# NSPopUpButton

**Inherits From:** NSButton : NSControl : NSView : NSResponder : NSObject

**Conforms To:** NSCoding (NSResponder)
NSObject (NSObject)

**Declared In:** AppKit/NSPopUpButton.h

## Class at a Glance

**Purpose**
An NSPopUpButton object controls a pop-up list or a pull-down list, from which a user can select an item.

**Principal Attribute**
A list of objects that conform to the NSMenuItem protocol.

**Creation**
Interface Builder

**Commonly Used Methods**

| | |
|---|---|
| – selectedItem | Returns the currently selected item. |
| – indexOfSelectedItem | Returns an integer identifying the currently selected item. |
| – titleOfSelectedItem | Returns a string identifying the currently selected item. |

## Class Description

The NSPopUpButton class defines objects that implement the pop-up and pull-down lists of the OpenStep graphical user interface. You normally create an NSPopUpButton using Interface Builder.

When configured to display a pop-up list, an NSPopUpButton contains a number of options and displays as its title the option that was last selected. A pop-up list is often used for selecting items from a small- to medium-sized set of options (like the zoom factor for a document window). It's a useful alternative to a matrix of radio buttons or an NSBrowser when screen space is at a premium; a zoom factor pop-up can easily fit next to a scroll bar at the bottom of a window, for example.

When configured to display a pull-down list, an NSPopUpButton is generally used for selecting commands in a very specific context. You can think of a pull-down list as a compact form of menu. A pull-down list's title isn't affected by the user's actions, and a pull-down list always displays a title that identifies the type of commands it contains. When the commands only make sense in the context of a particular display, a pull-down list can be used in that display to keep the related actions nearby and to keep them out of the way when that display isn't visible. For example, Interface Builder has a set of commands that only apply to the Classes view of the nib file window. Those commands are contained in a pull-down list in that view.

The items in the pop-up list or pull-down list (referred to simply as a list in this class description) are objects that conform to the NSMenuItem protocol. Thus, you can send any message defined in that protocol to any item in the list.

## Using an NSPopUpButton

Although the NSPopUpButton class defines an initialization method and methods that set up the list created by this class, you usually don't invoke these methods in your program. The typical way to create an NSPopUpButton is to use Interface Builder. You also define the NSPopUpButton's target and action, as well as the targets and actions of each item in the NSPopUpButton's list, using Interface Builder. The NSPopUpButton methods you use most often are the methods that tell you which item is selected.

For example, suppose you want to create a pop-up list from which your user may select a language. Your want your controller object to set an instance variable named **language** to an enum constant that corresponds to the value that the user has chosen. You use Interface Builder to create the NSPopUpButton object, name it (**languagePopUp** in this example) add items to it, and configure it as a pop-up list. The actual code you write might look like this:

```
typedef enum _languageValue {
    English,
    French,
    German
} languageValue;

- (void)setLanguage:(id)sender
{
    NSString *title = [languagePopUp titleOfSelectedItem];

    if ([title isEqualToString:@"English"])
        language = English;
    else if ([title isEqualToString:@"French"])
        language = French;
    else if ([title isEqualToString:@"German"])
        language = German;
}
```

## Method Types

| | |
|---|---|
| Initializing an NSPopUpButton | – initWithFrame:pullsDown: |
| Setting the Type of List | – setPullsDown:<br>– pullsDown<br>– setAutoenablesItems:<br>– autoenablesItems |
| Inserting and Deleting Items | – addItemWithTitle:<br>– addItemsWithTitles:<br>– insertItemWithTitle:atIndex:<br>– removeAllItems<br>– removeItemWithTitle:<br>– removeItemAtIndex: |
| Modifying an Item | – setFont:<br>– font |
| Getting the User’s Selection | – selectedItem<br>– titleOfSelectedItem<br>– indexOfSelectedItem<br>– stringValue |
| Setting the Current Selection | – selectItemAtIndex:<br>– selectItemWithTitle:<br>– setTitle: |
| Querying the NSPopUpButton | – indexOfItemWithTitle:<br>– numberOfItems<br>– itemArray<br>– itemAtIndex:<br>– itemTitleAtIndex:<br>– itemTitles<br>– itemWithTitle:<br>– lastItem |
| Target and Action | – setAction<br>– action<br>– setTarget:<br>– target |
| Setting the State | – synchronizeTitleAndSelectedItem |

## Instance Methods

### action

– (SEL)**action**

Returns the action to be sent to the NSPopUpButton's target when an item is selected from the list.

**See also:** – **setAction:**, – **target**

### addItemWithTitle:

– (void)**addItemWithTitle:**(NSString *)*title*

Adds an item named *title* to the end of the list. This method then calls **synchronizeTitleAndSelectedItem** to make sure that the title displayed matches the currently selected item.

**See also:** – **insertItemWithTitle:atIndex:**, – **removeItemWithTitle:**, – **setTitle:**

### addItemsWithTitles:

– (void)**addItemsWithTitles:**(NSArray *)*itemTitles*

Adds multiple items to the end of the list. The titles for the new items are taken from the *itemTitles* array. Once the items are added, this method uses **synchronizeTitleAndSelectedItem** to make sure that the title displayed matches the currently selected item.

**See also:** – **insertItemWithTitle:atIndex:**, – **removeAllItems**, – **removeItemWithTitle:**

### autoenablesItems

– (BOOL)**autoenablesItems**

Returns whether the NSPopUpButton automatically enables and disables its items every time a user event occurs. Autoenabling is turned on unless you send the message **setAutoenablesItems:NO** to the NSPopUpButton. See the NSMenuActionResponder informal protocol for more information on autoenabling menu items.

**See also:** – **setAutoenablesItems:**

### font

– (NSFont *)**font**

Returns the NSFont used for the items' titles.

**See also:** – **setFont:**

### indexOfItemWithTitle:

– (int)**indexOfItemWithTitle:**(NSString *)*title*

Returns the index of the item whose title matches *title* or –1 if no match is found.

### indexOfSelectedItem

– (int)**indexOfSelectedItem**

Returns the index of the item last selected by the user or –1 if there's no selected item.

**See also:** – **selectedItem**, – **titleOfSelectedItem**

### initWithFrame:pullsDown:

– (id)**initWithFrame:**(NSRect)*frameRect* **pullsDown:**(BOOL)*flag*

Initializes a newly allocated NSPopUpButton, giving it the dimensions specified by *frameRect.* If *flag* is YES, the receiver is initialized to operate as a pull-down list; otherwise, it operates as a pop-up list. If you allocate and initialize an NSPopUpButton in code, you are responsible for releasing it.

**See also:** – **pullsDown**, – **setPullsDown:**

### insertItemWithTitle:atIndex:

– (void)**insertItemWithTitle:**(NSString *)*title* **atIndex:**(int)*index*

Inserts an item with the name *title* at position *index* in the list. Index 0 indicates the top item. Once the item is inserted, this method uses **synchronizeTitleAndSelectedItem** to make sure that the title displayed matches the currently selected item.

If an item with the name *title* already exists in the list, it's removed and the new one is added. This essentially moves *title* to a new position. If you want to move an item, it's better to invoke **removeItemWithTitle:** explicitly and then send this method.

**See also:** **– addItemWithTitle:**, **– addItemsWithTitles:**, **– indexOfItemWithTitle:**, **– removeItemWithTitle:**

### itemArray

– (NSArray *)**itemArray**

Returns the NSArray that holds the list's items. The NSPopUpButton's list is actually an NSArray of objects conforming to the NSMenuItem protocol. Usually you access the list's items and modify the list by sending messages directly to the NSPopUpButton rather than accessing the NSArray.

**See also:** – **itemAtIndex:**, – **insertItemWithTitle:atIndex:**, – **removeItemAtIndex:**

### itemAtIndex:

– (id <NSMenuItem>)**itemAtIndex:**(int)*index*

Returns the list item at *index*. If there is no item at *index*, this method returns **nil**.

**See also:** – **itemWithTitle:**, – **lastItem**

### itemTitleAtIndex:

– (NSString *)**itemTitleAtIndex:**(int)*index*

Returns the title of the item at *index*. If there is no item at *index*, this method returns the empty string.

**See also:** – **itemTitles**

### itemTitles

– (NSArray *)**itemTitles**

Returns an NSArray object that holds the titles of all of the items in the list. The titles appear in the order in which the items appear in the list.

**See also:** – **itemTitleAtIndex:**, – **itemWithTitle:**, – **numberOfItems**

### itemWithTitle:

– (id <NSMenuItem>)**itemWithTitle:**(NSString *)*title*

Returns the item whose title is *title*. If there is no item with this title, this method returns **nil**.

**See also:** – **addItemWithTitle:**, – **selectItemWithTitle:**, – **itemAtIndex:**, – **indexOfItemWithTitle:**

### lastItem

– (id <NSMenuItem>)**lastItem**

Returns the last item in the list.

**See also:** – **itemAtIndex:**

### numberOfItems

– (int)**numberOfItems**

Returns the number of items in the list.

**See also:** – **lastItem**

### pullsDown

– (BOOL)**pullsDown**

Returns YES if the receiver is configured as a pull-down list or NO if it's configured as a pop-up list.

**See also:** – **setPullsDown:**

### removeAllItems

– (void)**removeAllItems**

Removes all items in the receiver's item list. This method then uses **synchronizeTitleAndSelectedItem** to refresh the list.

**See also:** – **numberOfItems**, – **removeItemAtIndex:**, – **removeItemWithTitle:**

### removeItemAtIndex:

– (void)**removeItemAtIndex:**(int)*index*

Removes the item at *index*. This method then uses **synchronizeTitlesAndSelectedItem** to make sure the title displayed matches the currently selected item.

**See also:** – **insertItemWithTitle:atIndex:**, – **removeAllItems**, – **removeItemWithTitle:**

### removeItemWithTitle:

– (void)**removeItemWithTitle:**(NSString *)*title*

Removes the item named *title*. This method then uses **synchronizeTitleAndSelectedItem** to refresh the list.

**See also:** – **addItemWithTitle:**, – **removeAllItems**, – **removeItemAtIndex:**

### selectedItem

– (id <NSMenuItem>)**selectedItem**

Returns the item last selected by the user (the item that was highlighted when the user released the mouse button). If there is no selected item, this method returns **nil**. It is possible for a pop-up list's selected item to be its title item.

**See also:** – **stringValue**

### selectItemAtIndex:

– (void)**selectItemAtIndex:**(int)*index*

Selects the item at *index* and invokes **synchronizeTitleAndSelectedItem** to make sure the title displayed matches the currently selected item.

**See also:** – **indexOfSelectedItem**

### selectItemWithTitle:

– (void)**selectItemWithTitle:**(NSString *)*title*

Selects the item named *title* and invokes **synchronizeTitleAndSelectedItem** to make sure the title displayed matches the currently selected item.

**See also:** – **indexOfItemWithTitle:**, – **addItemWithTitle:**, – **setTitle:**

### setAction:

– (void)**setAction:**(SEL)*aSelector*

Sets the NSPopUpButton's action method to *aSelector*. The action message is sent to the NSPopUpButton's target when an item is selected from the list.

**See also:** – **action**, – **setTarget:**

### setAutoenablesItems:

– (void)**setAutoenablesItems:**(BOOL)*flag*

Sets whether the NSPopUpButton automatically enables and disables its items every time a user event occurs. Autoenabling is turned on unless you specify NO as the value for *flag*. See the NSMenuActionResponder informal protocol for more information on autoenabling menu items.

**See also:** – **autoenablesItems**

### setFont:

– (void)**setFont:**(NSFont *)*fontObject*

Sets the font used for the items' titles to *fontObject*. The NSPopUpButton redraws itself at this point, but since it normally won't be on the screen when it receives this message, this shouldn't cause any undesirable side-effects.

**See also:** – **font**

### setPullsDown:

– (void)**setPullsDown:**(BOOL)*flag*

If *flag* is YES, the receiver is configured as a pull-down list. If *flag* is NO, the receiver is configured as a pop-up list.

**See also:** – **initWithFrame:pullsDown:**, – **pullsDown**

### setTarget:

– (void)**setTarget:**(id)*anObject*

Sets the target for action messages to *anObject*.

**See also:** – **target**, – **setAction:**

### setTitle:

– (void)**setTitle:**(NSString *)*title*

Selects the item named *title* if such an item exists. If the item does not exist, it first adds the item to the end of the list, then selects it and invokes the method **synchronizeTitleAndSelectedItem**.

**See also:** – **addItemWithTitle:**, – **selectItemWithTitle:**

### stringValue

– (NSString *)**stringValue**

Returns the title of the selected item by invoking **titleOfSelectedItem**.

**See also:** – **selectedItem**

### synchronizeTitleAndSelectedItem

– (void)**synchronizeTitleAndSelectedItem**

Ensures that the receiver's title agrees with the title of the selected item (see **indexOfSelectedItem**). If there's no selected item, this method selects the first item in the item list and sets the receiver's title to match. This method is useful in subclasses that directly select items in the item matrix or that override **setTitle:**. This method has no effect on pull-down lists.

**See also:** – **itemArray**

### target

– (id)**target**

Returns the target for action messages.

**See also:** – **action**

### titleOfSelectedItem

– (NSString *)**titleOfSelectedItem**

Returns the title of the item last selected by the user or the empty string if there's no such item.

**See also:** – **indexOfSelectedItem**, – **stringValue**

## Notifications

### NSPopUpButtonWillPopUpNotification

Posted when the NSPopUpButton receives a mouse-down event; that is, when the user is about to select an item from the list. The notification contains:

| | |
|---|---|
| **Notification Object** | the selected NSPopUpButton |
| **Userinfo** | None |