Vanilla Documentation

Release 0.1

Tal Leming

Aug 31, 2017

Contents

1 Concepts 1

2 Objects 3

3 Indices and tables 59

Python Module Index 61

CHAPTER 1

Concepts

CHAPTER 2

Objects

Windows

Window

class vanilla.**Window**(*posSize*, *title=''*, *minSize=None*, *maxSize=None*, *textured=False*, *autosaveName=None*, *closable=True*, *miniaturizable=True*, *initiallyVisible=True*, *fullScreenMode=None*, *screen=None*)

A window capable of containing controls.

To add a control to a window, simply set it as an attribute of the window.:

```
from vanilla import *

class WindowDemo(object):

    def __init__(self):
        self.w = Window((200, 70), "Window Demo")
        self.w.myButton = Button((10, 10, -10, 20), "My Button")
        self.w.myTextBox = TextBox((10, 40, -10, 17), "My Text Box")
        self.w.open()

WindowDemo()
```

No special naming is required for the attributes. However, each attribute must have a unique name.

posSize Tuple of form *(left, top, width, height)* representing the position and size of the window. It may also be a tuple of form *(width, height)*. In this case, the window will be positioned on screen automatically.

title The title to be set in the title bar of the window.

minSize Tuple of the form *(width, height)* representing the minimum size that the window can be resized to.

maxSize Tuple of the form *(width, height)* representing the maximum size that the window can be resized to.

textured Boolean value representing if the window should have a textured appearance or not.

autosaveName A string representing a unique name for the window. If given, this name will be used to store the window position and size in the application preferences.

closable Boolean value representing if the window should have a close button in the title bar.

miniaturizable Boolean value representing if the window should have a minimize button in the title bar.

initiallyVisible Boolean value representing if the window will be initially visible. Default is *True*. If *False*, you can show the window later by calling *window.show()*.

fullScreenMode An indication of the full screen mode. These are the options: +————————+————————————————————————————+ | *None* | The window does not allow full screen. | +————————+————————————————————————————+ | *"primary"* | Corresponds to NSWindowCollectionBehaviorFullScreenPrimary. | +————————+————————————————————————————+ | *"auxiliary"* | Corresponds to NSWindowCollectionBehaviorFullScreenAuxiliary. | +————————+————————————————————————————+

screen A NSScreen object indicating the screen that the window should be drawn to. When None the window will be drawn to the main screen.

`Window.`**`assignToDocument`**`(`*document*`)`
Add this window to the list of windows associated with a document.

document should be a *NSDocument* instance.

`Window.`**`getNSWindow`**`()`
Return the *NSWindow* that this Vanilla object wraps.

`Window.`**`getNSWindowController`**`()`
Return an *NSWindowController* for the *NSWindow* that this Vanilla object wraps, creating a one if needed.

`Window.`**`open`**`()`
Open the window.

`Window.`**`close`**`()`
Close the window.

Once a window has been closed it can not be re-opened.

`Window.`**`hide`**`()`
Hide the window.

`Window.`**`show`**`()`
Show the window if it is hidden.

`Window.`**`makeKey`**`()`
Make the window the key window.

`Window.`**`makeMain`**`()`
Make the window the main window.

`Window.`**`setTitle`**`(`*title*`)`
Set the title in the window's title bar.

title shoud be a string.

`Window.`**`getTitle`**`()`
The title in the window's title bar.

`Window.`**`select`**`()`
Select the window if it is not the currently selected window.

`Window.`**`isVisible`**`()`
A boolean value representing if the window is visible or not.

`Window.`**`getPosSize`**`()`
A tuple of form *(left, top, width, height)* representing the window's position and size.

`Window.`**`setPosSize`**`(`*posSize*, *animate=True*`)`
Set the position and size of the window.

posSize A tuple of form *(left, top, width, height)*.

`Window.`**`center`**`()`
Center the window within the screen.

`Window.`**`move`**`(`*x*, *y*, *animate=True*`)`
Move the window by **x** units and **y** units.

`Window.`**`resize`**`(`*width*, *height*, *animate=True*`)`
Change the size of the window to **width** and **height**.

`Window.`**`setDefaultButton`**`(`*button*`)`
Set the default button in the window.

button will be bound to the Return and Enter keys.

`Window.`**`bind`**`(`*event*, *callback*`)`
Bind a callback to an event.

event A string representing the desired event. The options are:

"should close"	Called when the user attempts to close the window. This must return a bool indicating if the window should be closed or not.
"close"	Called immediately before the window closes.
"move"	Called immediately after the window is moved.
"resize"	Caled immediately after the window is resized.
"became main"	Called immediately after the window has become the main window.
"resigned main"	Called immediately after the window has lost its main window status.
"became key"	Called immediately after the window has become the key window.
"resigned key"	Called immediately after the window has lost its key window status.

For more information about main and key windows, refer to the Cocoa 'documentation <http://developer.apple.com/documentation/Cocoa/Conceptual/WinPanel/Concepts/ChangingMainKeyWindow.html>'_ on the subject.

callback The callback that will be called when the event occurs. It should accept a *sender* argument which will be the Window that called the callback.:

```
class WindowBindDemo(object):

    def __init__():
        self.w = Window((200, 200))
        self.w.bind("move", self.windowMoved)
        self.w.open()

    def windowMoved(sender):
        print "window moved!", sender

WindowBindDemo()
```

Window.**unbind**(*event*, *callback*)
Unbind a callback from an event.

event A string representing the desired event. Refer to *bind* for the options.

callback The callback that has been bound to the event.

Window.**addToolbar**(*toolbarIdentifier*, *toolbarItems*, *addStandardItems=True*)
Add a toolbar to the window.

toolbarIdentifier A string representing a unique name for the toolbar.

toolbarItems An ordered list of dictionaries containing the following items:

itemIdentifier	A unique string identifier for the item. This is only used internally.
label (optional)	The text label for the item. Defaults to *None*.
paletteLabel (optional)	The text label shown in the customization palette. Defaults to *label*.
toolTip (optional)	The tool tip for the item. Defaults to *label*.
imagePath (optional)	A file path to an image. Defaults to *None*.
imageNamed (optional)	The name of an image already loaded as a *NSImage* by the application. Defaults to *None*.
imageObject (optional)	A _NSImage_ object. Defaults to *None*.
selectable (optional)	A boolean representing if the item is selectable or not. The default value is _False_. For more information on selectable toolbar items, refer to Apple's documentation
view (optional)	A *NSView* object to be used instead of an image. Defaults to *None*.
visibleByDefault (optional)	If the item should be visible by default pass True to this argument. If the item should be added to the toolbar only through the customization palette, use a value of _False_. Defaults to _True_. |

addStandardItems A boolean, specifying whether the standard Cocoa toolbar items should be added. Defaults to *True*. If you set it to *False*, you must specify any standard items manually in *toolbarItems*, by using the constants from the AppKit module:

NSToolbarSeparatorItemIdentifier	The Separator item.
NSToolbarSpaceItemIdentifier	The Space item.
NSToolbarFlexibleSpaceItemIdentifier	The Flexible Space item.
NSToolbarShowColorsItemIdentifier	The Colors item. Shows the color panel.
NSToolbarShowFontsItemIdentifier	The Fonts item. Shows the font panel.
NSToolbarCustomizeToolbarItemIdentifier	The Customize item. Shows the customization palette.
NSToolbarPrintItemIdentifier	The Print item. Refer to Apple's *NSToolbarItem* documentation for more information.

Returns a dictionary containing the created toolbar items, mapped by itemIdentifier.

FloatingWindow

class vanilla.**FloatingWindow**(*posSize*, *title=''*, *minSize=None*, *maxSize=None*, *textured=False*, *autosaveName=None*, *closable=True*, *initiallyVisible=True*, *screen=None*)

A window that floats above all other windows.

To add a control to a window, simply set it as an attribute of the window.

```
from vanilla import *

class FloatingWindowDemo(object):

    def __init__(self):
        self.w = FloatingWindow((200, 70), "FloatingWindow Demo")
        self.w.myButton = Button((10, 10, -10, 20), "My Button")
        self.w.myTextBox = TextBox((10, 40, -10, 17), "My Text Box")
        self.w.open()

FloatingWindowDemo()
```

No special naming is required for the attributes. However, each attribute must have a unique name.

posSize Tuple of form *(left, top, width, height)* representing the position and size of the window. It may also be a tuple of form *(width, height)*. In this case, the window will be positioned on screen automatically.

title The title to be set in the title bar of the window.

minSize Tuple of the form *(width, height)* representing the minimum size that the window can be resized to.

maxSize Tuple of the form *(width, height)* representing the maximum size that the window can be resized to.

textured Boolean value representing if the window should have a textured appearance or not.

autosaveName A string representing a unique name for the window. If given, this name will be used to store the window position and size in the application preferences.

closable Boolean value representing if the window should have a close button in the title bar.

screen A NSScreen object indicating the screen that the window should be drawn to. When None the window will be drawn to the main screen.

FloatingWindow.**assignToDocument**(*document*)

Add this window to the list of windows associated with a document.

document should be a *NSDocument* instance.

FloatingWindow.**getNSWindow**()

Return the *NSWindow* that this Vanilla object wraps.

FloatingWindow.**getNSWindowController**()

Return an *NSWindowController* for the *NSWindow* that this Vanilla object wraps, creating a one if needed.

FloatingWindow.**open**()

Open the window.

FloatingWindow.**close**()

Close the window.

Once a window has been closed it can not be re-opened.

FloatingWindow.**hide**()

Hide the window.

FloatingWindow.**show**()

Show the window if it is hidden.

FloatingWindow.**makeKey**()
Make the window the key window.

FloatingWindow.**makeMain**()
Make the window the main window.

FloatingWindow.**setTitle**(*title*)
Set the title in the window's title bar.

title shoud be a string.

FloatingWindow.**getTitle**()
The title in the window's title bar.

FloatingWindow.**select**()
Select the window if it is not the currently selected window.

FloatingWindow.**isVisible**()
A boolean value representing if the window is visible or not.

FloatingWindow.**getPosSize**()
A tuple of form *(left, top, width, height)* representing the window's position and size.

FloatingWindow.**setPosSize**(*posSize*, *animate=True*)
Set the position and size of the window.

posSize A tuple of form *(left, top, width, height)*.

FloatingWindow.**center**()
Center the window within the screen.

FloatingWindow.**move**(*x*, *y*, *animate=True*)
Move the window by **x** units and **y** units.

FloatingWindow.**resize**(*width*, *height*, *animate=True*)
Change the size of the window to **width** and **height**.

FloatingWindow.**setDefaultButton**(*button*)
Set the default button in the window.

button will be bound to the Return and Enter keys.

FloatingWindow.**bind**(*event*, *callback*)
Bind a callback to an event.

event A string representing the desired event. The options are:

"should close"	Called when the user attempts to close the window. This must return a bool indicating if the window should be closed or not.
"close"	Called immediately before the window closes.
"move"	Called immediately after the window is moved.
"resize"	Caled immediately after the window is resized.
"became main"	Called immediately after the window has become the main window.
"resigned main"	Called immediately after the window has lost its main window status.
"became key"	Called immediately after the window has become the key window.
"resigned key"	Called immediately after the window has lost its key window status.

For more information about main and key windows, refer to the Cocoa 'documentation <http://developer.apple.com/documentation/Cocoa/Conceptual/WinPanel/Concepts/ChangingMainKeyWindow.html>'_ on the subject.

callback The callback that will be called when the event occurs. It should accept a *sender* argument which will be the Window that called the callback.:

```
class WindowBindDemo(object):

    def __init__():
        self.w = Window((200, 200))
        self.w.bind("move", self.windowMoved)
        self.w.open()

    def windowMoved(sender):
        print "window moved!", sender

WindowBindDemo()
```

`FloatingWindow.`**`unbind`**`(`*event*, *callback*`)`

Unbind a callback from an event.

event A string representing the desired event. Refer to *bind* for the options.

callback The callback that has been bound to the event.

`FloatingWindow.`**`addToolbar`**`(`*toolbarIdentifier*, *toolbarItems*, *addStandardItems=True*`)`

Add a toolbar to the window.

toolbarIdentifier A string representing a unique name for the toolbar.

toolbarItems An ordered list of dictionaries containing the following items:

itemIdentifier	A unique string identifier for the item. This is only used internally.
label (optional)	The text label for the item. Defaults to *None*.
paletteLabel (optional)	The text label shown in the customization palette. Defaults to *label*.
toolTip (optional)	The tool tip for the item. Defaults to *label*.
imagePath (optional)	A file path to an image. Defaults to *None*.
imageNamed (optional)	The name of an image already loaded as a *NSImage* by the application. Defaults to *None*.
imageObject (optional)	A _NSImage_ object. Defaults to *None*.
selectable (optional)	A boolean representing if the item is selectable or not. The default value is _False_. For more information on selectable toolbar items, refer to Apple's documentation
view (optional)	A *NSView* object to be used instead of an image. Defaults to *None*.
visibleByDefault (optional)	If the item should be visible by default pass True to this argument. If the item should be added to the toolbar only through the customization palette, use a value of _False_. Defaults to _True_. |

addStandardItems A boolean, specifying whether the standard Cocoa toolbar items should be added. Defaults to *True*. If you set it to *False*, you must specify any standard items manually in *toolbarItems*, by using the constants from the AppKit module:

NSToolbarSeparatorItemIdentifier	The Separator item.
NSToolbarSpaceItemIdentifier	The Space item.
NSToolbarFlexibleSpaceItemIdentifier	The Flexible Space item.
NSToolbarShowColorsItemIdentifier	The Colors item. Shows the color panel.
NSToolbarShowFontsItemIdentifier	The Fonts item. Shows the font panel.
NSToolbarCustomizeToolbarItemIdentifier	The Customize item. Shows the customization palette.
NSToolbarPrintItemIdentifier	The Print item. Refer to Apple's *NSToolbarItem* documentation for more information.

Returns a dictionary containing the created toolbar items, mapped by itemIdentifier.

Sheet

class vanilla.**Sheet**(*posSize*, *parentWindow*, *minSize=None*, *maxSize=None*, *autosaveName=None*)

A window that is attached to another window.

To add a control to a sheet, simply set it as an attribute of the sheet.:

```
from vanilla import *

class SheetDemo(object):

    def __init__(self, parentWindow):
        self.w = Sheet((200, 70), parentWindow)
        self.w.myButton = Button((10, 10, -10, 20), "My Button")
        self.w.myTextBox = TextBox((10, 40, -10, 17), "My Text Box")
        self.w.open()

SheetDemo()
```

No special naming is required for the attributes. However, each attribute must have a unique name.

posSize Tuple of form *(width, height)* representing the size of the sheet.

parentWindow The window that the sheet should be attached to.

minSize Tuple of the form *(width, height)* representing the minimum size that the sheet can be resized to.

maxSize Tuple of the form *(width, height)* representing the maximum size that the sheet can be resized to.

autosaveName A string representing a unique name for the sheet. If given, this name will be used to store the sheet size in the application preferences.

Sheet.**assignToDocument**(*document*)

Add this window to the list of windows associated with a document.

document should be a *NSDocument* instance.

Sheet.**getNSWindow**()

Return the *NSWindow* that this Vanilla object wraps.

Sheet.**getNSWindowController**()

Return an *NSWindowController* for the *NSWindow* that this Vanilla object wraps, creating a one if needed.

Sheet.**open**()

Open the window.

Sheet.**close**()
Close the window.

Once a window has been closed it can not be re-opened.

Sheet.**hide**()
Hide the window.

Sheet.**show**()
Show the window if it is hidden.

Sheet.**makeKey**()
Make the window the key window.

Sheet.**makeMain**()
Make the window the main window.

Sheet.**setTitle**(*title*)
Set the title in the window's title bar.

title shoud be a string.

Sheet.**getTitle**()
The title in the window's title bar.

Sheet.**select**()
Select the window if it is not the currently selected window.

Sheet.**isVisible**()
A boolean value representing if the window is visible or not.

Sheet.**getPosSize**()
A tuple of form *(left, top, width, height)* representing the window's position and size.

Sheet.**setPosSize**(*posSize*, *animate=True*)
Set the position and size of the window.

posSize A tuple of form *(left, top, width, height)*.

Sheet.**center**()
Center the window within the screen.

Sheet.**move**(*x*, *y*, *animate=True*)
Move the window by **x** units and **y** units.

Sheet.**resize**(*width*, *height*, *animate=True*)
Change the size of the window to **width** and **height**.

Sheet.**setDefaultButton**(*button*)
Set the default button in the window.

button will be bound to the Return and Enter keys.

Sheet.**bind**(*event*, *callback*)
Bind a callback to an event.

event A string representing the desired event. The options are:

"should close"	Called when the user attempts to close the window. This must return a bool indicating if the window should be closed or not.
"close"	Called immediately before the window closes.
"move"	Called immediately after the window is moved.
"resize"	Caled immediately after the window is resized.
"became main"	Called immediately after the window has become the main window.
"resigned main"	Called immediately after the window has lost its main window status.
"became key"	Called immediately after the window has become the key window.
"resigned key"	Called immediately after the window has lost its key window status.

For more information about main and key windows, refer to the Cocoa 'documentation <http://developer.apple.com/documentation/Cocoa/Conceptual/WinPanel/Concepts/ChangingMainKeyWindow.html>'_ on the subject.

callback The callback that will be called when the event occurs. It should accept a *sender* argument which will be the Window that called the callback.:

```
class WindowBindDemo(object):

    def __init__():
        self.w = Window((200, 200))
        self.w.bind("move", self.windowMoved)
        self.w.open()

    def windowMoved(sender):
        print "window moved!", sender

WindowBindDemo()
```

`Sheet.`**`unbind`**(*event*, *callback*)

Unbind a callback from an event.

event A string representing the desired event. Refer to *bind* for the options.

callback The callback that has been bound to the event.

`Sheet.`**`addToolbar`**(*toolbarIdentifier*, *toolbarItems*, *addStandardItems=True*)

Add a toolbar to the window.

toolbarIdentifier A string representing a unique name for the toolbar.

toolbarItems An ordered list of dictionaries containing the following items:

itemIdentifier	A unique string identifier for the item. This is only used internally.
label (optional)	The text label for the item. Defaults to *None*.
paletteLabel (optional)	The text label shown in the customization palette. Defaults to *label*.
toolTip (optional)	The tool tip for the item. Defaults to *label*.
imagePath (optional)	A file path to an image. Defaults to *None*.
imageNamed (optional)	The name of an image already loaded as a *NSImage* by the application. Defaults to *None*.
imageObject (optional)	A _NSImage_ object. Defaults to *None*.
selectable (optional)	A boolean representing if the item is selectable or not. The default value is _False_. For more information on selectable toolbar items, refer to Apple's documentation
view (optional)	A *NSView* object to be used instead of an image. Defaults to *None*.
visibleByDefault (optional)	If the item should be visible by default pass True to this argument. If the item should be added to the toolbar only through the customization palette, use a value of _False_. Defaults to _True_. |

addStandardItems A boolean, specifying whether the standard Cocoa toolbar items should be added. Defaults to *True*. If you set it to *False*, you must specify any standard items manually in *toolbarItems*, by using the constants from the AppKit module:

NSToolbarSeparatorItemIdentifier	The Separator item.
NSToolbarSpaceItemIdentifier	The Space item.
NSToolbarFlexibleSpaceItemIdentifier	The Flexible Space item.
NSToolbarShowColorsItemIdentifier	The Colors item. Shows the color panel.
NSToolbarShowFontsItemIdentifier	The Fonts item. Shows the font panel.
NSToolbarCustomizeToolbarItemIdentifier	The Customize item. Shows the customization palette.
NSToolbarPrintItemIdentifier	The Print item. Refer to Apple's *NSToolbarItem* documentation for more information.

Returns a dictionary containing the created toolbar items, mapped by itemIdentifier.

Drawer

class vanilla.**Drawer**(*size*, *parentWindow*, *minSize=None*, *maxSize=None*, *preferredEdge='left'*, *forceEdge=False*, *leadingOffset=20*, *trailingOffset=20*)

A drawer attached to a window. Drawers are capable of containing controls.

To add a control to a drawer, simply set it as an attribute of the drawer.:

```
from vanilla import *

class DrawerDemo(object):

    def __init__(self):
        self.w = Window((200, 200))
        self.w.button = Button((10, 10, -10, 20), "Toggle Drawer",
                               callback=self.toggleDrawer)
```

```
        self.d = Drawer((100, 150), self.w)
        self.d.textBox = TextBox((10, 10, -10, -10),
                                 "This is a drawer.")
        self.w.open()
        self.d.open()

    def toggleDrawer(self, sender):
        self.d.toggle()

DrawerDemo()
```

No special naming is required for the attributes. However, each attribute must have a unique name.

size Tuple of form *(width, height)* representing the size of the drawer.

parentWindow The window that the drawer should be attached to.

minSize Tuple of form *(width, height)* representing the minimum size of the drawer.

maxSize Tuple of form *(width, height)* representing the maximum size of the drawer.

preferredEdge The preferred edge of the window that the drawe should be attached to. If the drawer cannot be opened on the preferred edge, it will be opened on the opposite edge. The options are:

"left"
"right"
"top"
"bottom"

forceEdge Boolean representing if the drawer should *always* be opened on the preferred edge.

leadingOffset Distance between the top or left edge of the drawer and the parent window.

trailingOffset Distance between the bottom or right edge of the drawer and the parent window.

close()
Close the drawer.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSDrawer()
Return the *NSDrawer* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

isOpen()
Return True if the drawer is open, False if it is closed.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

open()
Open the drawer.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

toggle()
Open the drawer if it is closed. Close it if it is open.

Layout Views

Group

class vanilla.**Group**(*posSize*)
An invisible container for controls.

