free

Setting the next responder setNextResponder:
nextResponder

Determining the first responder acceptsFirstResponder
becomeFirstResponder
resignFirstResponder

Aiding event processing performKeyEquivalent:
tryToPerform:with:

Forwarding event messages mouseDown:
rightMouseDown:
mouseDragged:
rightMouseDragged:
mouseUp:
rightMouseUp:
mouseMoved:
mouseEntered:
mouseExited:
keyDown:
keyUp:
flagsChanged:
noResponderFor:

Services menu support validRequestorForSendType:andReturnType:

Help menu support helpRequested:

Archiving read:
write:

(BOOL)acceptsFirstResponder

Returns NO to indicate that, by default, a Responder doesn't agree to become the first responder.

Before making any object the first responder, the Application Kit gives it an opportunity to refuse by sending it an

returns self. Responder subclasses can implement their own versions to take whatever action may highlighting the selection.

By returning self, the receiver accepts being made the first responder. A Responder can refuse to b responder by returning nil.

becomeFirstResponder messages are initiated by the Window object (through its makeFirstRespon response to mouse-down events.

resignFirstResponder, makeFirstResponder: (Window)

flagsChanged:(NXEvent *)theEvent

Passes the flagsChanged: event message to the receiver's next responder.

free

Frees the space used by a Responder instance and removes it from the hash table used to locate hel

helpRequested:(NXEvent *)eventPtr

Invoked by a Window instance when the user has clicked for help. The Window instance sends thi responder. The receiver shows its help panel if it has one, and if not forwards the message to the n is no next responder to respond, the method executes NXBeep(). Your application should never inv directly. Returns self.

keyDown:(NXEvent *)theEvent

Passes the keyDown: event message to the receiver's next responder.

keyUp:(NXEvent *)theEvent

Passes the keyUp: event message to the receiver's next responder.

mouseDown:(NXEvent *)theEvent

Passes the mouseDown: event message to the receiver's next responder.

mouseDragged:(NXEvent *)theEvent

Passes the mouseDragged: event message to the receiver's next responder.

mouseMoved:(NXEvent *)theEvent

Passes the mouseMoved: event message to the receiver's next responder.

mouseUp:(NXEvent *)theEvent

Passes the mouseUp: event message to the receiver's next responder.

nextResponder

Returns the receiver's next responder.

setNextResponder:

noResponderFor:(const char *)eventType

Responds to an event message that has reached the end of the responder chain without finding an o
When the event is a key down, noResponderFor: generates a beep.

(BOOL)performKeyEquivalent:(NXEvent *)theEvent

Returns NO to indicate that, by default, the Responder doesn't have a key equivalent and can't resp
events as keyboard alternatives.

The Responder class implements this method so that any object that inherits from it can be asked to
performKeyEquivalent: message. Subclasses that define objects with key equivalents must implem
of performKeyEquivalent:. If the key in theEvent matches the receiver's key equivalent, it should
and return YES.

performKeyEquivalent: (View and Button)

read:(NXTypedStream *)stream

Reads the Responder from the typed stream stream. Returns self.

write:

resignFirstResponder

Notifies the receiver that it has been asked to relinquish its status as first responder for its Window
of the method simply returns self. Responder subclasses can implement their own versions to take
be necessary, such as unhighlighting the selection.

rightMouseDown:(NXEvent *)theEvent

Passes the rightMouseDown: event message to the receiver's next responder.

rightMouseDragged:(NXEvent *)theEvent

Passes the rightMouseDragged: event message to the receiver's next responder.

rightMouseUp:(NXEvent *)theEvent

Passes the rightMouseUp: event message to the receiver's next responder.

setNextResponder:aResponder

Makes aResponder the receiver's next responder.

nextResponder

(BOOL)tryToPerform:(SEL)anAction with:anObject

Aids in dispatching action messages. This method checks to see whether the receiving object can r
selector specified by anAction. If it can, the message is sent with anObject as an argument. Typic
initiator of the action message.

If the receiver can't respond, tryToPerform:with: checks to see whether the receiving object's next
continues to follow next responder links up the responder chain until it finds an object that it can s
to, or the chain is exhausted.

Even if the receiver can respond to anAction messages, it can ªrefuseº them by having its impleme
method return nil. In this case, the message is passed on to the next responder in the chain.

If successful in finding a receiver that doesn't refuse the message, tryToPerform: returns YES. Ot

This method is used (indirectly, through the sendAction:to:from: method) to dispatch action messa
objects. You'd rarely have reason to use it yourself.

sendAction:to:from: (Application)

validRequestorForSendType:(NXAtom)typeSent
andReturnType:(NXAtom)typeReturned

Implemented by subclasses to determine what services are available at any given time. In order to
menu current, the Application object sends validRequestorForSendType:andReturnType: message
with the send and return types for each service method of every service provider. Thus, a Respond
message many times per event. If the receiving object can place data of type typeSent on the paste
of type typeReturned back, it should return self otherwise it should return nil. The Application obj
value to determine whether to enable or disable commands in the Services menu.

When the user chooses a menu item for a service, a writeSelectionToPasteboard:types: message is
(if typeSent was not NULL). The Responder writes the requested data to the pasteboard and a rem
the service. If the service's typeReturned is not NULL, it places return data on the pasteboard, and
receives a readSelectionFromPasteboard: message.

The following example demonstrates an implementation of the validRequestorForSendType:andR
an object that can send and receive ASCII text. Pseudocode is in italics.

registerServicesMenuSendTypes:andReturnTypes: (Application), writeSelectionToPasteboard:ty
readSelectionFromPasteboard: (Application)

write:(NXTypedStream *)stream

Writes the receiving Responder to the typed stream stream. The next responder is not explicitly w

read: