

```
- (NSInteger)mouseDownFlags
```

Return Value
The flags in effect when the mouse-down event is generated.

Discussion
The NSMatrix mouseDown: (page 32) method obtains these flags by sending a modifierFlags message to the event passed into mouseDown: (page 32). Use this method if you want to access these flags. This method is valid only during tracking; it isn't useful if the target of the receiver initiates another tracking loop as part of its action method (as a cell that pops up a pop-up list does, for example).

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

See Also
- sendActionOn: (NSCell)

Declared In
NSMatrix.h

numberOfColumns

Returns the number of columns in the receiver.

```
- (NSInteger)numberOfColumns
```

Return Value
The number of columns in the matrix.

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

See Also
- getNumberOfRows:columns: (page 23)

Declared In
NSMatrix.h

numberOfRows

Returns the number of rows in the receiver.

```
- (NSInteger)numberOfRows
```

Return Value
The number of rows in the receiver.

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

See Also
- getNumberOfRows:columns: (page 23)

Declared In
NSMatrix.h

performKeyEquivalent:

Looks for a cell that has the given key equivalent and, if found, makes that cell respond as if clicked.

```
- (BOOL)performKeyEquivalent:(NSEvent *)theEvent
```

Parameters
theEvent
The event containing the character for which to find a key equivalent.

Return Value
YES if a cell in the receiver responds to the key equivalent in *theEvent*, NO if no cell responds.

Discussion
If there's a cell in the receiver that has a key equivalent equal to the character in [*theEvent* `charactersIgnoringModifiers`] (taking into account any key modifier flags) and that cell is enabled, that cell is made to react as if the user had clicked it: by highlighting, changing its state as appropriate, sending its action if it has one, and then unhighlighting.

Your code should never send this message—it is sent when the receiver or one of its superviews is the first responder and the user presses a key. You may want to override this method to change the way key equivalents are performed or displayed or to disable them in your subclass.

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

Declared In
`NSMatrix.h`

prototype

Returns the prototype cell that's copied when a new cell is created.,

```
- (id)prototype
```

Return Value
The cell that the matrix copies whenever it creates a new cell, or `nil` if there is none.

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

See Also
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 26)
- `makeCellAtRow:column:` (page 31)
- `setPrototype:` (page 48)

Declared In
`NSMatrix.h`

putCell:atRow:column:

Replaces the cell at the specified row and column with the new cell.

```
- (void)putCell:(NSCell *)newCell atRow:(NSInteger)row column:(NSInteger)column
```

Parameters
newCell
The cell to insert into the matrix.

row
The row in which to place the new cell.

column
The column in which to place the new cell.

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

Declared In
NSMatrix.h

removeColumn:

Removes the specified column at from the receiver.

```
- (void)removeColumn:(NSInteger)column
```

Parameters

column
The column to remove.

Discussion
The column's cells are autoreleased. This method redraws the receiver. Your code should normally send `sizeToCells` (page 52) after invoking this method to resize the receiver so it fits the reduced cell count.

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

See Also
- `removeRow:` (page 35)
- `addColumn` (page 14)
- `insertColumn:` (page 27)

Declared In
NSMatrix.h

removeRow:

Removes the specified row from the receiver.

```
- (void)removeRow:(NSInteger)row
```

Parameters

row
The row to remove.

Discussion
The row's cells are autoreleased. This method redraws the receiver. Your code should normally send `sizeToCells` (page 52) after invoking this method to resize the receiver so it fits the reduced cell count.

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

See Also
- `removeColumn:` (page 35)
- `addRow` (page 15)
- `insertRow:` (page 28)

Declared In
NSMatrix.h

renewRows:columns:

Changes the number of rows and columns in the receiver.

```
- (void)renewRows:(NSInteger)newRows columns:(NSInteger)newCols
```

Parameters

newRows

The new number of rows in the matrix.

newCols

The new number of columns in the matrix.

Discussion

This method uses the same cells as before, creating new cells only if the new size is larger; it never frees cells. It doesn't redisplay the receiver. Your code should normally send `sizeToCells` (page 52) after invoking this method to resize the receiver so it fits the changed cell arrangement. This method deselects all cells in the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addColumn` (page 14)
- `addRow` (page 15)
- `removeColumn:` (page 35)
- `removeRow:` (page 35)

Related Sample Code

NewsReader

Declared In

`NSMatrix.h`

resetCursorRects

Resets cursor rectangles so the cursor becomes an I-beam over text cells.

```
- (void)resetCursorRects
```

Discussion

This method resets the cursor rectangles by sending `resetCursorRect:inView:` to each cell in the receiver. Any cell that has a cursor rectangle to set up should then send `addCursorRect:cursor:` back to the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `resetCursorRect:inView:` (NSCell)
- `addCursorRect:cursor:` (NSView)

Declared In

`NSMatrix.h`

scrollCellToVisibleAtRow:column:

Scrolls the receiver so the specified cell is visible.

```
- (void)scrollCellToVisibleAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row

The row of the cell to make visible.

column

The column of the cell to make visible.

Discussion

This method scrolls if the receiver is in a scrolling view and *row* and *column* represent a valid cell within the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- scrollRectToVisible: (NSView)

Declared In

NSMatrix.h

selectAll:

Selects and highlights all cells in the receiver.

```
- (void)selectAll:(id)sender
```

Parameters

sender

This argument is ignored.

Discussion

Editable text cells and disabled cells are not selected. The receiver is redisplayed.

If the selection mode is not NSListModeMatrix (page 58), this method does nothing.

Availability

Available in Mac OS X v10.0 and later.

See Also

- selectCell: (NSControl)

Declared In

NSMatrix.h

selectCellAtRow:column:

Selects the cell at the specified row and column within the receiver.

```
- (void)selectCellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row

The row of the cell to select.

column

The column of the cell to select.

Discussion

If the specified cell is an editable text cell, its text is selected. If either *row* or *column* is –1, then the current selection is cleared (unless the receiver is an `NSRadioModeMatrix` and doesn't allow empty selection). This method redraws the affected cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `allowsEmptySelection` (page 16)
- `mode` (page 31)
- `selectCell:` (NSControl)

Declared In

`NSMatrix.h`

selectCellWithTag:

Selects the last cell with the given tag.

```
- (BOOL)selectCellWithTag:(NSInteger)anInt
```

Parameters

anInt

The tag of the cell to select.

Return Value

`YES` if the receiver contains a cell whose tag matches *anInt*, or `NO` if no such cell exists

Discussion

If the matrix has at least one cell whose tag is equal to *anInt*, the last cell (when viewing the matrix as a row-ordered array) is selected. If the specified cell is an editable text cell, its text is selected.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `cellWithTag:` (page 20)
- `selectCell:` (NSControl)

Declared In

`NSMatrix.h`

selectedCell

Returns the most recently selected cell.

```
- (id)selectedCell
```

Return Value
The most recently selected cell or `nil` if no cell is selected. If more than one cell is selected, this method returns the cell that is lowest and farthest to the right in the receiver.

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

Declared In
`NSMatrix.h`

selectedCells

Returns the receiver's selected and highlighted cells.

```
- (NSArray *)selectedCells
```

Return Value
An array containing all of the receiver's highlighted cells plus its selected cell.

Discussion
See the class description for a discussion of the selected cell.

As an alternative to using `setSelectionFrom:to:anchor:highlight:` (page 50) for programmatically making discontiguous selections of cells in a matrix, you could first set the single selected cell and then set subsequent cells to be highlighted; afterwards you can call `selectedCells` (page 39) to obtain the selection of cells.

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

See Also
- `setHighlighted:` (NSCell)
- `selectedCell` (page 38)

Declared In
`NSMatrix.h`

selectedColumn

Returns the column of the selected cell.

```
- (NSInteger)selectedColumn
```

Return Value
The column number of the selected cell or –1 if no cells are selected. If cells in multiple columns are selected, this method returns the number of the last (rightmost) column containing a selected cell.

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

Declared In
`NSMatrix.h`

selectedRow

Returns the row of the selected cell.

```
- (NSInteger)selectedRow
```

Return Value
the row number of the selected cell, or –1 if no cells are selected. If cells in multiple rows are selected, this method returns the number of the last row containing a selected cell.

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

Declared In
NSMatrix.h

selectText:

Selects text in the currently selected cell or in the key cell.

```
- (void)selectText:(id)sender
```

Discussion
If the currently selected cell is editable and enabled, its text is selected. Otherwise, the key cell is selected.

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

See Also
- keyCell (page 30)
- selectText: (NSTextField)

Declared In
NSMatrix.h

selectTextAtRow:column:

Selects the text in the cell at the specified location and returns the cell.

```
- (id)selectTextAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row
The row containing the text to select.

column
The column containing the text to select.

Return Value
If it is both editable and selectable, the cell at the specified row and column. If the cell at the specified location, is either not editable or not selectable, this method does nothing and returns nil. If *row* and *column* indicate a cell that is outside the receiver, this method does nothing and returns the receiver.

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

See Also
- `selectText:` (page 40)

Declared In
`NSMatrix.h`

sendAction

If the selected cell has both an action and a target, sends its action to its target.

```
- (BOOL)sendAction
```

Return Value
`YES` if an action was successfully sent to a target. If the selected cell is disabled, this method does nothing and returns `NO`.

Discussion
If the cell has an action but no target, its action is sent to the target of the receiver. If the cell doesn't have an action, or if there is no selected cell, the receiver sends its own action to its target.

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

See Also
- `sendDoubleAction` (page 42)
- `action` (NSCell)
- `target` (NSCell)

Declared In
`NSMatrix.h`

sendAction:to:forAllCells:

Iterates through the cells in the receiver, sending the specified selector to an object for each cell.

```
- (void)sendAction:(SEL)aSelector to:(id)anObject forAllCells:(BOOL)flag
```

Parameters

aSelector

The selector to send to the object for each cell. This must represent a method that takes a single argument: the `id` of the current cell in the iteration. *aSelector*'s return value must be a `BOOL`. If *aSelector* returns `NO` for any cell, `sendAction:to:forAllCells:` terminates immediately, without sending the message for the remaining cells. If it returns `YES`, `sendAction:to:forAllCells:` proceeds to the next cell.

anObject

The object that is sent the selector for each cell in the matrix.

flag

`YES` if the method should iterate through all cells in the matrix; `NO` if it should iterate through just the selected cells in the matrix.

Discussion
Iteration begins with the cell in the upper-left corner of the receiver, proceeding through the appropriate entries in the first row, then on to the next.

This method is not invoked to send action messages to target objects in response to mouse-down events in the receiver. Instead, you can invoke it if you want to have multiple cells in an `NSMatrix` interact with an object. For example, you could use it to verify the titles in a list of items or to enable a series of radio buttons based on their purpose in relation to *`anObject`*.

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

Declared In
`NSMatrix.h`

sendDoubleAction

Sends the double-click action message to the target of the receiver.

```
- (void)sendDoubleAction
```

Discussion
If the receiver doesn't have a double-click action, the double-click action message of the selected cell (as returned by `selectedCell` (page 38)) is sent to the selected cell's target. Finally, if the selected cell also has no action, then the single-click action of the receiver is sent to the target of the receiver.

If the selected cell is disabled, this method does nothing.

Your code shouldn't invoke this method; it's sent in response to a double-click event in the `NSMatrix`. Override it if you need to change the search order for an action to send.

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

See Also
- `sendAction` (page 41)
- `ignoresMultiClick` (NSControl)

Declared In
`NSMatrix.h`

setAllowsEmptySelection:

Sets whether a radio-mode matrix allows an empty selection.

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

Parameters

`flag`

`YES` to make the receiver allow one or zero cells to be selected. `NO` if the receiver should allow one and only one cell (not zero cells) to be selected. This setting has effect only in the `NSRadioModeMatrix` selection mode.

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

See Also
- `allowsEmptySelection` (page 16)

Declared In
`NSMatrix.h`

setAutoscroll:

Sets whether the receiver is automatically scrolled.

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

Parameters

flag

`YES` to indicate that the receiver, if it is in a scrolling view, should be automatically scrolled whenever the cursor is dragged outside the receiver after a mouse-down event within the bounds of the receiver.

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

Declared In
`NSMatrix.h`

setAutosizesCells:

Sets whether the cell sizes change when the receiver is resized.

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

Parameters

flag

`YES` to specify that, whenever the receiver is resized, the sizes of the cells change in proportion, keeping the intercell space constant; further, this method verifies that the cell sizes and intercell spacing add up to the exact size of the receiver, adjusting the size of the cells and updating the receiver if they don't. If *flag* is `NO`, then the intercell spacing and cell size remain constant.

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

See Also
- `autosizesCells` (page 16)

Declared In
`NSMatrix.h`

setBackgroundColor:

Sets the background color for the receiver and redraws the receiver.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The background color used to fill the space between cells or the space behind any non-opaque cells. The default background color is the color returned by the `NSColor` method `controlColor`.

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

See Also
- `drawsBackground` (page 22)
- `setCellBackgroundColor:` (page 44)
- `backgroundColor` (page 17)

Declared In
`NSMatrix.h`

setCellBackgroundColor:

Sets the background color for the cells in the receiver

```
- (void)setCellBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The background color used to fill the space behind non-opaque cells. The default cell background color is the color returned by the `NSColor` method `controlColor`

Discussion
.

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

See Also
- `drawsCellBackground` (page 23)
- `setBackgroundColor:` (page 43)
- `cellBackgroundColor` (page 18)

Declared In
`NSMatrix.h`

setCellClass:

Configures the receiver to use instances of the specified class when creating new cells.

```
- (void)setCellClass:(Class)aClass
```

Parameters

aClass

The class to use when creating new cells. This should be the `id` of a subclass of `NSCell`, which can be obtained by sending the `class` message to either the `NSCell` subclass object or to an instance of that subclass. The default cell class is that set with the class method `setCellClass:`, or `NSActionCell` if no other default cell class has been specified.

Discussion

You need to use this method only with matrices initialized with `initWithFrame:` (page 25), because the other initializers allow you to specify an instance-specific cell class or cell prototype.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addColumn` (page 14)
- `addRow` (page 15)
- `insertColumn:` (page 27)
- `insertRow:` (page 28)
- `makeCellAtRow:column:` (page 31)
- `setPrototype:` (page 48)
- `cellClass` (page 18)

Declared In

`NSMatrix.h`

setCellSize:

Sets the width and height of each of the cells in the matrix.

```
- (void)setCellSize:(NSSize)aSize
```

Parameters

aSize

The new width and height of cells in the receiver.

Discussion

This method may change the size of the receiver. It does not redraw the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `calcSize` (NSControl)
- `cellSize` (page 19)

Declared In

`NSMatrix.h`

setDelegate:

Sets the delegate for messages from the field editor.

```
- (void)setDelegate:(id)anObject
```

Parameters

anObject

The delegate.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textShouldBeginEditing:` (page 56)
- `textShouldEndEditing:` (page 56)
- `delegate` (page 20)

Declared In

`NSMatrix.h`

setDoubleAction:

Sets the action sent to the target of the receiver when the user double-clicks a cell.

```
- (void)setDoubleAction:(SEL)aSelector
```

Parameters

aSelector

The selector to make the double-click action of the receiver.

Discussion

A double-click action is always sent after the appropriate single-click action, which is the cell's single-click action, if it has one, or the receiver single-click action, otherwise. If *aSelector* is a non-`NULL` selector, this method also sets the *ignoresMultiClick* flag to `NO`; otherwise, it leaves the flag unchanged.

If an `NSMatrix` has no double-click action set, then by default a double click is treated as a single click.

For the method to have any effect, the receiver's action and target must be set to the class in which the selector is declared. See *Action Messages* for additional information on action messages.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `sendDoubleAction` (page 42)
- `setAction:` (NSControl)
- `setTarget:` (NSControl)
- `doubleAction` (page 21)

Declared In

`NSMatrix.h`

setDrawsBackground:

Sets whether the receiver draws its background.

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

Parameters

flag

YES if the receiver should draw its background (the space between the cells); NO if it should not.

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

See Also
- backgroundColor (page 17)
- setDrawsCellBackground: (page 47)
- drawsBackground (page 22)

Declared In
NSMatrix.h

setDrawsCellBackground:

Sets whether the receiver draws the background within each of its cells.

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

Parameters

flag

YES if the receiver should draw the background in its cells; NO if it should not.

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

See Also
- cellBackgroundColor (page 18)
- setDrawsBackground: (page 46)
- drawsCellBackground (page 23)

Declared In
NSMatrix.h

setIntercellSpacing:

Sets the spacing between cells in the matrix.

```
- (void)setIntercellSpacing:(NSSize)aSize
```

Parameters

aSize

The vertical and horizontal spacing to use between cells in the receiver. By default, both values are 1.0 in the receiver's coordinate system.

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

See Also
- `cellSize` (page 19)
- `intercellSpacing` (page 29)

Declared In
`NSMatrix.h`

setKeyCell:

Sets the cell that will be clicked when the user presses the Space bar.

```
- (void)setKeyCell:(NSCell *)aCell
```

Parameters

aCell

The cell to set as the key cell.

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

See Also
- `setTabKeyTraversesCells:` (page 51)
- `keyCell` (page 30)

Declared In
`NSMatrix.h`

setMode:

Sets the selection mode of the receiver.

```
- (void)setMode:(NSMatrixMode)aMode
```

Parameters

aMode

The selection mode of the matrix. Possible values are listed in `NSMatrixMode` (page 57).

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

See Also
- `initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:` (page 25)
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 26)
- `mode` (page 31)

Declared In
`NSMatrix.h`

setPrototype:

Sets the prototype cell that's copied whenever the matrix creates a new cell.

```
- (void)setPrototype:(NSCell *)aCell
```

Parameters

aCell

The cell to copy when creating new cells.

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

See Also
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 26)
- `makeCellAtRow:column:` (page 31)
- `prototype` (page 34)

Declared In
NSMatrix.h

setScrollable:

Specifies whether the cells in the matrix are scrollable.

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

Parameters

flag

`YES` to make all the cells in the receiver scrollable, so the text they contain scrolls to remain in view if the user types past the edge of the cell. If *flag* is `NO`, all cells are made nonscrolling. The prototype cell, if there is one, is also set accordingly

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

See Also
- `prototype` (page 34)
- `setScrollable:` (NSCell)

Declared In
NSMatrix.h

setSelectionByRect:

Sets whether the user can select a rectangle of cells in the receiver by dragging the cursor.

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

Parameters

flag

`YES` if the matrix should allow the user to select a rectangle of cells by dragging. `NO` if selection in the matrix should be on a row-by-row basis. The default is `YES`.

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

See Also
- `setSelectionFrom:to:anchor:highlight:` (page 50)

Declared In
`NSMatrix.h`

setSelectionFrom:to:anchor:highlight:

Programmatically selects a range of cells.

```
- (void)setSelectionFrom:(NSInteger)startPos to:(NSInteger)endPos
    anchor:(NSInteger)anchorPos highlight:(BOOL)lit
```

Parameters

startPos

The position of the cell that marks where the user would have pressed the mouse button.

endPos

The position of the cell that marks where the user would have released the mouse button.

anchorPos

The position of the cell to treat as the last cell the user would have selected. To simulate Shift-dragging (continuous selection) *anchorPos* should be the *endPos* used in the last method call. To simulate Command-dragging (discontinuous selection), *anchorPos* should be the same as this method call's *startPos*.

lit

`YES` if cells selected by this method should be highlighted.

Discussion
startPos, *endPos*, and *anchorPos* are cell positions, counting from 0 at the upper left cell of the receiver, in row order. For example, the third cell in the top row would be number 2.

To simulate dragging without a modifier key, deselecting anything that was selected before, call `deselectAllCells` (page 21) before calling this method.

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

See Also
- `isSelectionByRect` (page 30)
- `selectedCells` (page 39)

Declared In
`NSMatrix.h`

setState:atRow:column:

Sets the state of the cell at specified location.

```
- (void)setState:(NSInteger)value atRow:(NSInteger)row column:(NSInteger)column
```

Parameters

value

The value to assign to the cell.

row

The row in which the cell is located.

column

The column in which the cell is located.

Discussion

For radio-mode matrices, if *value* is nonzero the specified cell is selected before its state is set to *value*. If *value* is 0 and the receiver is a radio-mode matrix, the currently selected cell is deselected (providing that empty selection is allowed).

This method redraws the affected cell.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `allowsEmptySelection` (page 16)
- `setState:` (NSCell)
- `selectCellAtRow:column:` (page 37)

Declared In

`NSMatrix.h`

setTabKeyTraversesCells:

Sets whether pressing the Tab key advances the key cell to the next selectable cell.

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

Parameters

flag

`YES` if pressing the Tab key should advance the key cell to the next selectable cell in the receiver. If this is `NO` or if there aren't any selectable cells after the current one, the next view in the window becomes key when the user presses the Tab key.

Discussion

Pressing Shift-Tab causes the key cell to advance in the opposite direction (if *flag* is `NO`, or if there aren't any selectable cells before the current one, the previous view in the window becomes key).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `selectKeyViewFollowingView:` (NSWindow)
- `selectNextKeyView:` (NSWindow)
- `setKeyCell:` (page 48)
- `tabKeyTraversesCells` (page 54)

Declared In

`NSMatrix.h`

setToolTip:forCell:

Sets the tooltip for the cell.

```
- (void)setToolTip:(NSString *)toolTipString forCell:(NSCell *)cell
```

Parameters

toolTipString

The string to use as the cell's tooltip (or help tag).

cell

The cell to which to assign the tooltip.

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

See Also
- `toolTipForCell:` (page 57)

Declared In
`NSMatrix.h`

setValidateSize:

Specifies whether the receiver's size information is validated.

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

Parameters

flag

`YES` to assume that the size information in the receiver is correct. If *flag* is `NO`, the `NSControl` method `calcSize` will be invoked before any further drawing is done.

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

Declared In
`NSMatrix.h`

sizeToCells

Changes the width and the height of the receiver's frame so it exactly contains the cells.

```
- (void)sizeToCells
```

Discussion
This method does not redraw the receiver.

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

See Also
- `setFrameSize:` (NSView)
- `sizeToFit` (NSControl)

Declared In
NSMatrix.h

sortUsingFunction:context:

Sorts the receiver's cells in ascending order as defined by the specified comparison function.

```
- (void)sortUsingFunction:(int (*)(id, id, void *))comparator context:(void *)context
```

Parameters

comparator

The function to use when comparing cells. The comparison function is used to compare two elements at a time and should return NSOrderedAscending if the first element is smaller than the second, NSOrderedDescending if the first element is larger than the second, and NSOrderedSame if the elements are equal.

context

Context passed to the comparison function as its third argument, each time its called. This allows the comparison to be based on some outside parameter, such as whether character sorting is case-sensitive or case-insensitive.

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

See Also
- sortUsingFunction:context: (NSMutableArray)

Declared In
NSMatrix.h

sortUsingSelector:

Sorts the receiver's cells in ascending order as defined by the comparison method.

```
- (void)sortUsingSelector:(SEL)comparator
```

Parameters

comparator

The comparison method.

Discussion
The comparator message is sent to each object in the matrix and has as its single argument another object in the array. The comparison method is used to compare two elements at a time and should return NSOrderedAscending if the receiver is smaller than the argument, NSOrderedDescending if the receiver is larger than the argument, and NSOrderedSame if they are equal.

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

See Also
- sortUsingSelector: (NSMutableArray)

Declared In
NSMatrix.h

tabKeyTraversesCells

Returns a Boolean value indicating whether pressing the Tab key advances the key cell to the next selectable cell.

```
- (BOOL)tabKeyTraversesCells
```

Return Value
`YES` if pressing the Tab key advances the key cell to the next selectable cell in the receiver; otherwise `NO`.

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

See Also
- `keyCell` (page 30)
- `setTabKeyTraversesCells:` (page 51)

Declared In
`NSMatrix.h`

textDidBeginEditing:

Invoked when there's a change in the text after the receiver gains first responder status.

```
- (void)textDidBeginEditing:(NSNotification *)notification
```

Parameters
notification
The `NSControlTextDidBeginEditingNotification` notification.

Discussion
This method's default behavior is to post an `NSControlTextDidBeginEditingNotification` along with the receiving object to the default notification center. The posted notification's user info contains the contents of notification's user info dictionary, plus an additional key-value pair. The additional key is "`NSFieldEditor`"; the value for this key is the text object that began editing.

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

See Also
- `textDidChange:` (page 54)
- `textDidEndEditing:` (page 55)
- `textShouldEndEditing:` (page 56)

Declared In
`NSMatrix.h`

textDidChange:

Invoked when a key-down event or paste operation occurs that changes the receiver's contents.

```
- (void)textDidChange:(NSNotification *)notification
```

Parameters

notification

The `NSControlTextDidChangeNotification` notification.

Discussion

This method's default behavior is to pass this message on to the selected cell (if the selected cell responds to `textDidChange:`) and then to post an `NSControlTextDidChangeNotification` along with the receiving object to the default notification center. The posted notification's user info contains the contents of notification's user info dictionary, plus an additional key-value pair. The additional key is "`NSFieldEditor`"; the value for this key is the text object that changed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textDidBeginEditing:` (page 54)
- `textDidEndEditing:` (page 55)

Declared In

`NSMatrix.h`

textDidEndEditing:

Invoked when text editing ends.

```
- (void)textDidEndEditing:(NSNotification *)notification
```

Parameters

notification

The `NSControlTextDidEndEditingNotification` notification.

Discussion

This method's default behavior is to post an `NSControlTextDidEndEditingNotification` along with the receiving object to the default notification center. The posted notification's user info contains the contents of notification's user info dictionary, plus an additional key-value pair. The additional key is "`NSFieldEditor`"; the value for this key is the text object that began editing. After posting the notification, `textDidEndEditing:` sends an `endEditing:` message to the selected cell, draws and makes the selected cell key, and then takes the appropriate action based on which key was used to end editing (Return, Tab, or Back-Tab).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textDidBeginEditing:` (page 54)
- `textDidChange:` (page 54)
- `textShouldEndEditing:` (page 56)

Declared In

`NSMatrix.h`

textShouldBeginEditing:

Requests permission to begin editing text.

```
- (BOOL)textShouldBeginEditing:(NSText *)textObject
```

Parameters

textObject

The text object requesting permission to begin editing.

Return Value

YES if the text object should proceed to make changes. If the delegate returns NO, the text object abandons the editing operation.

The default behavior of this method is to return the value obtained from control:textShouldBeginEditing:, unless the delegate doesn't respond to that method, in which case it returns YES, thereby allowing text editing to proceed.

Discussion

This method is invoked to let the NSTextField respond to impending changes to its text. This method's default behavior is to send control:textShouldBeginEditing: to the receiver's delegate (passing the receiver and *textObject* as parameters).

Availability

Available in Mac OS X v10.0 and later.

See Also

- delegate (page 20)

Declared In

NSMatrix.h

textShouldEndEditing:

Requests permission to end editing.

```
- (BOOL)textShouldEndEditing:(NSText *)textObject
```

Parameters

textObject

The text object requesting permission to end editing.

Return Value

YES if the text object should proceed to finish editing and resign first responder status. If the delegate returns NO, the text object selects all of its text and remains the first responder.

The textShouldEndEditing: method returns NO if the text field contains invalid contents; otherwise it returns the value passed back from control:textShouldEndEditing:.

Discussion

This method is invoked to let the NSTextField respond to impending loss of first-responder status. This method's default behavior checks the text field for validity; providing that the field contents are deemed valid, and providing that the delegate responds, control:textShouldEndEditing: is sent to the receiver's delegate (passing the receiver and *textObject* as parameters).

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

See Also
- `delegate` (page 20)

Declared In
`NSMatrix.h`

toolTipForCell:

Returns the tooltip for the specified cell.

```
- (NSString *)toolTipForCell:(NSCell *)cell
```

Parameters

cell
The cell for which to return the tooltip.

Return Value
The string used as the cell's tooltip.

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

See Also
- `setToolTip:forCell:` (page 52)

Declared In
`NSMatrix.h`

Constants

NSMatrixMode

These constants determine how `NSCell` objects behave when an `NSMatrix` object is tracking the mouse.

```
typedef enum _NSMatrixMode {
    NSRadioModeMatrix     = 0,
    NSHighlightModeMatrix = 1,
    NSListModeMatrix      = 2,
    NSTrackModeMatrix     = 3
} NSMatrixMode;
```

Constants

`NSTrackModeMatrix`
The `NSCell` objects are asked to track the mouse with `trackMouse:inRect:ofView:untilMouseUp:` whenever the cursor is inside their bounds. No highlighting is performed.

Available in Mac OS X v10.0 and later.

Declared in `NSMatrix.h`.

`NSHighlightModeMatrix`
: An `NSCell` is highlighted before it's asked to track the mouse, then unhighlighted when it's done tracking.

  Available in Mac OS X v10.0 and later.

  Declared in `NSMatrix.h`.

`NSRadioModeMatrix`
: Selects no more than one `NSCell` at a time.

  Any time an `NSCell` is selected, the previously selected `NSCell` is unselected.

  Available in Mac OS X v10.0 and later.

  Declared in `NSMatrix.h`.

`NSListModeMatrix`
: `NSCell` objects are highlighted, but don't track the mouse.

  Available in Mac OS X v10.0 and later.

  Declared in `NSMatrix.h`.

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

Declared In
`NSMatrix.h`

Document Revision History

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

Date	Notes
2008-10-15	Clarified usage of selectAll:.
2007-03-02	Fixed minor bugs and revised task headings.
	Removed mention of error action message from `textShouldEndEditing`. Corrected typo in -cellSize description.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index

N

P

R

S

T