To add a control to a group, simply set it as an attribute of the group.:

```
from vanilla import *

class GroupDemo(object):

    def __init__(self):
        self.w = Window((150, 50))
        self.w.group = Group((10, 10, -10, -10))
        self.w.group.text = TextBox((0, 0, -0, -0),
                                    "This is a group")
        self.w.open()

GroupDemo()
```

No special naming is required for the attributes. However, each attribute must have a unique name.

posSize Tuple of form *(left, top, width, height)* representing the position and size of the group.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSView()
Return the *NSView* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

ScrollView

class vanilla.**ScrollView**(*posSize*, *nsView*, *hasHorizontalScroller=True*, *hasVerticalScroller=True*, *autohidesScrollers=False*, *backgroundColor=None*, *clipView=None*, *drawsBackground=True*)
A view with scrollers for containing another view.:

```
from AppKit import NSView, NSColor, NSRectFill
from vanilla import *

class DemoView(NSView):

    def drawRect_(self, rect):
        NSColor.redColor().set()
        NSRectFill(self.bounds())


class ScrollViewDemo(object):

    def __init__(self):
        self.w = Window((200, 200))
        self.view = DemoView.alloc().init()
        self.view.setFrame_(((0, 0), (300, 300)))
        self.w.scrollView = ScrollView((10, 10, -10, -10),
                                       self.view)
        self.w.open()

ScrollViewDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the scroll view.

nsView A *NSView* object.

hasHorizontalScroller Boolean representing if the scroll view has horizontal scrollers.

hasVerticalScroller Boolean representing if the scroll view has vertical scrollers.

autohidesScrollers Boolean representing if the scroll view auto-hides its scrollers.

backgroundColor A *NSColor* object representing the background color of the scroll view.

drawsBackground Boolean representing if the background should be drawn.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSScrollView()
Return the *NSScrollView* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setBackgroundColor(*color*)
Set the background of the scrol view to _color_.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

SplitView

vanilla.**SplitView**
alias of _NoRBSplitView

Box

class vanilla.**Box**(*posSize*, *title=None*)
A bordered container for other controls.

To add a control to a box, simply set it as an attribute of the box.:

```
from vanilla import *

class BoxDemo(object):

    def __init__(self):
        self.w = Window((150, 70))
        self.w.box = Box((10, 10, -10, -10))
        self.w.box.text = TextBox((10, 10, -10, -10), "This is a box")
        self.w.open()

BoxDemo()
```

No special naming is required for the attributes. However, each attribute must have a unique name.

posSize Tuple of form *(left, top, width, height)* representing the position and size of the box.

title The title to be displayed dabove the box. Pass *None* if no title is desired.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSBox()
Return the *NSBox* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the title of the box.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the title of the box.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

HorizontalLine

class vanilla.**HorizontalLine**(*posSize*)
A horizontal line.:

```
from vanilla import *

class HorizontalLineDemo(object):

    def __init__(self):
        self.w = Window((100, 20))
        self.w.line = HorizontalLine((10, 10, -10, 1))
        self.w.open()

HorizontalLineDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the line.

Standard Dimensions	
H	1

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSBox()
Return the *NSBox* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the title of the box.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the title of the box.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

VerticalLine

class vanilla.**VerticalLine**(*posSize*)
A vertical line.:

```
from vanilla import *

class VerticalLineDemo(object):

    def __init__(self):
        self.w = Window((80, 100))
        self.w.line = VerticalLine((40, 10, 1, -10))
        self.w.open()

VerticalLineDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the line.

Standard Dimensions	
V	1

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSBox()
Return the *NSBox* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the title of the box.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the title of the box.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

Data Views

List

class vanilla.**List**(*posSize*, *items*, *dataSource=None*, *columnDescriptions=None*, *showColumnTitles=True*, *selectionCallback=None*, *doubleClickCallback=None*, *editCallback=None*, *enableDelete=False*, *enableTypingSensitivity=False*, *allowsMultipleSelection=True*, *allowsEmptySelection=True*, *drawVerticalLines=False*, *drawHorizontalLines=False*, *autohidesScrollers=True*, *drawFocusRing=True*, *rowHeight=17.0*, *selfDropSettings=None*, *selfWindowDropSettings=None*, *selfDocumentDropSettings=None*, *selfApplicationDropSettings=None*, *otherApplicationDropSettings=None*, *dragSettings=None*)
A control that shows a list of items. These lists can contain one or more columns.

A single column example:

```
from vanilla import *

class ListDemo(object):

    def __init__(self):
        self.w = Window((100, 100))
        self.w.myList = List((0, 0, -0, -0), ["A", "B", "C"],
                    selectionCallback=self.selectionCallback)
        self.w.open()

    def selectionCallback(self, sender):
        print sender.getSelection()

ListDemo()
```

A mutliple column example:

```
from vanilla import *

class ListDemo(object):

    def __init__(self):
        self.w = Window((100, 100))
        self.w.myList = List((0, 0, -0, -0),
                    [{"One": "A", "Two": "a"}, {"One": "B", "Two": "b"}],
                    columnDescriptions=[{"title": "One"}, {"title": "Two"}],
                    selectionCallback=self.selectionCallback)
        self.w.open()

    def selectionCallback(self, sender):
        print sender.getSelection()
```

```
ListDemo()
```

List objects behave like standard Python lists. For xample, given this List::

```
self.w.myList = List((10, 10, 200, 100), ["A", "B", "C"])
```

The following Python list methods work::

```
# Getting the length of the List.
>>> len(self.w.myList)
3

# Retrieving an item or items from a List.
>>> self.w.myList[1]
"B"
>>> self.w.myList[:2]
["A", "B"]

# Setting an item in a List.
>>> self.w.myList[1] = "XYZ"
>>> self.w.myList.get()
["A", "XYZ", "C"]

# Deleting an item at an index in a List.
>>> del self.w.myList[1]
>>> self.w.myList.get()
["A", "C"]

# Appending an item to a List.
>>> self.w.myList.append("Z")
>>> self.w.myList.get()
["A", "B", "C", "Z"]

# Removing the first occurance of an item in a List.
>>> self.w.myList.remove("A")
>>> self.w.myList.get()
["B", "C"]

# Getting the index for the first occurance of an item in a List.
>>> self.w.myList.index("B")
1

# Inserting an item into a List.
>>> self.w.myList.insert(1, "XYZ")
>>> self.w.myList.get()
["A", "XYZ", "B", "C"]

# Extending a List.
>>> self.w.myList.extend(["X", "Y", "Z"])
>>> self.w.myList.get()
["A", "B", "C", "X", "Y", "Z"]

# Iterating over a List.
>>> for i in self.w.myList:
>>>     i
"A"
"B"
```

```
"C"
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the list.

items The items to be displayed in the list. In the case of multiple column lists, this should be a list of dictionaries with the data for each column keyed by the column key as defined in columnDescriptions. If you intend to use a dataSource, *items* must be *None*.

dataSource A Cocoa object supporting the *NSTableDataSource* protocol. If *dataSource* is given, *items* must be *None*.

columnDescriptions An ordered list of dictionaries describing the columns. This is only necessary for multiple column lists.

"title"	The title to appear in the column header.
"key" (optional)	The key from which this column should get its data from each dictionary in *items*. If nothing is given, the key will be the string given in *title*.
"formatter" (optional)	An *NSFormatter* <http://tinyurl.com/NSFormatter>‘_ for cntrolling the display and input of the column’s cells.
"cell" (optional)	A cell type to be displayed in the column. If nothing is given, a text cell is used.
"editable" (optional)	Enable or disable editing in the column. If nothing is given, it will follow the editability of the rest of the list.
"width" (optional)	The width of the column. In OS 10.3 and lower the width must be defined for *all* columns if the width is defined for one column.
"typingSensitive" (optional)	A boolean representing that this column should be the column that responds to user key input. Only one column can be flagged as True. If no column is flagged, the first column will automatically be flagged.
binding (optional)	A string indicating which binding object the column’s cell should be bound to. By default, this is “value.” You should only override this in very specific cases.

showColumnTitles Boolean representing if the column titles should be shown or not. Column titles will not be shown in single column lists.

selectionCallback Callback to be called when the selection in the list changes.

doubleClickCallback Callback to be called when an item is double clicked.

editCallback Callback to be called after an item has been edited.

enableDelete A boolean representing if items in the list can be deleted via the interface.

enableTypingSensitivity A boolean representing if typing in the list will jump to the closest match as the entered keystrokes. *Available only in single column lists.*

allowsMultipleSelection A boolean representing if the list allows more than one item to be selected.

allowsEmptySelection A boolean representing if the list allows zero items to be selected.

drawVerticalLines Boolean representing if vertical lines should be drawn in the list.

drawHorizontalLines Boolean representing if horizontal lines should be drawn in the list.

drawFocusRing Boolean representing if the standard focus ring should be drawn when the list is selected.

rowHeight The height of the rows in the list.

autohidesScrollers Boolean representing if scrollbars should automatically be hidden if possible.

selfDropSettings A dictionary defining the drop settings when the source of the drop is this list. The dictionary form is described below.

selfWindowDropSettings A dictionary defining the drop settings when the source of the drop is contained the same document as this list. The dictionary form is described below.

selfDocumentDropSettings A dictionary defining the drop settings when the source of the drop is contained the same window as this list. The dictionary form is described below.

selfApplicationDropSettings A dictionary defining the drop settings when the source of the drop is contained the same application as this list. The dictionary form is described below.

otherApplicationDropSettings A dictionary defining the drop settings when the source of the drop is contained an application other than the one that contains this list. The dictionary form is described below.

The drop settings dictionaries should be of this form:

type	A single drop type indicating what drop types the list accepts. For example, NSFilenamesPboardType or "MyCustomPboardType".
operation (optional)	A "drag operation <http://tinyurl.com/NSDraggingIn>'_ that the list accepts. The default is *NSDragOperationCopy*.
allowDropBetweenRows (optional)	A boolean indicating if the list accepts drops between rows. The default is True.
allowDropOnRow (optional)	A boolean indicating if the list accepts drops on rows. The default is False.
callback	Callback to be called when a drop is proposed and when a drop is to occur. This method should return a boolean representing if the drop is acceptable or not. This method must accept *sender* and *dropInfo* arguments. The _dropInfo_ will be a dictionary as described below.

The dropInfo dictionary passed to drop callbacks will be of this form:

data	The data proposed for the drop. This data will be of the type specified by dropDataFormat.
rowIndex	The row where the drop is proposed.
source	The source from which items are being dragged. If this object is wrapped by Vanilla, the Vanilla object will be passed as the source.
dropOnRow	A boolean representing if the row is being dropped on. If this is False, the drop should occur between rows.
isProposal	A boolean representing if this call is simply proposing the drop or if it is time to accept the drop.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the list of items in the list.

getNSScrollView()
Return the *NSScrollView* that this object wraps.

getNSTableView()
Return the *NSTableView* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getSelection()
Get a list of indexes of selected items in the list.

isVisible()
Return a bool indicting if the object is visible or not.

move (*x*, *y*)
Move the object by **x** units and **y** units.

resize (*width*, *height*)
Change the size of the object to **width** and **height**.

scrollToSelection ()
Scroll the selected rows to visible.

set (*items*)
Set the items in the list.

items should follow the same format as described in the constructor.

setPosSize (*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setSelection (*selection*)
Set the selected items in the list.

selection should be a list of indexes.

show (*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

List Item Cells

vanilla.**LevelIndicatorListCell** (*style='discrete'*, *minValue=0*, *maxValue=10*, *warningValue=None*, *criticalValue=None*, *imagePath=None*, *imageNamed=None*, *imageObject=None*)
An object that displays a level indicator in a List column.

This object should only be used in the *columnDescriptions* argument during the construction of a List.:

```
from vanilla import *

class LevelIndicatorListCellDemo(object):

    def __init__(self):
        self.w = Window((340, 140))
        items = [
            {"discrete": 3, "continuous": 4, "rating": 1, "relevancy": 9},
            {"discrete": 8, "continuous": 3, "rating": 5, "relevancy": 5},
            {"discrete": 3, "continuous": 7, "rating": 3, "relevancy": 4},
            {"discrete": 2, "continuous": 5, "rating": 4, "relevancy": 7},
            {"discrete": 6, "continuous": 9, "rating": 3, "relevancy": 2},
            {"discrete": 4, "continuous": 0, "rating": 6, "relevancy": 8},
        ]
        columnDescriptions = [
            {"title": "discrete",
            "cell": LevelIndicatorListCell(style="discrete", warningValue=7,␣
↪criticalValue=9)},
            {"title": "continuous",
            "cell": LevelIndicatorListCell(style="continuous", warningValue=7,␣
↪criticalValue=9)},
            {"title": "rating",
            "cell": LevelIndicatorListCell(style="rating", maxValue=6)},
```

```
            {"title": "relevancy",
            "cell": LevelIndicatorListCell(style="relevancy")},
        ]
        self.w.list = List((0, 0, -0, -0), items=items,
                           columnDescriptions=columnDescriptions)
        self.w.open()

LevelIndicatorListCellDemo()
```

style The style of the level indicator. The options are:

"continuous"	A continuous bar.
"discrete"	A segmented bar.
"rating"	A row of stars. Similar to the rating indicator in iTunes.
"relevancy"	A row of lines. Similar to the search result relevancy indicator in Mail.

minValue The minimum value allowed by the level indicator.

maxValue The maximum value allowed by the level indicator.

warningValue The value at which the filled portions of the level indicator should display the warning color. Applies only to discrete and continuous level indicators.

criticalValue The value at which the filled portions of the level indicator should display the critical color. Applies only to discrete and continuous level indicators.

vanilla.**CheckBoxListCell**(*title=None*)
An object that displays a check box in a List column.

This object should only be used in the *columnDescriptions* argument during the construction of a List.

title The title to be set in *all* items in the List column.

vanilla.**SliderListCell**(*minValue=0*, *maxValue=100*)
An object that displays a slider in a List column.

This object should only be used in the *columnDescriptions* argument during the construction of a List.

minValue The minimum value for the slider.

maxValue The maximum value for the slider.

ImageView

class vanilla.**ImageView**(*posSize*, *horizontalAlignment='center'*, *verticalAlignment='center'*, *scale='proportional'*)
A view that displays an image.

posSize Tuple of form *(left, top, width, height)* representing the position and size of the view.

horizontalAlignment A string representing the desired horizontal alignment of the image in the view. The options are:

"left"	Image is aligned left.
"right"	Image is aligned right.
"center"	Image is centered.

verticalAlignment A string representing the desired vertical alignment of the image in the view. The options are:

“top”	Image is aligned top.
“bottom”	Image is aligned bottom.
“center”	Image is centered.

scale A string representing the desired scale style of the image in the view. The options are:

“porportional”	Proportionally scale the image to fit in the view if it is larger than the view.
“fit”	Distort the proportions of the image until it fits exactly in the view.
“none”	Do not scale the image.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSImageView()
Return the *NSImageView* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setImage(*imagePath=None*, *imageNamed=None*, *imageObject=None*)
Set the image in the view.

imagePath A file path to an image.

imageNamed The name of an image already load as a *NSImage* by the application.

imageObject A *NSImage* object.

Only one of imagePath, imageNamed, imageObject should be set.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

LevelIndicator

class vanilla.**LevelIndicator**(*posSize*, *style='discrete'*, *value=5*, *minValue=0*, *maxValue=10*, *warningValue=None*, *criticalValue=None*, *tickMarkPosition=None*, *minorTickMarkCount=None*, *majorTickMarkCount=None*, *callback=None*)
A control which shows a value on a linear scale.:

```
from vanilla import *

class LevelIndicatorDemo(object):

    def __init__(self):
```

```
        self.w = Window((200, 68))
        self.w.discreteIndicator = LevelIndicator(
                    (10, 10, -10, 18), callback=self.levelIndicatorCallback)
        self.w.continuousIndicator = LevelIndicator(
                    (10, 40, -10, 18), style="continuous",
                    callback=self.levelIndicatorCallback)
        self.w.open()

    def levelIndicatorCallback(self, sender):
        print "level indicator edit!", sender.get()

LevelIndicatorDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the level indicator.

Standard Dimensions()
discrete without ticks
H	18
discrete with minor ticks
H	22
discrete with major ticks
H	25
continuous without ticks
H	16
continuous with minor ticks
H	20
continuous with major ticks
H	23

style The style of the level indicator. The options are:

"continuous"	A continuous bar.
"discrete"	A segmented bar.

value The initial value of the level indicator.

minValue The minimum value allowed by the level indicator.

maxValue The maximum value allowed by the level indicator.

warningValue The value at which the filled portions of the level indicator should display the warning color.

criticalValue The value at which the filled portions of the level indicator should display the critical color.

tickMarkPosition The position of the tick marks in relation to the level indicator. The options are:

"above"
"below"

minorTickMarkCount The number of minor tick marcks to be displayed on the level indicator. If *None* is given, no minor tick marks will be displayed.

majorTickMarkCount The number of major tick marcks to be displayed on the level indicator. If *None* is given, no major tick marks will be displayed.

callback The method to be called when the level indicator has been edited. If no callback is given, the level indicator will not be editable.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the value of the level indicator.

getCriticalValue(*value*)
Get the critical value of the level indicator.

getMaxValue()
Get the maximum of the level indicator.

getMinValue()
Get the minimum value of the level indicator.

getNSLevelIndicator()
Return the *NSLevelIndicator* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

getWarningValue(*value*)
Get the warning value of the level indicator.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the value of the level indicator.

setCriticalValue(*value*)
Set the critical value of the level indicator.

setMaxValue(*value*)
Set the maximum of the level indicator.

setMinValue(*value*)
Set the minimum value of the level indicator.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

setWarningValue(*value*)
Set the warning value of the level indicator.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

Buttons

Button

class vanilla.**Button**(*posSize*, *title*, *callback=None*, *sizeStyle='regular'*)

A standard button.:

```
from vanilla import *

class ButtonDemo(object):

    def __init__(self):
        self.w = Window((100, 40))
        self.w.button = Button((10, 10, -10, 20), "A Button",
                               callback=self.buttonCallback)
        self.w.open()

    def buttonCallback(self, sender):
        print "button hit!"

ButtonDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the button. The size of the button sould match the appropriate value for the given *sizeStyle*.

Standard Dimensions		
Regular	H	20
Small	H	17
Mini	H	14

title The text to be displayed on the button. Pass *None* is no title is desired.

callback The method to be called when the user presses the button.

sizeStyle A string representing the desired size style of the button. The options are:

"regular"
"small"
"mini"

bind(*key*, *modifiers*)

Bind a key to the button.

key A single character or one of the following:

"help"
"home"
"end"
"pageup"
"pagedown"
"forwarddelete"
"leftarrow"
"rightarrow"
"uparrow"
"downarrow"

modifiers A list containing nothing or as many of the following as desired:

“command”
“control”
“option”
“shift”
“capslock”

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSButton()
Return the *NSButton* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

SquareButton

class vanilla.**SquareButton**(*posSize*, *title*, *callback=None*, *sizeStyle='regular'*)
A standard square button.:

```
from vanilla import *

class SquareButtonDemo(object):

    def __init__(self):
        self.w = Window((200, 100))
        self.w.button = SquareButton((10, 10, -10, -10), "A Button",
                            callback=self.buttonCallback)
        self.w.open()

    def buttonCallback(self, sender):
        print "button hit!"

SquareButtonDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the button.

title The text to be displayed on the button. Pass _None_ is no title is desired.

callback The method to be called when the user presses the button.

sizeStyle A string representing the desired size style of the button. The options are:

"regular"
"small"
"mini"

bind(*key*, *modifiers*)

Bind a key to the button.

key A single character or one of the following:

"help"
"home"
"end"
"pageup"
"pagedown"
"forwarddelete"
"leftarrow"
"rightarrow"
"uparrow"
"downarrow"

modifiers A list containing nothing or as many of the following as desired:

"command"
"control"
"option"
"shift"
"capslock"

enable(*onOff*)

Enable or disable the object. **onOff** should be a boolean.

getNSButton()

Return the *NSButton* that this object wraps.

getPosSize()

The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()

Get the control title.

isVisible()

Return a bool indicting if the object is visible or not.

move(*x*, *y*)

Move the object by **x** units and **y** units.

resize(*width*, *height*)

Change the size of the object to **width** and **height**.

setPosSize(*posSize*)

Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

ImageButton

class vanilla.**ImageButton**(*posSize*, *imagePath=None*, *imageNamed=None*, *imageObject=None*, *title=None*, *bordered=True*, *imagePosition='top'*, *callback=None*, *sizeStyle='regular'*)
A button with an image.:

```
from vanilla import *

class ImageButtonDemo(object):

    def __init__(self):
        path = "/path/to/an/image"
        self.w = Window((50, 50))
        self.w.button = ImageButton((10, 10, 30, 30), imagePath=path,
                                    callback=self.buttonCallback)
        self.w.open()

    def buttonCallback(self, sender):
        print "button hit!"

ImageButtonDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the button.

title The text to be displayed on the button. Pass *None* is no title is desired.

bordered Boolean representing if the button should be bordered.

imagePath A file path to an image.

imageNamed The name of an image already load as a *NSImage* by the application.

imageObject A *NSImage* object.

Only one of imagePath, imageNamed, imageObject should be set.

imagePosition The position of the image relative to the title. The options are:

"top"
"bottom"
"left"
"right"

callback The method to be called when the user presses the button.

sizeStyle A string representing the desired size style of the button. The options are:

"regular"
"small"
"mini"

bind(*key*, *modifiers*)
Bind a key to the button.

key A single character or one of the following:

"help"
"home"
"end"
"pageup"
"pagedown"
"forwarddelete"
"leftarrow"
"rightarrow"
"uparrow"
"downarrow"

modifiers A list containing nothing or as many of the following as desired:

"command"
"control"
"option"
"shift"
"capslock"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSButton()
Return the *NSButton* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setImage(*imagePath=None*, *imageNamed=None*, *imageObject=None*)
Set the image in the button.

imagePath A file path to an image.

imageNamed The name of an image already load as a *NSImage* by the application.

imageObject A *NSImage* object.

Only one of imagePath, imageNamed, imageObject should be set.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

GradientButton

class vanilla.**GradientButton**(*posSize*, *imagePath=None*, *imageNamed=None*, *imageObject=None*, *title=None*, *bordered=True*, *imagePosition='top'*, *callback=None*, *sizeStyle='regular'*)

bind(*key*, *modifiers*)
Bind a key to the button.

key A single character or one of the following:

"help"
"home"
"end"
"pageup"
"pagedown"
"forwarddelete"
"leftarrow"
"rightarrow"
"uparrow"
"downarrow"

modifiers A list containing nothing or as many of the following as desired:

"command"
"control"
"option"
"shift"
"capslock"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSButton()
Return the *NSButton* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setImage(*imagePath=None*, *imageNamed=None*, *imageObject=None*)
Set the image in the button.

imagePath A file path to an image.

imageNamed The name of an image already load as a *NSImage* by the application.

imageObject A *NSImage* object.

Only one of imagePath, imageNamed, imageObject should be set.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

HelpButton

class vanilla.**HelpButton**(*posSize*, *callback=None*, *page=None*, *anchor=None*)
A standard help button.:

```
from vanilla import *

class HelpButtonDemo(object):

    def __init__(self):
        self.w = Window((90, 40))
        self.w.button = HelpButton((10, 10, 21, 20),
                            callback=self.buttonCallback)
        self.w.open()

    def buttonCallback(self, sender):
        print "help button hit!"

HelpButtonDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the button. The size of the button sould match the standard dimensions.

Standard Dimensions	
Width	21
Height	20

callback The method to be called when the user presses the button.

bind(*key*, *modifiers*)
Bind a key to the button.

key A single character or one of the following:

“help”
“home”
“end”
“pageup”
“pagedown”
“forwarddelete”
“leftarrow”
“rightarrow”
“uparrow”
“downarrow”

modifiers A list containing nothing or as many of the following as desired:

“command”
“control”
“option”
“shift”
“capslock”

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSButton()
Return the *NSButton* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

SegmentedButton

class vanilla.**SegmentedButton**(*posSize*, *segmentDescriptions*, *callback=None*, *selectionStyle='one'*, *sizeStyle='small'*)

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

Inputs

TextEditor

class vanilla.**TextEditor**(*posSize*, *text=''*, *callback=None*, *readOnly=False*, *checksSpelling=False*)
Standard long text entry control.:

```
from vanilla import *

class TextEditorDemo(object):

    def __init__(self):
        self.w = Window((200, 200))
        self.w.textEditor = TextEditor((10, 10, -10, 22),
                                callback=self.textEditorCallback)
        self.w.open()

    def textEditorCallback(self, sender):
        print "text entry!", sender.get()

TextEditorDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the text entry control.

text The text to be displayed in the text entry control.

callback The method to be called when the user presses the text entry control.

readOnly Boolean representing if the text can be edited or not.

checksSpelling Boolean representing if spelling should be automatically checked or not.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the contents of the text entry control.

getNSScrollView()
Return the *NSScrollView* that this object wraps.

getNSTextView()
Return the *NSTextView* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

selectAll()
Select all text in the text entry control.

set(*value*)
Set the contents of the text box.

value A string representing the contents of the text box.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

TextBox

class vanilla.**TextBox**(*posSize*, *text=''*, *alignment='natural'*, *selectable=False*, *sizeStyle='regular'*)
A rectangle containing static text.:

```
from vanilla import *

class TextBoxDemo(object):

    def __init__(self):
        self.w = Window((100, 37))
        self.w.textBox = TextBox((10, 10, -10, 17), "A TextBox")
        self.w.open()

TextBoxDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the text box.

Standard Dimensions		
Regular	H	17
Small	H	14
Mini	H	12

text The text to be displayed in the text box. If the object is a *NSAttributedString*, the attributes will be used for display.

alignment A string representing the desired visual alignment of the text in the text box. The options are:

"left"	Text is aligned left.
"right"	Text is aligned right.
"center"	Text is centered.
"justified"	Text is justified.
"natural"	Follows the natural alignment of the text's script.

selectable Boolean representing if the text is selectable or not.

sizeStyle A string representing the desired size style of the button. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the contents of the text box.

getNSTextField()
Return the *NSTextField* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the contents of the text box.

value A string representing the contents of the text box.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

EditText

class vanilla.**EditText**(*posSize*, *text=''*, *callback=None*, *continuous=True*, *readOnly=False*, *formatter=None*, *placeholder=None*, *sizeStyle='regular'*)
Standard short text entry control.:

```
from vanilla import *

class EditTextDemo(object):

    def __init__(self):
        self.w = Window((100, 42))
        self.w.editText = EditText((10, 10, -10, 22),
                            callback=self.editTextCallback)
        self.w.open()

    def editTextCallback(self, sender):
        print "text entry!", sender.get()

EditTextDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the text entry control.

Standard Dimensions		
Regular	H	22
Small	H	19
Mini	H	16

text An object representing the contents of the text entry control. If no formatter has been assigned to the control, this should be a string. If a formatter has been assigned, this should be an object of the type that the formatter expects.

callback The method to be called when the user enters text.

continuous If True, the callback (if any) will be called upon each keystroke, if False, only call the callback when editing finishes. Default is True.

readOnly Boolean representing if the text can be edited or not.

formatter An NSFormatter for controlling the display and input of the text entry.

placeholder A placeholder string to be shown when the text entry control is empty.

sizeStyle A string representing the desired size style of the text entry control. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the contents of the text entry control.

If no formatter has been assigned to the control, this returns a string. If a formatter has been assigned, this returns an object which has been translated by the formatter.

getNSTextField()
Return the *NSTextField* that this object wraps.

getPlaceholder()
Get the placeholder string displayed in the control.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

selectAll()
Select all text in the text entry control.

set(*value*)
Set the contents of the text entry control.

value An object representing the contents of the text entry control. If no formatter has been assigned to the control, this should be a string. If a formatter has been assigned, this should be an object of the type that the formatter expects.

setPlaceholder(*value*)
Set **value** as the placeholder string to be displayed in the control. **value** must be a string.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

