NS_DEV_DOCFOR:objc_class:Panel;,Panel

Inherits From: Window : Responder : Object

Declared In: appkit/Panel.h

Class Description

A Panel is a Window that serves an auxiliary function within an application; it contains Views that give information to users and let users give instructions to the application. Usually, the Views are Control objects of some sortÐButtons, Forms, NXBrowsers, TextFields, Sliders, and so on. Menu is a sublcass of Panel.

Panels behave differently from other Windows in only a small number of ways, but the ways are important to the user interface:

- Panels pass Command key-down events to the objects in their view hierarchies. This permits them to have keyboard alternatives.
- Panels aren't destroyed when closed; they're simply moved off-screen (taken out of the screen list).
- On-screen Panels are removed from the screen list when the user begins to work in another application, and are restored to the screen when the user returns to the Panel's application.
- Panels have a light gray, rather than white, background in their content area.

To facilitate their intended roles in the user interface, some panels can be assigned special behaviors:

- A panel can be precluded from becoming the key window until the user makes a selection (makes a View the first responder) indicating an intention to begin typing. This prevents key window status from shifting to the Panel unnecessarily.
- 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.
- A Panel can be made to workÐto receive mouse and keyboard eventsÐeven when there's an attention panel on-screen. This permits actions within the Panel to affect the attention panel.

Instance Variables

None declared in this class.

Method Types

Initializing a new Panel	- init - initContent:style:backing:buttonMask:defer:
Handling events	- commandKey: - keyDown:
Determining the Panel interface	- setBecomeKeyOnlyIfNeeded: - doesBecomeKeyOnlyIfNeeded - setFloatingPanel: - isFloatingPanel

- setWorksWhenModal:
- worksWhenModal

Instance Methods

NS_DEV_DOCFOR:objc_method:[Panel-commandKey:];, commandKey:

- (BOOL)**commandKey:**(NXEvent *)*theEvent*

Intercepts **commandKey:** messages being passed from Window to Window, and translates them to **performKeyEquivalent:** messages for the Views within the Panel. This method returns YES if any of the Views can handle the event as its keyboard alternative, and NO if none of them can. A return value of NO continues the **commandKey:** message down the Application object's list of Windows; a return value of YES terminates it.

The Application object initiates **commandKey:** messages when it gets key-down events with the Command key pressed. The Panel also initiates them, but just to itself, when it gets a **keyDown:** event message. The argument, *theEvent*, is a pointer to the key-down event.

Before any **performKeyEquivalent:** messages are sent, a Panel that's not on-screen receives an **update** message. This gives it a chance to make sure that its Views are properly enabled or disabled to reflect the current state of the application.

See also: - **keyDown:**, - **performKeyEquivalent:** (View)

NS_DEV_DOCFOR:objc_method:[Panel-doesBecomeKeyOnlyIfNeeded];, doesBecomeKeyOnlyIfNeeded

- (BOOL)**doesBecomeKeyOnlyIfNeeded**

Returns whether the Panel refrains from becoming the key window until the user clicks within a View that can become the first responder. The default return value is NO.

See also: - **setBecomeKeyOnlyIfNeeded:**

NS_DEV_DOCFOR:objc_method:[Panel-init];, init

- **init**

Initializes the receiver, a newly allocated Panel object, by sending it an **initContent:style:backing:buttonMask:defer:** message with default parameters, and returns **self**.

The Panel will have a content rectangle of minimal size. The Window Server won't create a window for the Panel until the Panel is ready to be displayed on-screen; the window will be buffered. The Panel will have a title bar and close button, but no resize bar. Like all Windows, it's initially placed out of the screen list. The Panel has no title.

See also: - **initContent:style:backing:buttonMask:defer:**

NS_DEV_DOCFOR:objc_method:[Panel-initContent:style:backing:buttonMask:defer:];, initContent:style:backing:buttonMask:defer:

- **initContent:**(const NXRect *)*contentRect*
style:(int)*aStyle*
backing:(int)*bufferingType*
buttonMask:(int)*mask*
defer:(BOOL)*flag*

Initializes the receiver, a newly allocated Panel instance, and returns **self**.

This method is the designated initializer for this class. It's identical to the Window method of the

same name, except that it additionally initializes the receiver so that it will behave like a panel in the user interface:

- The Panel's background color is set to be light gray.
- The Panel will hide when the application it belongs to is deactivated.
- The Panel won't be freed when the user closes it.

The new Panel is initially out of the Window Server's screen list. To make it visible, you must **display** it (into the buffer) and then move it on-screen.

See also: - **initContent:style:backing:buttonMask:defer:** (Window)

NS_DEV_DOCFOR:objc_method:[Panel-isFloatingPanel];, isFloatingPanel

- (BOOL)**isFloatingPanel**

Returns whether the Panel floats above standard windows and other panels. The default is NO.

See also: - **setFloatingPanel:**

NS_DEV_DOCFOR:objc_method:[Panel-keyDown:];, keyDown:

- **keyDown:**(NXEvent *)*theEvent*

Translates the key-down event into a **commandKey:** message for the Panel, thus interpreting the event as a potential keyboard alternative. If the Panel has a button that displays the Return symbol and the key-down event is for the Return key, it will operate the button.

A Panel receives **keyDown:** event messages only when it's the key window and either:

- none of its Views is the first responder,
- none of the Views in its responder chain implements a **keyDown:** method, or
- the Views in its responder chain that implement a **keyDown:** method include the message [super keyDown:theEvent].

See also: - **commandKey:**

NS_DEV_DOCFOR:objc_method:[Panel-setBecomeKeyOnlyIfNeeded:];, setBecomeKeyOnlyIfNeeded:

- **setBecomeKeyOnlyIfNeeded:**(BOOL)*flag*

Sets whether the Panel becomes the key window only when the user makes a selection (causing one of its Views to become the first responder). Since this requires the user to perform an extra action (clicking in the View) before being able to type within the window, it's appropriate only for Panels that don't normally require text entry. You should consider setting this attribute only if (1) most of the controls within the Panel are not text fields, and (2) the choices that can be made by entering text can also be made in another way (or are only incidental to the way the panel is normally used). The default is NO. Returns **self**.

See also: - **doesBecomeKeyOnlyIfNeeded**, - **keyDown:**

NS_DEV_DOCFOR:objc_method:[Panel-setFloatingPanel:];, setFloatingPanel:

- **setFloatingPanel:**(BOOL)*flag*

Sets whether the Panel should be assigned to a window tier above standard windows. The default is NO. It's appropriate for a Panel to float above other windows only if:

- It's oriented to the mouse rather than the keyboardÐthat is, it doesn't become the key window (or becomes the key window only if needed),

- It needs to remain visible while the user works in the application's standard windowsÐfor example, if the user must frequently move the cursor back and forth between a standard window and the panel (such as a tool palette) or the panel gives information relevant to the user's actions within a standard window,
- It's small enough not to obscure much of what's behind it, and
- It doesn't remain on-screen when the application is deactivated.

All four of these conditions should be true for *flag* to be set to YES. Returns **self**.

See also: - isFloatingPanel

NS_DEV_DOCFOR:objc_method:[Panel-setWorksWhenModal:];, setWorksWhenModal:

- setWorksWhenModal:(BOOL)*flag*

Sets whether the Panel remains enabled to receive events and possibly become the key window even when a modal panel (attention panel) is on-screen. This is appropriate only for a Panel that needs to operate on attention panels. The default is NO. Returns **self**.

See also: - worksWhenModal

NS_DEV_DOCFOR:objc_method:[Panel-worksWhenModal];, worksWhenModal

- (BOOL)**worksWhenModal**

Returns whether the Panel can receive keyboard and mouse events and possibly become the key window, even when a modal panel (attention panel) is on-screen. The default is NO.

See also: - setWorksWhenModal: