NSMatrix

Inherits From: NSControl : NSView : NSResponder : NSObject

Declared In: appkit/NSMatrix.h

Initializing the NSMatrix Class

+ (Class)**cellClass**	Returns the default class used to make cells.
+ (void)**setCellClass:**(Class)*classId*	Sets the default class used to make cells.

Initializing an NSMatrix Object

- (id)**initWithFrame:**(NSRect)*frameRect*	Initializes a new NSMatrix object in *frameRect*.
- (id)**initWithFrame:**(NSRect)*frameRect* **mode:**(int)*aMode* **cellClass:**(Class)*classId* **numberOfRows:**(int)*numRows* **numberOfColumns:**(int)*numCols*	Initializes a new NSMatrix object in *frameRect*, with *aMode* as the selection mode, *classId* as the class used to make new cells, and having *numRows* rows and *numCols* columns.
- (id)**initWithFrame:**(NSRect)*frameRect* **mode:**(int)*aMode* **prototype:**(NSCell *)*aCell* **numberOfRows:**(int)*numRows* **numberOfColumns:**(int)*numCols*	Initializes a new NSMatrix object with the given values with *aMode* as the selection mode, *aCell* as the prototype copied to make new cells, and having *numRows* rows and *numCols* columns.

Setting the Selection Mode

- (void)**setMode:**(NSMatrixMode)*aMode*	Sets the selection mode of the matrix.

- (NSMatrixMode)**mode** — Returns the selection mode of the matrix.

Configuring the NSMatrix

- (void)**setAllowsEmptySelection:**(BOOL)*flag* — Sets whether there may be no cells selected.
- (BOOL)**allowsEmptySelection** — Returns whether there may be no cells selected.
- (void)**setSelectionByRect:**(BOOL)*flag* — Sets whether a user can drag a rectangular selection (the default is YES).
- (BOOL)**isSelectionByRect** — Returns whether a user can drag a rectangular selection.

Setting the Cell Class

- (Class)**cellClass** — Returns the subclass of NSCell used to make new cells.
- (void)**setCellClass:**(Class)*classId* — Sets the subclass of NSCell used to make new cells.
- (void)**setPrototype:**(NSCell *)*aCell* — Sets the prototype cell copied to make new cells.
- (id)**prototype** — Returns the prototype cell copied to make new cells.

Laying Out the NSMatrix

- (void)**addColumn** — Adds a new column of cells to the right of the last column.
- (void)**addRow** — Adds a new row of cells below the last row.
- (void)**addRowWithCells:**(NSArray *)*cellArray* — Adds a new row of cells, using those contained in *cellArray.*
- (void)**addColumnWithCells:**(NSArray *)*cellArray* — Adds a new column of cells, using those contained in *cellArray.*
- (void)**insertColumn:**(int)*column* — Inserts a new column of cells at *column*, creating as many as needed to make the matrix *column* columns wide.
- (void)**insertColumn:**(int)*column* **withCells:**(NSArray *)*cellArray* — Inserts a new row of cells at *column*, using those contained in *cellArray.*
- (void)**insertRow:**(int)*row* — Inserts a new row of cells at *row*, creating as many as needed to make the matrix *row* rows wide.
- (void)**insertRow:**(int)*row* **withCells:**(NSArray *)*cellArray* — Inserts a new row of cells at *row*, using those contained in *cellArray.*
- (void)**removeColumn:**(int)*column* **andRelease:**(BOOL)*flag* — Removes the column at *column*, releasing the cells if *flag* is YES.
- (void)**removeRow:**(int)*row* **andRelease:**(BOOL)*flag* — Removes the row at *row*, releasing the cells if *flag* is YES.

- (NSCell *)**makeCellAtRow:**(int)*row*
  column:(int)*column*
  Creates a new cell at *row*, *column* in the matrix and returns it.

- (void)**putCell:**(NSCell *)*newCell*
  atRow:(int)*row*
  column:(int)*column*
  Replaces the cell at *row* and *column* with *newCell*.

- (void)**renewRows:**(int)*newRows*
  columns:(int)*newColumns*
  Changes the number of rows and columns in the receiver without freeing any cells.

- (void)**setCellSize:**(NSSize)*aSize*
  Sets the width and height of all cells in the matrix.

- (NSSize)**cellSize**
  Returns the width and height of cells in the matrix.

- (NSRect)**cellFrameAtRow:**(int)*row*
  column:(int)*column*
  Returns the frame rectangle of the cell at *row* and *column*.

- (void)**setIntercellSpacing:**(NSSize)*aSize*
  Sets the vertical and horizontal spacing between cells.

- (NSSize)**intercellSpacing**
  Returns the vertical and horizontal spacing between cells

- (void)**getNumberOfRows:**(int *)*rowCount*
  columns:(int *)*columnCount*
  Gets the number of rows and columns in the matrix.

- (void)**sortUsingFunction:**(int (*)(id *element1*,id *element2*,void **userData*))*comparator*
  context:(void *)*context*
  Sorts the receiver's cells in ascending order as defined by the comparison function *comparator*. *context* is passed as the function's third argument.

- (void)**sortUsingSelector:**(SEL)*comparator*
  Sorts the receiver's cells in ascending order as defined by the comparison method *comparator*.

Finding Matrix Coordinates

- (NSCell *)**getRow:**(int *)*row*
  column:(int *)*column*
  ofCell:(NSCell *)*aCell*
  Gets the row and column position of *aCell*.

- (NSCell *)**getRow:**(int *)*row*
  column:(int *)*column*
  forPoint:(NSPoint)*aPoint*
  Gets the row and column position corresponding to *aPoint*, and returns the cell at that point.

Modifying Individual Cells

- (void)**setImage:**(NSImage *)*iconName*
  atRow:(int)*row*
  column:(int)*column*
  Sets the image for the cell at *row* and *column* to the NSImage named *iconName*.

- (void)**setState:**(int)*value*
  atRow:(int)*row*
  column:(int)*column*
  Sets the state of the cell at *row* and *column* to *value*.
- (void)**setTitle:**(NSString *)*aString*
  atRow:(int)*row*
  column:(int)*column*
  Assigns cell at *row* and *column* the title in *aString*.

Selecting Cells

- (void)**deselectAllCells**
  Clears the receiver's selection, assuming that the NSMatrix allows an empty selection.
- (void)**deselectSelectedCell**
  Deselects the selected cell.
- (void)**selectCellAtRow:**(int)*row*
  column:(int)*column*
  Selects the cell at *row* and *col*.
- (BOOL)**selectCellWithTag:**(int)*anInt*
  Selects the cell with the tag *anInt*.
- (void)**setSelectionFrom:**(int)*startPos*
  to:(int)*endPos*
  anchor:(int)*anchorPos*
  highlight:(BOOL)*flag*
  Selects the cells in the matrix from *startPos* to *endPos*, counting in row order from the upper left, as though *anchorPos* were the number of the last cell selected, and highlighting the cells according to *flag*.
- (void)**selectAll:**(id)*sender*
  Selects all the cells in the matrix.
- (id)**selectedCell**
  Returns the last (lowest and rightmost) selected cell.
- (NSArray *)**selectedCells**
  Returns an array containing the selected cells.
- (int)**selectedColumn**
  Returns the column of the selected cell.
- (int)**selectedRow**
  Returns the row of the selected cell.

Finding Cells

- (id)**cellWithTag:**(int)*anInt*
  Returns the cell having *anInt* as its tag.
- (id)**cellAtRow:**(int)*row*
  column:(int)*column*
  Returns the cell at row *row* and column *col*.
- (NSArray *)**cells**
  Returns the matrix's array of cells.

Modifying Graphic Attributes

- (void)**setBackgroundColor:**(NSColor *)*aColor*
  Sets the color of the background between cells to *aColor*.

- (NSColor *)**backgroundColor** Returns the color of the background between cells.
- (void)**setCellBackgroundColor:**(NSColor *)*aColor*
  Sets the color of the background within cells to *aColor*.
- (NSColor *)**cellBackgroundColor** Returns the color of the background within cells.
- (void)**setDrawsBackground:**(BOOL)*flag* Sets whether the receiver draws the background between cells.
- (BOOL)**drawsBackground** Returns whether the receiver draws the background between cells.
- (void)**setDrawsCellBackground:**(BOOL)*flag* Sets whether the receiver draws the background within cells.
- (BOOL)**drawsCellBackground** Returns whether the receiver draws the background within cells.

Editing Text in Cells

- (void)**selectText:**(id)*sender* Selects the text in the first or last editable cell.
- (id)**selectTextAtRow:**(int)*row* Selects the text of the cell at *row*, *column* in the matrix.
  column:(int)*column*
- (void)**textDidBeginEditing:**(NSNotification *)*notification*
  Invoked when there's a change in the text after the receiver gains first responder status. Default behavior is pass to this message on to the text delegate.
- (void)**textDidChange:**(NSNotification *)*notification*
  Invoked upon a key-down event or paste operation that changes the receiver's contents. Default behavior is to pass this message on to the text delegate.
- (void)**textDidEndEditing:**(NSNotification *)*notification*
  Invoked when text editing ends and then forwarded to the text delegate.
- (BOOL)**textShouldBeginEditing:**(NSText *)*textObject*
  Invoked to let the NSTextField respond to impending changes to its text and then forwarded to the text delegate.
- (BOOL)**textShouldEndEditing:**(NSText *)*textObject*
  Invoked to let the NSTextField respond to impending loss of first responder status and then forwarded to the text delegate.

Setting Tab Key Behavior

- (id)**nextText** Returns the object selected when the user presses Tab while editing the last text cell.

- (id)**previousText** — Returns the object selected when the user presses Tab while editing the first text cell.
- (void)**setNextText:**(id)*anObject* — Sets the object selected when the user presses Tab while editing the last text cell.
- (void)**setPreviousText:**(id)*anObject* — Sets the object selected when user presses Shift-Tab while editing the first text cell.

Assigning a Text Delegate

- (void)**setTextDelegate:**(id)*anObject* — Sets the delegate for messages from the field editor.
- (id)**textDelegate** — Returns the delegate for messages from the field editor.

Resizing the Matrix and Cells

- (void)**setAutosizesCells:**(BOOL)*flag* — Sets whether the matrix resizes its cells automatically.
- (BOOL)**autosizesCells** — Returns whether the matrix resizes its cells automatically.
- (void)**sizeToCells** — Resizes the matrix to fit its cells exactly.
- (void)**setValidateSize:**(BOOL)*flag* — Sets whether the cell size needs to be recalculated.

Scrolling

- (void)**setAutoscroll:**(BOOL)*flag* — Sets whether the matrix automatically scrolls when dragged in.
- (BOOL)**isAutoscroll** — Returns whether the matrix automatically scrolls when dragged in.
- (void)**setScrollable:**(BOOL)*flag* — If *flag* is YES, makes all the cells scrollable.
- (void)**scrollCellToVisibleAtRow:**(int)*row* **column:**(int)*column* — Scrolls the matrix so that the cell at *row* and *column* is visible.

Displaying

- (void)**drawCellAtRow:**(int)*row* **column:**(int)*column* — Displays the cell at *row* and *col*.
- (void)**highlightCell:**(BOOL)*flag* **atRow:**(int)*row* **column:**(int)*column* — Highlights (or unhighlights) the cell at *row*, *col*.

Target and Action

- (void)**setDoubleAction:**(SEL)*aSelector* Sets the action method used on double-clicks to *aSelector*.
- (SEL)**doubleAction** Returns the action method for double clicks.
- (void)**setErrorAction:**(SEL)*aSelector* Sets the action method for editing errors to *aSelector*.
- (SEL)**errorAction** Returns the action method for editing errors.
- (BOOL)**sendAction** Sends the selected cell's action, or the NSMatrix's action if the cell doesn't have one.
- (void)**sendAction:**(SEL)*aSelector* Sends *aSelector* to *anObject*, for all cells if *flag* is YES.
  to:(id)*anObject*
  forAllCells:(BOOL)*flag*
- (void)**sendDoubleAction** Sends the action corresponding to a double-click.

Handling Event and Action Messages

- (BOOL)**acceptsFirstMouse:**(NSEvent *)*theEvent* Returns NO only if receiver's mode is NSListModeMatrix.
- (void)**mouseDown:**(NSEvent *)*theEvent* Responds to a mouse-down event.
- (int)**mouseDownFlags** Returns the event flags in effect at start of tracking.
- (BOOL)**performKeyEquivalent:**(NSEvent *)*theEvent*
  Simulates a mouse click in the appropriate cell.

Managing the Cursor

- (void)**resetCursorRects** Resets cursor rectangles so that the cursor becomes an I-beam over text cells.