NSEvent Class Reference

Cocoa > Events & Other Input

2009-02-04

Contents

NSEvent Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Cocoa Event-Handling Guide
Declared in	NSEvent.h
Related sample code	Cropped Image DragItemAround NSGLImage OpenGL Screensaver Sketch-112

Overview

An `NSEvent` object, or simply an event, contains information about an input action such as a mouse click or a key down. The Application Kit associates each such user action with a window, reporting the event to the application that created the window. The `NSEvent` object contains pertinent information about each event, such as where the cursor was located or which character was typed. As the application receives events, it temporarily places them in a buffer called the event queue. When the application is ready to process an event, it takes one from the queue.

Beginning with Mac OS X version 10.4, `NSEvent` objects can represent tablet-pointing and tablet-proximity events. A tablet-proximity event is generated when a pointing device enters or leaves proximity of its tablet; such event objects have a type of `NSTypeProximity` or a mouse subtype of `NSTabletProximityEventSubtype`. A tablet-pointing event is generated when a pointing device changes state, such as location, pressure, or tilt; such event objects have a type of `NSTypePoint` or a mouse subtype of `NSTabletPointEventSubtype`. The Application Kit reports all pure tablet events to responder objects through the `NSResponder` methods `tabletPoint:` and `tabletProximity:`. Mouse events can also contain tablet data (as event subtypes), so you can handle these events by overriding the `NSResponder` methods `mouseDown:`, `mouseDragged:`, and `mouseUp:`.

Adopted Protocols

NSCoding
  - `encodeWithCoder:`
  - `initWithCoder:`

NSCopying
  - `copyWithZone:`

Tasks

Creating Events

`+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 14)
: Returns a new `NSEvent` object describing a key event.

`+ mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:` (page 15)
: Returns a new NSEvent object describing a mouse-down, -up, -moved, or -dragged event.

`+ enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 12)
: Returns a new `NSEvent` object describing a tracking-rectangle or cursor-update event.

`+ otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 17)
: Returns a new NSEvent object describing a custom event.

`+ eventWithEventRef:` (page 13)
: Creates an event object that is based on a Carbon type of event.

`+ eventWithCGEvent:` (page 13)
: Creates and returns an event object that is based on a Core Graphics type of event.

Getting General Event Information

`- context` (page 24)
: Returns the display graphics context of the receiver.

`- locationInWindow` (page 29)
: Returns the receiver's location in the base coordinate system of the associated window.

`- modifierFlags` (page 29)
: Returns an integer bit field indicating the modifier keys in effect for the receiver.

`- timestamp` (page 34)
: Returns the time the receiver occurred in seconds since system startup.

`- type` (page 35)
: Returns the type of the receiving event.

- `window` (page 38)
  Returns the window object associated with the receiver.
- `windowNumber` (page 38)
  Returns the identifier for the window device associated with the receiver.
- `eventRef` (page 27)
  Returns the Carbon type associated with the receiver for representing an event.
- `CGEvent` (page 21)
  Returns a Core Graphics event object corresponding to the receiver.

Getting Key Event Information

- `characters` (page 22)
  Returns the characters associated with the receiving key-up or key-down event.
- `charactersIgnoringModifiers` (page 22)
  Returns the characters generated by the receiving key event as if no modifier key (except for Shift) applies.
- `isARepeat` (page 27)
  Returns `YES` if the receiving key event is a repeat caused by the user holding the key down, `NO` if the key event is new.
- `keyCode` (page 28)
  Returns the virtual key code for the keyboard key associated with the receiving key event.

Getting Mouse Event Information

+ `mouseLocation` (page 16)
  Reports the current mouse position in screen coordinates.

- `buttonNumber` (page 21)
  Returns the button number for the mouse button that generated an `NSOtherMouse...` event.
- `clickCount` (page 23)
  Returns the number of mouse clicks associated with the receiver, which represents a mouse-down or mouse-up event.
- `pressure` (page 31)
  Returns a value from 0.0 through 1.0 indicating the pressure applied to the input device (used for appropriate devices).

+ `setMouseCoalescingEnabled:` (page 18)
  Controls whether mouse-movement event coalescing is enabled.

+ `isMouseCoalescingEnabled` (page 14)
  Indicates whether mouse-movement event coalescing is enabled.

Getting Mouse-Tracking Event Information

- `eventNumber` (page 27)
  Returns the counter value of the latest mouse or tracking-rectangle event object; every system-generated mouse and tracking-rectangle event increments this counter.

– `trackingNumber` (page 35)
  Returns the identifier of a mouse-tracking event.

– `trackingArea` (page 34)
  Returns the `NSTrackingArea` object that generated the event represented by the receiver.

– `userData` (page 37)
  Returns data associated with a mouse-tracking event,

Getting Custom Event Information

– `data1` (page 24)
  Returns additional data associated with the receiver.

– `data2` (page 24)
  Returns additional data associated with the receiver.

– `subtype` (page 32)
  Returns the subtype of the receiving event object.

Getting Scroll Wheel Event Information

– `deltaX` (page 25)
  Returns the x-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

– `deltaY` (page 25)
  Returns the y-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

– `deltaZ` (page 26)
  Returns the z-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

Getting Tablet Proximity Information

– `capabilityMask` (page 21)
  Returns a mask whose set bits indicate the capabilities of the tablet device that generated the event represented by the receiver.

– `deviceID` (page 26)
  Returns a special identifier that is used to match tablet-pointer events with the tablet-proximity event represented by the receiver.

– `isEnteringProximity` (page 28)
  Returns `YES` to indicate that a pointing device is entering the proximity of its tablet and `NO` when it is leaving it.

– `pointingDeviceID` (page 30)
  Returns the index of the pointing device currently in proximity with the tablet.

– `pointingDeviceSerialNumber` (page 30)
  Returns the vendor-assigned serial number of a pointing device of a certain type.

– `pointingDeviceType` (page 31)
  Returns a `NSPointingDeviceType` constant indicating the kind of pointing device associated with the receiver.

- `systemTabletID` (page 32)
  Returns the index of the tablet device connected to the system.
- `tabletID` (page 33)
  Returns the USB model identifier of the tablet device associated with the receiver.
- `uniqueID` (page 36)
  Returns the unique identifier of the pointing device that generated the event represented by the receiver.
- `vendorID` (page 37)
  Returns the vendor identifier of the tablet associated with the receiver.
- `vendorPointingDeviceType` (page 38)
  Returns a coded bit field whose set bits indicate the type of pointing device (within a vendor selection) associated with the receiver.

Getting Tablet Pointing Information

- `absoluteX` (page 19)
  Reports the absolute x coordinate of a pointing device on its tablet at full tablet resolution.
- `absoluteY` (page 20)
  Reports the absolute y coordinate of a pointing device on its tablet at full tablet resolution.
- `absoluteZ` (page 20)
  Reports the absolute z coordinate of pointing device on its tablet at full tablet resolution.
- `buttonMask` (page 20)
  Returns a bit mask identifying the buttons pressed when the tablet event represented by the receiver was generated.
- `rotation` (page 32)
  Returns the rotation in degrees of the tablet pointing device associated with the receiver.
- `tangentialPressure` (page 33)
  Reports the tangential pressure on the device that generated the event represented by the receiver.
- `tilt` (page 34)
  Reports the scaled tilt values of the pointing device that generated the event represented by the receiver.
- `vendorDefined` (page 37)
  Returns an array of three vendor-defined `NSNumber` objects associated with the pointing-type event represented by the receiver.

Requesting and Stopping Periodic Events

+ `startPeriodicEventsAfterDelay:withPeriod:` (page 18)
  Begins generating periodic events for the current thread.

+ `stopPeriodicEvents` (page 19)
  Stops generating periodic events for the current thread and discards any periodic events remaining in the queue.

Class Methods

enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:

Returns a new NSEvent object describing a tracking-rectangle or cursor-update event.

```
+ (NSEvent *)enterExitEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNumber context:(NSGraphicsContext *)context
    eventNumber:(NSInteger)eventNumber trackingNumber:(NSInteger)trackingNumber
    userData:(void *)userData
```

Parameters

type

One of the following event-type constants: NSMouseEntered, NSMouseExited, NSCursorUpdate. If the specified constant is not one of these, an NSInternalInconsistencyException is raised

location

The cursor location in the base coordinate system of the window specified by *windowNum*.

flags

An integer bit field containing any of the modifier key masks described in "Constants" (page 39), combined using the C bitwise OR operator.

time

The time the event occurred in seconds since system startup.

windowNum

An integer that identifies the window device associated with the event, which is associated with the NSWindow that will receive the event.

context

The display graphics context of the event.

eventNumber

An identifier for the new event. It's normally taken from a counter for mouse events, which continually increases as the application runs.

trackingNumber

A number that identifies the tracking rectangle. This identifier is the same as that returned by the NSView method addTrackingRect:owner:userData:assumeInside:.

userData

Data arbitrarily associated with the tracking rectangle when it was set up using the NSView method addTrackingRect:owner:userData:assumeInside:.

Return Value

The created NSEvent object or nil if the object could not be created.

Availability

Available in Mac OS X v10.0 and later.

See Also

- eventNumber (page 27)
- trackingNumber (page 35)

- `userData` (page 37)

Declared In
`NSEvent.h`

eventWithCGEvent:

Creates and returns an event object that is based on a Core Graphics type of event.

```
+ (NSEvent *)eventWithCGEvent:(CGEventRef)cgEvent
```

Parameters

`cgEvent`
: A `CGEventRef` opaque type that represents an event.

Return Value
An autoreleased `NSEvent` object that is equivalent to *`cgEvent`*.

Discussion
The returned object retains the `CGEventRef` object (`cgEvent`) until it (the Objective-C object) is freed—it then releases the `CGEventRef` object. If no Cocoa event corresponds to the `CGEventRef` object, this method returns `nil`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `CGEvent` (page 21)

Declared In
`NSEvent.h`

eventWithEventRef:

Creates an event object that is based on a Carbon type of event.

```
+ (NSEvent *)eventWithEventRef:(const void *)eventRef
```

Parameters

`eventRef`
: The `EventRef` opaque type to be associated with the created `NSEvent` object.

Return Value
An autoreleased `NSEvent` object corresponding to *`eventRef`* or `nil` if *`eventRef`* cannot be converted into an equivalent `NSEvent` object.

Discussion
This method is valid for all events. The created `NSEvent` object retains the `EventRef` object and is released when the `NSEvent` object is freed.

Availability
Available in Mac OS X version 10.5 and later.

See Also
`- eventRef` (page 27)

Declared In
`NSEvent.h`

isMouseCoalescingEnabled

Indicates whether mouse-movement event coalescing is enabled.

```
+ (BOOL)isMouseCoalescingEnabled
```

Return Value
`YES` if mouse-movement event coalescing is enabled, `NO` if it is disabled.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ setMouseCoalescingEnabled:` (page 18)

Declared In
`NSEvent.h`

keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:

Returns a new `NSEvent` object describing a key event.

```
+ (NSEvent *)keyEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNum context:(NSGraphicsContext *)context
    characters:(NSString *)characters charactersIgnoringModifiers:(NSString
    *)unmodCharacters isARepeat:(BOOL)repeatKey keyCode:(unsigned short)code
```

Parameters

`type`
: One of the following event-type constants: `NSKeyDown`, `NSKeyUp`, `NSFlagsChanged`. If anything else is specified, an `NSInternalInconsistencyException` is raised.

`location`
: The cursor location in the base coordinate system of the window specified by *`windowNum`*.

`flags`
: An integer bit field containing any of the modifier key masks described in "Constants" (page 39), combined using the C bitwise OR operator.

`time`
: The time the event occurred in seconds since system startup.

`windowNum`
: An integer that identifies the window device associated with the event, which is associated with the `NSWindow` that will receive the event.

context
: The display graphics context of the event.

characters
: A string of characters associated with the key event. Though most key events contain only one character, it is possible for a single keypress to generate a series of characters.

unmodCharacters
: The string of characters generated by the key event as if no modifier key had been pressed (except for Shift). This argument is useful for getting the "basic" key value in a hardware-independent manner.

repeatKey
: `YES` if the key event is a repeat caused by the user holding the key down, `NO` if the key event is new.

code
: A number that identifies the keyboard key associated with the key event. Its value is hardware-independent.

Return Value
The created `NSEvent` instance or `nil` if the instance could not be created.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `characters` (page 22)
- `charactersIgnoringModifiers` (page 22)
- `isARepeat` (page 27)
- `keyCode` (page 28)

Declared In
`NSEvent.h`

mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:

Returns a new NSEvent object describing a mouse-down, -up, -moved, or -dragged event.

```
+ (NSEvent *)mouseEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNum context:(NSGraphicsContext *)context
    eventNumber:(NSInteger)eventNumber clickCount:(NSInteger)clickNumber
    pressure:(float)pressure
```

Parameters

type
: One of the modifier key masks described in "Constants" (page 39), or an `NSInternalInconsistencyException` is raised.

location
: The cursor location in the base coordinate system of the window specified by *windowNum*.

flags
: An integer bit field containing any of the modifier key masks described in "Constants" (page 39), combined using the C bitwise OR operator.

time
: The time the event occurred in seconds since system startup.

windowNum
: An integer that identifies the window device associated with the event, which is associated with the `NSWindow` that will receive the event.

context
: The display graphics context of the event.

eventNumber
: An identifier for the new event. It's normally taken from a counter for mouse events, which continually increases as the application runs.

clickNumber
: The number of mouse clicks associated with the mouse event.

pressure
: A value from 0.0 to 1.0 indicating the pressure applied to the input device on a mouse event, used for an appropriate device such as a graphics tablet. For devices that aren't pressure-sensitive, the value should be either 0.0 or 1.0.

Return Value
The created `NSEvent` instance or `nil` if the instance could not be created.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clickCount` (page 23)
- `eventNumber` (page 27)
- `pressure` (page 31)

Declared In
`NSEvent.h`

mouseLocation

Reports the current mouse position in screen coordinates.

```
+ (NSPoint)mouseLocation
```

Discussion
This method is similar to the `NSWindow` method `mouseLocationOutsideOfEventStream`. It returns the location regardless of the current event or pending events. The difference between these methods is that `mouseLocationOutsideOfEventStream` returns a point in the receiving window's coordinates and `mouseLocation` returns the same information in screen coordinates.

Note: The y coordinate in the returned point starts from a base of 1, not 0.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ImageMapExample

Quartz Composer Matrix

Declared In
`NSEvent.h`

otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:

Returns a new NSEvent object describing a custom event.

```
+ (NSEvent *)otherEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNum context:(NSGraphicsContext *)context
    subtype:(short)subtype data1:(NSInteger)data1 data2:(NSInteger)data2
```

Parameters

type

One of the following event-type constants:

`NSAppKitDefined`

`NSSystemDefined`

`NSApplicationDefined`

`NSPeriodic`

If *type* is anything else, an `NSInternalInconsistencyException` is raised. Your code should only create events of type `NSApplicationDefined`.

location

The cursor location in the base coordinate system of the window specified by *windowNum*.

flags

An integer bit field containing any of the modifier key masks described in "Constants" (page 39), combined using the C bitwise OR operator.

time

The time the event occurred in seconds since system startup.

windowNum

An integer that identifies the window device associated with the event, which is associated with the `NSWindow` that will receive the event.

context

The display graphics context of the event.

subtype

A numeric identifier that further differentiates custom events of types `NSAppKitDefined`, `NSSystemDefined`, and `NSApplicationDefined`. `NSPeriodic` events don't use this attribute.

data1

Additional data associated with the event. `NSPeriodic` events don't use these attributes.

data2

Additional data associated with the event. `NSPeriodic` events don't use these attributes.

Return Value
The created `NSEvent` object or `nil` if the object couldn't be created.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- subtype` (page 32)
- `- data1` (page 24)
- `- data2` (page 24)

Declared In
`NSEvent.h`

setMouseCoalescingEnabled:

Controls whether mouse-movement event coalescing is enabled.

```
+ (void)setMouseCoalescingEnabled:(BOOL)flag
```

Parameters

flag
  `YES` to enable mouse-movement event coalescing, `NO` to disable it.

Discussion
This method affects mouse-moved, mouse-dragged, and tablet events. Mouse-movement event coalescing is enabled by default.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ isMouseCoalescingEnabled` (page 14)

Declared In
`NSEvent.h`

startPeriodicEventsAfterDelay:withPeriod:

Begins generating periodic events for the current thread.

```
+ (void)startPeriodicEventsAfterDelay:(NSTimeInterval)delaySeconds
    withPeriod:(NSTimeInterval)periodSeconds
```

Parameters

delaySeconds
  The number of seconds that `NSEvent` should wait before beginning to generate periodic events.

periodSeconds
  The period in seconds between the generated events.

Discussion
Raises an `NSInternalInconsistencyException` if periodic events are already being generated for the current thread. This method is typically used in a modal loop while tracking mouse-dragged events.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ stopPeriodicEvents` (page 19)

Declared In
`NSEvent.h`

stopPeriodicEvents

Stops generating periodic events for the current thread and discards any periodic events remaining in the queue.

```
+ (void)stopPeriodicEvents
```

Discussion
This message is ignored if periodic events aren't currently being generated.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ startPeriodicEventsAfterDelay:withPeriod:` (page 18)

Declared In
`NSEvent.h`

Instance Methods

absoluteX

Reports the absolute x coordinate of a pointing device on its tablet at full tablet resolution.

```
- (NSInteger)absoluteX
```

Discussion
For the coordinate to be valid, the receiver should represent an event generated by a tablet pointing device (otherwise 0 is returned). This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`. Use this value if you want to scale from tablet location to screen location yourself; otherwise use the class method `mouseLocation` (page 16) or the instance method `locationInWindow` (page 29).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- absoluteY` (page 20)
`- absoluteZ` (page 20)

Declared In
`NSEvent.h`

absoluteY

Reports the absolute y coordinate of a pointing device on its tablet at full tablet resolution.

```
- (NSInteger)absoluteY
```

Discussion
For the coordinate to be valid, the receiver should represent an event generated by a tablet pointing device (otherwise 0 is returned). This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`. Use this value if you want to scale from tablet location to screen location yourself; otherwise use the class method `mouseLocation` (page 16) or the instance method `locationInWindow` (page 29).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- absoluteX` (page 19)
`- absoluteZ` (page 20)

Declared In
`NSEvent.h`

absoluteZ

Reports the absolute z coordinate of pointing device on its tablet at full tablet resolution.

```
- (NSInteger)absoluteZ
```

Discussion
For the coordinate to be valid, the receiver should represent an event generated by a tablet pointing device (otherwise 0 is returned). The z coordinate does not represent pressure. It registers the depth coordinate returned by some tablet devices with wheels; if the device is something other than these, 0 is returned. This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- absoluteX` (page 19)
`- absoluteY` (page 20)

Declared In
`NSEvent.h`

buttonMask

Returns a bit mask identifying the buttons pressed when the tablet event represented by the receiver was generated.

```
- (NSUInteger)buttonMask
```

Discussion
Use one or more of the button-mask constants described in "Constants" (page 39) to determine which buttons of the pointing device are pressed. This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSEvent.h`

buttonNumber

Returns the button number for the mouse button that generated an `NSOtherMouse...` event.

```
- (NSInteger)buttonNumber
```

Discussion
This method is intended for use with the `NSOtherMouseDown`, `NSOtherMouseUp`, and `NSOtherMouseDragged` events, but will return values for `NSLeftMouse...` and `NSRightMouse...` events also.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`NSEvent.h`

capabilityMask

Returns a mask whose set bits indicate the capabilities of the tablet device that generated the event represented by the receiver.

```
- (NSUInteger)capabilityMask
```

Discussion
These bits are vendor-defined. This method is valid only for mouse events with a subtype of `NSTabletProximityEventSubtype` and for events of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSEvent.h`

CGEvent

Returns a Core Graphics event object corresponding to the receiver.

```
- (CGEventRef)CGEvent
```

Discussion
The returned `CGEventRef` opaque type is autoreleased. If no `CGEventRef` object corresponding to the `NSEvent` object can be created, this method returns `NULL`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ eventWithCGEvent:` (page 13)

Declared In
`NSEvent.h`

characters

Returns the characters associated with the receiving key-up or key-down event.

```
- (NSString *)characters
```

Discussion
These characters are derived from a keyboard mapping that associates various key combinations with Unicode characters. Raises an `NSInternalInconsistencyException` if sent to any other kind of event object.

This method returns an empty string for dead keys, such as Option-e. However, for a key combination such as Option-Shift-e this method returns the standard accent ("´").

For a list of constants corresponding to commonly-used Unicode characters, see *NSText Class Reference*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- charactersIgnoringModifiers` (page 22)
`+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 14)

Related Sample Code
CocoaDVDPlayer
NSGLImage
OpenGL Screensaver
TrackBall

Declared In
`NSEvent.h`

charactersIgnoringModifiers

Returns the characters generated by the receiving key event as if no modifier key (except for Shift) applies.

```
- (NSString *)charactersIgnoringModifiers
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to a nonkey event.

This method returns the non-modifier key character pressed for dead keys, such as Option-e. For example, Option-e (no shift key) returns an “e" for this method, whereas the `characters` (page 22) method returns an empty string.

This method is useful for determining “basic” key values in a hardware-independent manner, enabling such features as keyboard equivalents defined in terms of modifier keys plus character keys. For example, to determine if the user typed Alt-S, you don’t have to know whether Alt-S generates a German double ess, an integral sign, or a section symbol. You simply examine the string returned by this method along with the event’s modifier flags, checking for “s” and `NSAlternateKeyMask`.

For a list of constants corresponding to commonly-used Unicode characters, see *NSText Class Reference*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- characters` (page 22)
`- modifierFlags` (page 29)
`+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 14)

Related Sample Code
DragItemAround
LayerBackedOpenGLView
NSOpenGL Fullscreen
PDFKitLinker2
QTQuartzPlayer

Declared In
`NSEvent.h`

clickCount

Returns the number of mouse clicks associated with the receiver, which represents a mouse-down or mouse-up event.

```
- (NSInteger)clickCount
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to a nonmouse event.

Returns 0 for a mouse-up event if a time threshold has passed since the corresponding mouse-down event. This is because if this time threshold passes before the mouse button is released, it is no longer considered a mouse click, but a mouse-down event followed by a mouse-up event.

The return value of this method is meaningless for events other than mouse-down or mouse-up events.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure: (page 15)

Declared In
`NSEvent.h`

context

Returns the display graphics context of the receiver.

```
- (NSGraphicsContext *)context
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSEvent.h`

data1

Returns additional data associated with the receiver.

```
- (NSInteger)data1
```

Discussion
The value returned by this method is dependent on the event type, and is defined by the originator of the event. Raises an `NSInternalInconsistencyException` if sent to an event not of type `NSAppKitDefined`, `NSSystemDefined`, `NSApplicationDefined`, or `NSPeriodic`.

`NSPeriodic` events don't use this attribute.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `data2` (page 24)
- `subtype` (page 32)
+ otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2: (page 17)

Declared In
`NSEvent.h`

data2

Returns additional data associated with the receiver.

```
- (NSInteger)data2
```

Discussion
The value returned by this method is dependent on the event type, and is defined by the originator of the event. Raises an `NSInternalInconsistencyException` if sent to an event not of type `NSAppKitDefined`, `NSSystemDefined`, `NSApplicationDefined`, or `NSPeriodic`.

`NSPeriodic` events don't use this attribute.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– data1` (page 24)
`– subtype` (page 32)
`+ otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 17)

Declared In
`NSEvent.h`

deltaX

Returns the x-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

```
- (CGFloat)deltaX
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`– deltaY` (page 25)
`– deltaZ` (page 26)

Declared In
`NSEvent.h`

deltaY

Returns the y-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

```
- (CGFloat)deltaY
```

Discussion
The behavior of this method may seem counter-intuitive: as the mouse moves up the screen, the value is negative; and as it moves down the screen, the value is positive. The reason for this behavior is that `NSEvent` computes this delta value in device space, which is flipped, but both the screen and the window's base coordinate system are not flipped.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– deltaX` (page 25)

`- deltaZ` (page 26)

Declared In
`NSEvent.h`

deltaZ

Returns the z-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

```
- (CGFloat)deltaZ
```

Discussion
This value is typically 0.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- deltaX` (page 25)
`- deltaY` (page 25)

Declared In
`NSEvent.h`

deviceID

Returns a special identifier that is used to match tablet-pointer events with the tablet-proximity event represented by the receiver.

```
- (NSUInteger)deviceID
```

Discussion
All tablet-pointer events generated in the period between the device entering and leaving tablet proximity have the same device ID. This message is valid only for mouse events with subtype `NSTabletPointEventSubtype` or `NSTabletProximityEventSubtype`, and for `NSTabletPoint` and `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- pointingDeviceID` (page 30)
`- systemTabletID` (page 32)
`- tabletID` (page 33)

Declared In
`NSEvent.h`

eventNumber

Returns the counter value of the latest mouse or tracking-rectangle event object; every system-generated mouse and tracking-rectangle event increments this counter.

```
- (NSInteger)eventNumber
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to any other type of event object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 12)
`+ mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:` (page 15)

Declared In
`NSEvent.h`

eventRef

Returns the Carbon type associated with the receiver for representing an event.

```
- (const void *)eventRef
```

Return Value
Returns an `EventRef` opaque type corresponding to the receiver. User-input events typically are created with an associated `EventRef`. An `NSEvent` object created through other means creates an `EventRef` in this method if that is necessary and possible. If there is no equivalent `NSEvent` for the receiver, this method returns `NULL`.

Discussion
This method is valid for all types of events. The `EventRef` object is retained by the receiver, so it is valid as long as the `NSEvent` object is valid, and is released when the `NSEvent` object is freed. You can use `RetainEvent` to extend the lifetime of the `EventRef` object, with a corresponding `ReleaseEvent` when you are done with it.

Availability
Available in Mac OS X version 10.5 and later.

See Also
`+ eventWithEventRef:` (page 13)

Declared In
`NSEvent.h`

isARepeat

Returns `YES` if the receiving key event is a repeat caused by the user holding the key down, `NO` if the key event is new.

```
- (BOOL)isARepeat
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to an `NSFlagsChanged` event or other nonkey event.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode: (page 14)

Declared In
`NSEvent.h`

isEnteringProximity

Returns `YES` to indicate that a pointing device is entering the proximity of its tablet and `NO` when it is leaving it.

```
- (BOOL)isEnteringProximity
```

Discussion
This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSEvent.h`

keyCode

Returns the virtual key code for the keyboard key associated with the receiving key event.

```
- (unsigned short)keyCode
```

Return Value
The virtual key code. The returned value is hardware-independent. The value returned is the same as the value returned in the `kEventParamKeyCode` when using Carbon Events.

Discussion
Raises an `NSInternalInconsistencyException` if sent to a non-key event.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode: (page 14)

Declared In
`NSEvent.h`

locationInWindow

Returns the receiver's location in the base coordinate system of the associated window.

```
- (NSPoint)locationInWindow
```

Discussion
For nonmouse events the return value of this method is undefined.

With `NSMouseMoved` and possibly other events, the receiver can have a `nil` window (that is, `window` (page 38) returns nil). In this case, `locationInWindow` returns the event location in screen coordinates.

In a method of a custom view that handles mouse events, you commonly use the `locationInWindow` method in conjunction with the NSView method `convertPoint:fromView:` to get the mouse location in the view's coordinate system. For example:

```
NSPoint event_location = [theEvent locationInWindow];
NSPoint local_point = [self convertPoint:event_location fromView:nil];
```

Note: The y coordinate in the returned point starts from a base of 1, not 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- window` (page 38)

Related Sample Code
Dicey
NSGLImage
QTCoreVideo102
QTCoreVideo201
Sketch-112

Declared In
`NSEvent.h`

modifierFlags

Returns an integer bit field indicating the modifier keys in effect for the receiver.

```
- (NSUInteger)modifierFlags
```

Discussion
You can examine individual flag settings using the C bitwise AND operator with the predefined key masks described in "Constants" (page 39). The lower 16 bits of the modifier flags are reserved for device-dependent bits.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Clock Control

CubePuzzle

GLChildWindowDemo

PDFKitLinker2

Sketch-112

Declared In
`NSEvent.h`

pointingDeviceID

Returns the index of the pointing device currently in proximity with the tablet.

```
- (NSUInteger)pointingDeviceID
```

Discussion
This index is significant for multimode (or Dual Tracking) tablets that support multiple concurrent pointing devices; the index is incremented for each pointing device that comes into proximity. Otherwise, zero is always returned. The receiver of this message should be a mouse event object with subtype `NSTabletProximityEventSubtype` or an event of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– pointingDeviceSerialNumber` (page 30)

`– pointingDeviceType` (page 31)

`– systemTabletID` (page 32)

Declared In
`NSEvent.h`

pointingDeviceSerialNumber

Returns the vendor-assigned serial number of a pointing device of a certain type.

```
- (NSUInteger)pointingDeviceSerialNumber
```

Discussion
Devices of different types, such as a puck and a pen, may have the same serial number. The receiver of this message should be a mouse event object with subtype `NSTabletProximityEventSubtype` or an event of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– pointingDeviceID` (page 30)

- `pointingDeviceType` (page 31)

Declared In
`NSEvent.h`

pointingDeviceType

Returns a `NSPointingDeviceType` constant indicating the kind of pointing device associated with the receiver.

```
- (NSPointingDeviceType)pointingDeviceType
```

Discussion
For example, the device could be a pen, eraser, or cursor pointing device. This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events. See "Constants" (page 39) for descriptions of valid `NSPointingDeviceType` constants.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceSerialNumber` (page 30)
- `pointingDeviceType` (page 31)

Declared In
`NSEvent.h`

pressure

Returns a value from 0.0 through 1.0 indicating the pressure applied to the input device (used for appropriate devices).

```
- (float)pressure
```

Discussion
For devices that aren't pressure-sensitive, the value is either 0.0 or 1.0. Raises an `NSInternalInconsistencyException` if sent to a nonmouse event.

For tablet pointing devices that are in proximity, the pressure value is 0.0 if they are not actually touching the tablet. As the device is pressed into the tablet, the value is increased.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ `mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:` (page 15)
- `rotation` (page 32)

Declared In
`NSEvent.h`

rotation

Returns the rotation in degrees of the tablet pointing device associated with the receiver.

```
- (float)rotation
```

Discussion
Many devices do not support rotation, in which case the returned value is 0.0. This method is valid only for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- pressure` (page 31)
`- tilt` (page 34)

Declared In
`NSEvent.h`

subtype

Returns the subtype of the receiving event object.

```
- (short)subtype
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to an event not of type `NSAppKitDefined`, `NSSystemDefined`, `NSApplicationDefined`, or `NSPeriodic`.

`NSPeriodic` events don't use this attribute.

This method is also valid for mouse events on Mac OS X v10.4 and later. See "Constants" (page 39) for the predefined mouse and tablet subtypes.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- data1` (page 24)
`- data2` (page 24)
`+ otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 17)

Declared In
`NSEvent.h`

systemTabletID

Returns the index of the tablet device connected to the system.

```
- (NSUInteger)systemTabletID
```

Discussion
If multiple tablets are connected to the system, the system-tablet ID is incremented for each subsequent one. If there is only one tablet device, its system-tablet ID is zero. The receiver of this message should be a mouse event object with subtype `NSTabletProximityEventSubtype` or an event of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceID` (page 30)
- `tabletID` (page 33)

Declared In
`NSEvent.h`

tabletID

Returns the USB model identifier of the tablet device associated with the receiver.

```
- (NSUInteger)tabletID
```

Discussion
This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceID` (page 30)
- `systemTabletID` (page 32)

Declared In
`NSEvent.h`

tangentialPressure

Reports the tangential pressure on the device that generated the event represented by the receiver.

```
- (float)tangentialPressure
```

Discussion
The value returned can range from -1.0 to 1.0. Tangential pressure is also known as barrel pressure. Only some pointing devices support tangential pressure. This method is valid for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pressure` (page 31)

Declared In
`NSEvent.h`

tilt

Reports the scaled tilt values of the pointing device that generated the event represented by the receiver.

```
- (NSPoint)tilt
```

Discussion
The value returned can range from -1.0 to 1.0 for both axes. An x-coordinate value that is negative indicates a tilt to the left and a positive value indicates a tilt to the right; a y-coordinate value that is negative indicates a tilt to the top and a positive value indicates a tilt to the bottom. If the device is perfectly perpendicular to the table surface, the values are 0.0 for both axes. This method is valid for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– pressure` (page 31)
`– rotation` (page 32)

Declared In
`NSEvent.h`

timestamp

Returns the time the receiver occurred in seconds since system startup.

```
- (NSTimeInterval)timestamp
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSEvent.h`

trackingArea

Returns the `NSTrackingArea` object that generated the event represented by the receiver.

```
- (NSTrackingArea *)trackingArea
```

Return Value
Returns the `NSTrackingArea` object that generated the event represented by the receiver. If the receiver is not a mouse-tracking event (that is, an event of type `NSMouseEntered`, `NSMouseExited`, or `NSCursorUpdate`), this method raises an `NSInternalInconsistencyException`. This method returns `nil` if the event was generated by a tracking rectangle (pre-Mac OS X version 10.5) instead of a `NSTrackingArea` object.

Discussion
If no `NSTrackingArea` object is associated with the event because the event corresponds to a tracking rectangle installed with the `NSView` method `addTrackingRect:owner:userData:assumeInside:`, this method returns `nil`. Note that the `trackingNumber` (page 35) method returns either an `NSTrackingArea` object or the `NSTrackingRectTag` constant depending on how the event was generated.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`NSEvent.h`

trackingNumber

Returns the identifier of a mouse-tracking event.

```
- (NSInteger)trackingNumber
```

Discussion
This method returns either an `NSTrackingArea` object or a `NSTrackingRectTag` constant depending on whether the event was generated from an `NSTrackingArea` object or a call to `addTrackingRect:owner:userData:assumeInside:`. Valid mouse-tracking methods are of types `NSMouseEntered`, `NSMouseExited`, and `NSCursorUpdate`. This method raises an `NSInternalInconsistencyException` if sent to any other type of event.

The `NSTrackingArea` class is new with Mac OS X version 10.5

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 12)
`- trackingArea` (page 34)

Declared In
`NSEvent.h`

type

Returns the type of the receiving event.

```
- (NSEventType)type
```

Discussion
The type must be one of the following:

```
NSLeftMouseDown
NSLeftMouseUp
NSRightMouseDown
NSRightMouseUp
NSOtherMouseDown
```

`NSOtherMouseUp`
`NSMouseMoved`
`NSLeftMouseDragged`
`NSRightMouseDragged`
`NSOtherMouseDragged`
`NSMouseEntered`
`NSMouseExited`
`NSKeyDown`
`NSKeyUp`
`NSFlagsChanged`
`NSAppKitDefined`
`NSSystemDefined`
`NSApplicationDefined`
`NSPeriodic`
`NSCursorUpdate`
`NSScrollWheel`

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
People
VBL

Declared In
`NSEvent.h`

uniqueID

Returns the unique identifier of the pointing device that generated the event represented by the receiver.

```
- (unsigned long long)uniqueID
```

Discussion
Also known as tool ID, this is a unique number recorded in the chip inside every pointing device. The unique ID makes it possible to assign a specific pointing device to a specific tablet. You can also use it to "sign" documents or to restrict access to document layers to a specific pointing device. This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- vendorDefined` (page 37)
`- vendorID` (page 37)

Declared In
`NSEvent.h`

userData

Returns data associated with a mouse-tracking event,

```
- (void *)userData
```

Discussion
The returned data was assigned to the mouse-tracking event when it was set up using the `NSView` method `addTrackingRect:owner:userData:assumeInside:`. It is only valid to send this message if the receiver represents an `NSMouseEntered` or `NSMouseExited` event. Raises an `NSInternalInconsistencyException` if sent to any other type of event object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 12)

Declared In
`NSEvent.h`

vendorDefined

Returns an array of three vendor-defined `NSNumber` objects associated with the pointing-type event represented by the receiver.

```
- (id)vendorDefined
```

Discussion
The `NSNumber` objects encapsulate `short` values that vendors may return for various reasons; see the vendor documentation for details.This method is valid for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSEvent.h`

vendorID

Returns the vendor identifier of the tablet associated with the receiver.

```
- (NSUInteger)vendorID
```

Discussion
The tablet is typically a USB device. This method is valid only for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `tabletID` (page 33)
- `vendorPointingDeviceType` (page 38)

Declared In
`NSEvent.h`

vendorPointingDeviceType

Returns a coded bit field whose set bits indicate the type of pointing device (within a vendor selection) associated with the receiver.

```
- (NSUInteger)vendorPointingDeviceType
```

Discussion
See the vendor documentation for an interpretation of significant bits. This method is valid only for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `vendorID` (page 37)

Declared In
`NSEvent.h`

window

Returns the window object associated with the receiver.

```
- (NSWindow *)window
```

Discussion
A periodic event, however, has no window; in this case the return value is undefined.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `windowNumber` (page 38)

Declared In
`NSEvent.h`

windowNumber

Returns the identifier for the window device associated with the receiver.

```
- (NSInteger)windowNumber
```

Discussion
A periodic event, however, has no window; in this case the return value is undefined.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- window` (page 38)

Declared In
`NSEvent.h`

Constants

NSEventType

These constants represent various kinds of events. They are returned by `type` (page 35) and are used as the first argument to the methods `enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 12), `keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 14), `mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:` (page 15), and `otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 17).

```
typedef enum _NSEventType {
    NSLeftMouseDown      = 1,
    NSLeftMouseUp        = 2,
    NSRightMouseDown     = 3,
    NSRightMouseUp       = 4,
    NSMouseMoved         = 5,
    NSLeftMouseDragged   = 6,
    NSRightMouseDragged  = 7,
    NSMouseEntered       = 8,
    NSMouseExited        = 9,
    NSKeyDown            = 10,
    NSKeyUp              = 11,
    NSFlagsChanged       = 12,
    NSAppKitDefined      = 13,
    NSSystemDefined      = 14,
    NSApplicationDefined = 15,
    NSPeriodic           = 16,
    NSCursorUpdate       = 17,
    NSScrollWheel        = 22,
    NSTabletPoint        = 23,
    NSTabletProximity    = 24,
    NSOtherMouseDown     = 25,
    NSOtherMouseUp       = 26,
    NSOtherMouseDragged  = 27
} NSEventType;
```

Constants

`NSLeftMouseDown`

See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftMouseUp`

See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseDown`

See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseUp`

See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSOtherMouseDown`

See “Mouse Events”.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSOtherMouseUp`
: See “Mouse Events”.

  Available in Mac OS X v10.1 and later.

  Declared in `NSEvent.h`.

`NSMouseMoved`
: See “Mouse Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSLeftMouseDragged`
: See “Mouse Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSRightMouseDragged`
: See “Mouse Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSOtherMouseDragged`
: See “Mouse Events”.

  Available in Mac OS X v10.1 and later.

  Declared in `NSEvent.h`.

`NSMouseEntered`
: See “Tracking-Rectangle and Cursor-Update Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSMouseExited`
: See “Tracking-Rectangle and Cursor-Update Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSCursorUpdate`
: See “Tracking-Rectangle and Cursor-Update Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSKeyDown`
: See “Keyboard Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSKeyUp`
: See “Keyboard Events”.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSFlagsChanged`
: See "Keyboard Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSAppKitDefined`
: See "Other Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSSystemDefined`
: See "Other Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSApplicationDefined`
: See "Other Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPeriodic`
: See "Periodic Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSScrollWheel`
: See "Mouse Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSTabletPoint`
: An event representing the current state of a tablet pointing device, including its location, pressure, and tilt.

  Available in Mac OS X v10.4 and later.

  Declared in `NSEvent.h`.

`NSTabletProximity`
: An event representing the proximity of a pointing device to its tablet.

  Available in Mac OS X v10.4 and later.

  Declared in `NSEvent.h`.

Declared In
`NSEvent.h`

Masks for event types

These constants are masks for the events defined in "NSEventType" (page 39). Pass them to the `NSCell` method `sendActionOn:` to specify when an `NSCell` should send its action message.

```
enum {
   NSLeftMouseDownMask      = 1 << NSLeftMouseDown,
   NSLeftMouseUpMask        = 1 << NSLeftMouseUp,
   NSRightMouseDownMask     = 1 << NSRightMouseDown,
   NSRightMouseUpMask       = 1 << NSRightMouseUp,
   NSMouseMovedMask         = 1 << NSMouseMoved,
   NSLeftMouseDraggedMask   = 1 << NSLeftMouseDragged,
   NSRightMouseDraggedMask  = 1 << NSRightMouseDragged,
   NSMouseEnteredMask       = 1 << NSMouseEntered,
   NSMouseExitedMask        = 1 << NSMouseExited,
   NSKeyDownMask            = 1 << NSKeyDown,
   NSKeyUpMask              = 1 << NSKeyUp,
   NSFlagsChangedMask       = 1 << NSFlagsChanged,
   NSAppKitDefinedMask      = 1 << NSAppKitDefined,
   NSSystemDefinedMask      = 1 << NSSystemDefined,
   NSApplicationDefinedMask = 1 << NSApplicationDefined,
   NSPeriodicMask           = 1 << NSPeriodic,
   NSCursorUpdateMask       = 1 << NSCursorUpdate,
   NSScrollWheelMask        = 1 << NSScrollWheel,
   NSOtherMouseDownMask     = 1 << NSOtherMouseDown,
   NSOtherMouseUpMask       = 1 << NSOtherMouseUp,
   NSOtherMouseDraggedMask  = 1 << NSOtherMouseDragged,
   NSAnyEventMask           = 0xffffffffU
};
```

Constants

`NSLeftMouseDownMask`

Corresponds to `NSLeftMouseDown`. See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftMouseUpMask`

Corresponds to `NSLeftMouseUp`. See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseDownMask`

Corresponds to `NSRightMouseDown`. See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseUpMask`

Corresponds to `NSRightMouseUp`. See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSOtherMouseDownMask`

Corresponds to `NSOtherMouseDown`. See “Mouse Events”.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSOtherMouseUpMask`

Corresponds to `NSOtherMouseUp`. See “Mouse Events”.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSMouseMovedMask`
Corresponds to `NSMouseMoved`. See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftMouseDraggedMask`
Corresponds to `NSLeftMouseDragged`. See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseDraggedMask`
Corresponds to `NSRightMouseDragged`. See “Mouse Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSOtherMouseDraggedMask`
Corresponds to `NSOtherMouseDragged`. See “Mouse Events”.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSMouseEnteredMask`
Corresponds to `NSMouseEntered`. See “Tracking-Rectangle and Cursor-Update Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSMouseExitedMask`
Corresponds to `NSMouseExited`. See “Tracking-Rectangle and Cursor-Update Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSCursorUpdateMask`
Corresponds to `NSCursorUpdate`. See “Tracking-Rectangle and Cursor-Update Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSKeyDownMask`
Corresponds to `NSKeyDown`. See “Keyboard Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSKeyUpMask`
Corresponds to `NSKeyUp`. See “Keyboard Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSFlagsChangedMask`
Corresponds to `NSFlagsChanged`. See “Keyboard Events”.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSAppKitDefinedMask`
: Corresponds to `NSAppKitDefined`. See "Other Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSSystemDefinedMask`
: Corresponds to `NSSystemDefined`. See "Other Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSApplicationDefinedMask`
: Corresponds to `NSApplicationDefined`. See "Other Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPeriodicMask`
: Corresponds to `NSPeriodic`. See "Periodic Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSScrollWheelMask`
: Corresponds to `NSScrollWheel`. See "Mouse Events".

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSTabletPointMask`
: Corresponds to `NSTabletPoint`.

  Available in Mac OS X v10.4 and later.

  Declared in `NSEvent.h`.

`NSTabletProximityMask`
: Corresponds to `NSTabletProximity`.

  Available in Mac OS X v10.4 and later.

  Declared in `NSEvent.h`.

`NSAnyEventMask`
: Corresponds to any of the above events.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

Declared In
`NSEvent.h`

Modifier Flags

The following constants (except for `NSDeviceIndependentModifierFlagsMask`) represent device-independent bits found in event modifier flags:

```
enum {
    NSAlphaShiftKeyMask = 1 << 16,
    NSShiftKeyMask      = 1 << 17,
    NSControlKeyMask    = 1 << 18,
    NSAlternateKeyMask  = 1 << 19,
    NSCommandKeyMask    = 1 << 20,
    NSNumericPadKeyMask = 1 << 21,
    NSHelpKeyMask       = 1 << 22,
    NSFunctionKeyMask   = 1 << 23,
    NSDeviceIndependentModifierFlagsMask = 0xffff0000U
};
```

Constants

`NSAlphaShiftKeyMask`

Set if Caps Lock key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSShiftKeyMask`

Set if Shift key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSControlKeyMask`

Set if Control key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSAlternateKeyMask`

Set if Option or Alternate key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSCommandKeyMask`

Set if Command key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSNumericPadKeyMask`

Set if any key in the numeric keypad is pressed. The numeric keypad is generally on the right side of the keyboard. This is also set if any of the arrow keys are pressed (`NSUpArrowFunctionKey` (page 52), `NSDownArrowFunctionKey` (page 52), `NSLeftArrowFunctionKey` (page 52), and `NSRightArrowFunctionKey` (page 52)).

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSHelpKeyMask`

Set if the Help key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSFunctionKeyMask`
Set if any function key is pressed. The function keys include the F keys at the top of most keyboards (F1, F2, and so on) and the navigation keys in the center of most keyboards (Help, Forward Delete, Home, End, Page Up, Page Down, and the arrow keys).

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSDeviceIndependentModifierFlagsMask`
Used to retrieve only the device-independent modifier flags, allowing applications to mask off the device-dependent modifier flags, including event coalescing information.

Available in Mac OS X v10.4.

Declared in `NSEvent.h`.

Declared In
`NSEvent.h`

NSPointingDeviceType

The following constants represent pointing-device types for `NSTabletProximity` events or mouse events with subtype `NSTabletProximityEventSubtype`. The `pointingDeviceType` (page 31) method returns one of these constants.

```
typedef enum {
    NSUnknownPointingDevice = NX_TABLET_POINTER_UNKNOWN,
    NSPenPointingDevice     = NX_TABLET_POINTER_PEN,
    NSCursorPointingDevice  = NX_TABLET_POINTER_CURSOR,
    NSEraserPointingDevice  = NX_TABLET_POINTER_ERASER
} NSPointingDeviceType;
```

Constants
`NSUnknownPointingDevice`
Represents an unknown type of pointing device.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSPenPointingDevice`
Represents the tip end of a stylus-like pointing device.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSCursorPointingDevice`
Represents a cursor (or puck-like) pointing device.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSEraserPointingDevice`
Represents the eraser end of a stylus-like pointing device.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

Declared In
`NSEvent.h`

Mouse-event subtypes

The following constants represent mouse-event subtypes for mouse and tablet events (accessed with the `subtype` (page 32) method).

```
enum {
    NSMouseEventSubtype           = NX_SUBTYPE_DEFAULT,
    NSTabletPointEventSubtype     = NX_SUBTYPE_TABLET_POINT,
    NSTabletProximityEventSubtype = NX_SUBTYPE_TABLET_PROXIMITY
};
```

Constants

`NSMouseEventSubtype`

Indicates a purely mouse event.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSTabletPointEventSubtype`

Indicates a tablet-pointer event; see description of `NSTabletPoint`.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSTabletProximityEventSubtype`

Indicates a tablet-proximity event; see description of `NSTabletProximity`.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

Declared In

`NSEvent.h`

Tablet event masks

The following constants represent button masks for `NSTabletPoint` events or mouse events with subtype `NSTabletPointEventSubtype`. The `buttonMask` (page 20) method returns a bit mask, which you test with one or more of these constants to determine the state of the buttons on a tablet pointing device.

```
enum {
    NSPenTipMask =       NX_TABLET_BUTTON_PENTIPMASK,
    NSPenLowerSideMask = NX_TABLET_BUTTON_PENLOWERSIDEMASK,
    NSPenUpperSideMask = NX_TABLET_BUTTON_PENUPPERSIDEMASK
};
```

Constants

`NSPenTipMask`

The pen tip is activated.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSPenLowerSideMask`

The button on the lower side of the device is activated.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSPenUpperSideMask`
: The button on the upper side of the device is activated.

  Available in Mac OS X v10.4 and later.

  Declared in `NSEvent.h`.

Declared In
`NSEvent.h`

Types Defined by the Application Kit

These constants represent the types of events defined by the Application Kit.

```
enum {
   NSWindowExposedEventType = 0,
   NSApplicationActivatedEventType = 1,
   NSApplicationDeactivatedEventType = 2,
   NSWindowMovedEventType = 4,
   NSScreenChangedEventType = 8,
   NSAWTEventType = 16
};
```

Constants
`NSWindowExposedEventType`
: A non-retained NSWindow has been exposed.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSApplicationActivatedEventType`
: The application has been activated.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSApplicationDeactivatedEventType`
: The application has been deactivated.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSWindowMovedEventType`
: An NSWindow has moved.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSScreenChangedEventType`
: An NSWindow has changed screens.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSAWTEventType`
: An event type used to support Java applications.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

Declared In
`NSEvent.h`

Power-off event

This constant denotes that the user is turning off the computer.

```
enum {
   NSPowerOffEventType = 1
};
```

Constants

`NSPowerOffEventType`
: Specifies that the user is turning off the computer.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

Declared In
`NSEvent.h`

Function-Key Unicodes

These constants represent Unicode characters (0xF700–0xF8FF) that are reserved for function keys on the keyboard. Combined in NSStrings, they are the return values of the NSEvent methods `characters` (page 22) and `charactersIgnoringModifiers` (page 22) and may be used in some parameters in the NSEvent method `keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 14).

```
enum {
   NSUpArrowFunctionKey = 0xF700,
   NSDownArrowFunctionKey = 0xF701,
   NSLeftArrowFunctionKey = 0xF702,
   NSRightArrowFunctionKey = 0xF703,
   NSF1FunctionKey = 0xF704,
   NSF2FunctionKey = 0xF705,
   NSF3FunctionKey = 0xF706,
   NSF4FunctionKey = 0xF707,
   NSF5FunctionKey = 0xF708,
   NSF6FunctionKey = 0xF709,
   NSF7FunctionKey = 0xF70A,
   NSF8FunctionKey = 0xF70B,
   NSF9FunctionKey = 0xF70C,
   NSF10FunctionKey = 0xF70D,
   NSF11FunctionKey = 0xF70E,
   NSF12FunctionKey = 0xF70F,
   NSF13FunctionKey = 0xF710,
   NSF14FunctionKey = 0xF711,
   NSF15FunctionKey = 0xF712,
   NSF16FunctionKey = 0xF713,
   NSF17FunctionKey = 0xF714,
   NSF18FunctionKey = 0xF715,
   NSF19FunctionKey = 0xF716,
   NSF20FunctionKey = 0xF717,
   NSF21FunctionKey = 0xF718,
   NSF22FunctionKey = 0xF719,
   NSF23FunctionKey = 0xF71A,
   NSF24FunctionKey = 0xF71B,
   NSF25FunctionKey = 0xF71C,
   NSF26FunctionKey = 0xF71D,
   NSF27FunctionKey = 0xF71E,
   NSF28FunctionKey = 0xF71F,
   NSF29FunctionKey = 0xF720,
   NSF30FunctionKey = 0xF721,
   NSF31FunctionKey = 0xF722,
   NSF32FunctionKey = 0xF723,
   NSF33FunctionKey = 0xF724,
   NSF34FunctionKey = 0xF725,
   NSF35FunctionKey = 0xF726,
   NSInsertFunctionKey = 0xF727,
   NSDeleteFunctionKey = 0xF728,
   NSHomeFunctionKey = 0xF729,
   NSBeginFunctionKey = 0xF72A,
   NSEndFunctionKey = 0xF72B,
   NSPageUpFunctionKey = 0xF72C,
   NSPageDownFunctionKey = 0xF72D,
   NSPrintScreenFunctionKey = 0xF72E,
   NSScrollLockFunctionKey = 0xF72F,
   NSPauseFunctionKey = 0xF730,
   NSSysReqFunctionKey = 0xF731,
   NSBreakFunctionKey = 0xF732,
   NSResetFunctionKey = 0xF733,
   NSStopFunctionKey = 0xF734,
   NSMenuFunctionKey = 0xF735,
   NSUserFunctionKey = 0xF736,
   NSSystemFunctionKey = 0xF737,
   NSPrintFunctionKey = 0xF738,
   NSClearLineFunctionKey = 0xF739,
   NSClearDisplayFunctionKey = 0xF73A,
   NSInsertLineFunctionKey = 0xF73B,
   NSDeleteLineFunctionKey = 0xF73C,
   NSInsertCharFunctionKey = 0xF73D,
   NSDeleteCharFunctionKey = 0xF73E,
   NSPrevFunctionKey = 0xF73F,
   NSNextFunctionKey = 0xF740,
   NSSelectFunctionKey = 0xF741,
   NSExecuteFunctionKey = 0xF742,
   NSUndoFunctionKey = 0xF743,
   NSRedoFunctionKey = 0xF744,
   NSFindFunctionKey = 0xF745,
   NSHelpFunctionKey = 0xF746,
   NSModeSwitchFunctionKey = 0xF747
};
```

Constants

`NSUpArrowFunctionKey`

Up Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSDownArrowFunctionKey`

Down Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftArrowFunctionKey`

Left Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightArrowFunctionKey`

Right Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF1FunctionKey`

F1 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF2FunctionKey`

F2 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF3FunctionKey`

F3 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF4FunctionKey`
: F4 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF5FunctionKey`
: F5 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF6FunctionKey`
: F6 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF7FunctionKey`
: F7 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF8FunctionKey`
: F8 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF9FunctionKey`
: F9 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF10FunctionKey`
: F10 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF11FunctionKey`
: F11 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF12FunctionKey`
: F12 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF13FunctionKey`
: F13 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF14FunctionKey`
: F14 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF15FunctionKey`
: F15 key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF16FunctionKey`
: F16 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF17FunctionKey`
: F17 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF18FunctionKey`
: F18 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF19FunctionKey`
: F19 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF20FunctionKey`
: F20 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF21FunctionKey`
: F21 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF22FunctionKey`
: F22 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF23FunctionKey`
: F23 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF24FunctionKey`
: F24 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF25FunctionKey`
: F25 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF26FunctionKey`
: F26 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF27FunctionKey`
: F27 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF28FunctionKey`
: F28 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF29FunctionKey`
: F29 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF30FunctionKey`
: F30 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF31FunctionKey`
: F31 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF32FunctionKey`
: F32 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF33FunctionKey`
: F33 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF34FunctionKey`
: F34 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSF35FunctionKey`
: F35 key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSInsertFunctionKey`
: Insert key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSDeleteFunctionKey`
: Forward Delete key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSHomeFunctionKey`
: Home key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSBeginFunctionKey`
: Begin key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSEndFunctionKey`
: End key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPageUpFunctionKey`
: Page Up key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPageDownFunctionKey`
: Page Down key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPrintScreenFunctionKey`
: Print Screen key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSScrollLockFunctionKey`
: Scroll Lock key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPauseFunctionKey`
: Pause key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSSysReqFunctionKey`
: System Request key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSBreakFunctionKey`
: Break key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSResetFunctionKey`
: Reset key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSStopFunctionKey`
: Stop key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSMenuFunctionKey`
: Menu key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSUserFunctionKey`
: User key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSSystemFunctionKey`
: System key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPrintFunctionKey`
: Print key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSClearLineFunctionKey`
: Clear/Num Lock key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSClearDisplayFunctionKey`
: Clear Display key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSInsertLineFunctionKey`
: Insert Line key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSDeleteLineFunctionKey`
: Delete Line key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSInsertCharFunctionKey`
: Insert Character key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSDeleteCharFunctionKey`
: Delete Character key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSPrevFunctionKey`
: Previous key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSNextFunctionKey`
: Next key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSSelectFunctionKey`
: Select key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSExecuteFunctionKey`
: Execute key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSUndoFunctionKey`
: Undo key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSRedoFunctionKey`
: Redo key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSFindFunctionKey`
: Find key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSHelpFunctionKey`
: Help key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSModeSwitchFunctionKey`
: Mode Switch key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

Discussion

Note that some function keys are handled at a lower level and are never seen by your application. They include the Volume Up key, Volume Down key, Volume Mute key, Eject key, and Function key found on many iBook and PowerBook computers.

Declared In

`NSEvent.h`

Document Revision History

This table describes the changes to *NSEvent Class Reference*.

Date	Notes
2009-02-04	Added descriptions of CGEvent and mouse-movement event coalescing methods.
2008-10-15	Documented four more methods for Leopard and made minor corrections.
2007-03-01	Updated for Mac OS X version 10.5.
2006-11-07	Made various minor corrections.
2006-05-23	First publication of this content as a separate document.

Index