NSPanel

Inherits From:	NSWindow : NSResponder : NSObject
Conforms To:	NSCoding (NSResponder) NSObject (NSObject)
Declared In:	AppKit/NSPanel.h

Class Description

A panel is a special kind of window, typically serving an auxiliary function in an application. NSPanel adds a few special behaviors to NSWindow in support of the role panels play:

- Panels are by default not released when they're closed, since they're usually lightweight and often reused.
- On-screen panels, except for attention panels, are removed from the screen when the application isn't active, and are restored when the application again becomes active. This reduces screen clutter.
- Panels can become the key window, but not the main window.
- If a panel is the key window and has a close button, it closes itself when the user presses the Escape key.

In addition to these automatic behaviors, NSPanel allows you to configure certain other behaviors common to some kinds of panels:

- A panel can be precluded from becoming the key window unless the user clicks in a view that responds to typing. This prevents key window from shifting to the panel unnecessarily. The **setBecomesKeyOnlyIfNeeded:** method controls this behavior.
- Palettes and similar panels can be made to float above standard windows and other panels. This prevents them from being covered and keeps them readily available to the user. The **setFloatingPanel:** method controls this behavior.
- A panel can be made to receive mouse and keyboard events even when another window or panel is being run modally or run in a modal session. This permits actions in the panel to affect the modal window or panel. The **setWorksWhenModal:** method controls this behavior. See "Modal Windows" in the NSWindow class specification for more information on modal windows and panels.

Method Types

Configuring panel behavior	– setFloatingPanel:
	– isFloatingPanel
	– setBecomesKeyOnlyIfNeeded:
	– becomesKeyOnlyIfNeeded
	– setWorksWhenModal:
	– worksWhenModal

Instance Methods

becomesKeyOnlyIfNeeded

– (BOOL)**becomesKeyOnlyIfNeeded**

Returns YES if the receiver becomes the key window only when the user clicks a view object that needs to be first responder to receive event and action messages; for example if it edits text or otherwise accepts keyboard input. Returns NO if it becomes the key window whenever clicked. NSPanel by default returns NO, indicating that panels become key as other windows do.

See also: – **setBecomesKeyOnlyIfNeeded:**, – **needsPanelToBecomeKey** (NSView)

isFloatingPanel

– (BOOL)**isFloatingPanel**

Returns YES if the receiver is set to float above normal windows, NO otherwise. A floating panel's window level is NSFloatingWindowLevel. NSPanels by default returns NO, indicating that they inhabit the normal window level.

See also: – **setFloatingPanel:**, – **level** (NSWindow)

setBecomesKeyOnlyIfNeeded:

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

Controls whether the receiver only becomes the key window when the user clicks a view object that edits text or otherwise accepts keyboard input. If *flag* is YES, the receiver only becomes the key window when keyboard input is needed; if *flag* is NO, it becomes the key window whenever clicked. This behavior is by default not set. You should consider setting it only if most controls in the NSPanel aren't text fields, and if the choices that can be made by entering text can also be made in another way (such as by clicking an item in a pick list).

See also: – **becomesKeyOnlyIfNeeded**, – **needsPanelToBecomeKey** (NSView)

setFloatingPanel:

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

Controls whether the receiver floats above normal windows. If *flag* is YES, sets the receiver’s window level to NSFloatingWindowLevel; if *flag* is NO, sets the receiver’s window level to NSNormalWindowLevel. The default is NO. It’s appropriate for an NSPanel to float above other windows only if all of the following conditions are true:

- It’s small enough not to obscure whatever’s behind it.
- It’s oriented more to the mouse than to the keyboard—that is, if it doesn’t become the key window or becomes so only when needed.
- It needs to remain visible while the user works in the application’s normal windows; for example, if the user must frequently move the cursor back and forth between a normal window and the panel (such as a tool palette), or if the panel gives information relevant to the user’s actions in a normal window.
- It hides when the application is deactivated (the default behavior for panels).

See also: – **isFloatingPanel**, – **setWindowLevel:** (NSWindow)

setWorksWhenModal:

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

Controls whether the receiver receives keyboard and mouse events even when some other window is being run modally. If *flag* is YES, the application object sends events to the receiver even during a modal loop or session; if *flag* is NO, the receiver gets no events while a modal loop or session is running. See “Modal Windows” in the NSWindow class specification for more information on modal windows and panels.

See also: – **worksWhenModal**, – **runModalForWindow:** (NSApplication),
– **runModalSession:** (NSApplication)

worksWhenModal

– (BOOL)**worksWhenModal**

Returns YES if the receiver is able to receive keyboard and mouse events even when some other window is being run modally, NO otherwise. NSPanels by default returns NO, indicating their ineligibility for events during a modal loop or session. See “Modal Windows” in the NSWindow class specification for more information on modal windows and panels.

See also: – **setWorksWhenModal:**, – **runModalForWindow:** (NSApplication),
– **runModalSession:** (NSApplication)