SecureEditText

class vanilla.**SecureEditText**(*posSize*, *text=''*, *callback=None*, *continuous=True*, *readOnly=False*, *formatter=None*, *placeholder=None*, *sizeStyle='regular'*)
Standard secure text entry control.:

```
from vanilla import *

class SecureEditTextDemo(object):
```

```
    def __init__(self):
        self.w = Window((100, 42))
        self.w.secureEditText = SecureEditText((10, 10, -10, 22),
                              callback=self.secureEditTextCallback)
        self.w.open()

    def secureEditTextCallback(self, sender):
        print "text entry!", sender.get()

SecureEditTextDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the text entry control.

Standard Dimensions		
Regular	H	22
Small	H	19
Mini	H	16

text An object representing the contents of the text entry control. If no formatter has been assigned to the control, this should be a string. If a formatter has been assigned, this should be an object of the type that the formatter expects.

callback The method to be called when the user enters text.

continuous If True, the callback (if any) will be called upon each keystroke, if False, only call the callback when editing finishes. Default is True.

readOnly Boolean representing if the text can be edited or not.

formatter An NSFormatter for controlling the display and input of the text entry.

placeholder A placeholder string to be shown when the text entry control is empty.

sizeStyle A string representing the desired size style of the text entry control. The options are:

“regular”
“small”
“mini”

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the contents of the text entry control.

If no formatter has been assigned to the control, this returns a string. If a formatter has been assigned, this returns an object which has been translated by the formatter.

getNSSecureTextField()
Return the *NSSecureTextField* that this object wraps.

getNSTextField()
Return the *NSTextField* that this object wraps.

getPlaceholder()
Get the placeholder string displayed in the control.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

selectAll()
Select all text in the text entry control.

set(*value*)
Set the contents of the text entry control.

value An object representing the contents of the text entry control. If no formatter has been assigned to the control, this should be a string. If a formatter has been assigned, this should be an object of the type that the formatter expects.

setPlaceholder(*value*)
Set **value** as the placeholder string to be displayed in the control. **value** must be a string.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

DatePicker

class vanilla.**DatePicker**(*posSize*, *date=None*, *minDate=None*, *maxDate=None*, *showStepper=True*, *mode='text'*, *timeDisplay='hourMinuteSecond'*, *dateDisplay='yearMonthDay'*, *callback=None*, *sizeStyle='regular'*)
posSize Tuple of form *(left, top, width, height)* representing the position and size of the date picker control.

Standard Dimensions - Text Mode		
Regular	H	22
Small	H	19
Mini	H	16

Standard Dimensions - Graphical Mode	
Calendar and Clock	227w 148h
Calendar	139w 148h
Clock	122w 123h

date A *NSDate* object representing the date and time that should be set in the control.

minDate A *NSDate* object representing the lowest date and time that can be set in the control.

maxDate A *NSDate* object representing the highest date and time that can be set in the control.

showStepper A boolean indicating if the thumb stepper should be shown in text mode.

mode A string representing the desired mode for the date picker control. The options are:

"text"
"graphical"

timeDisplay A string representing the desired time units that should be displayed in the date picker control. The options are:

None	Do not display time.
"hourMinute"	Display hour and minute.
"hourMinuteSecond"	Display hour, minute, second.

dateDisplay A string representing the desired date units that should be displayed in the date picker control. The options are:

None	Do not display date.
"yearMonth"	Display year and month.
"yearMonthDay"	Display year, month and day.

sizeStyle A string representing the desired size style of the date picker control. This only applies in text mode. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the contents of the date picker control.

getNSDatePicker()
Return the *NSDatePicker* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the contents of the date picker control.

value A *NSDate* object.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

SearchBox

class vanilla.**SearchBox**(*posSize*, *text=''*, *callback=None*, *formatter=None*, *placeholder=None*, *sizeStyle='regular'*)
A text entry field similar to the search field in Safari.:

```
from vanilla import *

class SearchBoxDemo(object):

    def __init__(self):
        self.w = Window((100, 42))
        self.w.searchBox = SearchBox((10, 10, -10, 22),
                                     callback=self.searchBoxCallback)
        self.w.open()

    def searchBoxCallback(self, sender):
        print "search box entry!", sender.get()

SearchBoxDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the search box.

Standard Dimensions		
Regular	H	22
Small	H	19
Mini	H	15

text The text to be displayed in the search box.

callback The method to be called when the user presses the search box.

formatter A NSFormatter for controlling the display and input of the text entry.

placeholder A placeholder string to be shown when the text entry control is empty.

sizeStyle A string representing the desired size style of the search box. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the contents of the search box.

getNSSearchField()
Return the *NSSearchField* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the contents of the search box.

value A string representing the contents of the search box.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

CheckBox

class vanilla.**CheckBox**(*posSize*, *title*, *callback=None*, *value=False*, *sizeStyle='regular'*)
A standard check box.:

```
from vanilla import *

class CheckBoxDemo(object):

    def __init__(self):
        self.w = Window((100, 40))
        self.w.checkBox = CheckBox((10, 10, -10, 20), "A CheckBox",
                          callback=self.checkBoxCallback, value=True)
        self.w.open()

    def checkBoxCallback(self, sender):
        print "check box state change!", sender.get()

CheckBoxDemo()
```

posSize Tuple of form (left, top, width, height) representing the position and size of the check box. The size of the check box should match the appropriate value for the given *sizeStyle*.

Standard Dimensions		
Regular	H	22
Small	H	18
Mini	H	10

title The text to be displayed next to the check box. Pass *None* is no title is desired.

callback The method to be called when the user changes the state of the check box.

value A boolean representing the state of the check box.

sizeStyle A string representing the desired size style of the check box. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the state of the check box.

getNSButton()
Return the *NSButton* that this object wraps.

This is currently not implemented for CheckBox.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the state of the check box.

value A boolean representing the state of the check box.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

toggle()
Toggle the state of the check box.

If the check box is on, turn it off. If the check box is off, turn it on.

ColorWell

class vanilla.**ColorWell**(*posSize*, *callback=None*, *color=None*)
A control that allows for showing and choosing a color value.

ColorWell objects handle NSColor objects.:

```
from AppKit import NSColor
from vanilla import *

class ColorWellDemo(object):

    def __init__(self):
        self.w = Window((100, 50))
        self.w.colorWell = ColorWell((10, 10, -10, -10),
                                     callback=self.colorWellEdit,
                                     color=NSColor.redColor())
        self.w.open()

    def colorWellEdit(self, sender):
        print "color well edit!", sender.get()

ColorWellDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the color well.

callback The method to be caled when the user selects a new color.

color A *NSColor* object. If *None* is given, the color shown will be white.

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the *NSColor* object representing the current color in the color well.

getNSColorWell()
Return the *NSColorWell* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*color*)
Set the color in the color well.

color A *NSColor* object representing the color to be displayed in the color well.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

ComboBox

class vanilla.**ComboBox**(*posSize*, *items*, *completes=True*, *continuous=False*, *callback=None*, *formatter=None*, *sizeStyle='regular'*)

A text entry control that allows direct text entry or selection for a list of options.:

```
from vanilla import *

class ComboBoxDemo(object):

    def __init__(self):
        self.w = Window((100, 41))
        self.w.comboBox = ComboBox((10, 10, -10, 21),
                                   ["AA", "BB", "CC", "DD"],
                                   callback=self.comboBoxCallback)
        self.w.open()

    def comboBoxCallback(self, sender):
        print "combo box entry!", sender.get()

ComboBoxDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the combo box control. The size of the combo box sould match the appropriate value for the given *sizeStyle*.

Standard Dimensions		
Regular	H	21
Small	H	17
Mini	H	14

items The items to be displayed in the combo box.

completes Boolean representing if the combo box auto completes entered text.

continuous If True, the callback (if any) will be called upon each keystroke, if False, only call the callback when editing finishes or after item selection. Default is False.

callback The method to be called when the user enters text.

formatter An NSFormatter for controlling the display and input of the combo box.

sizeStyle A string representing the desired size style of the combo box. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the text entered in the combo box.

getNSComboBox()
Return the *NSComboBox* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the text in the text field of the combo box.

value A string to set in the combo box.

setItems(*items*)
Set the items in the combo box list.

items A list of strings to set in the combo box list.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

PopUpButton

class vanilla.**PopUpButton**(*posSize*, *items*, *callback=None*, *sizeStyle='regular'*)
A button which, when selected, displays a list of items for the user to choose from.:

```
from vanilla import *

class PopUpButtonDemo(object):

    def __init__(self):
        self.w = Window((100, 40))
        self.w.popUpButton = PopUpButton((10, 10, -10, 20),
                                 ["A", "B", "C"],
                                 callback=self.popUpButtonCallback)
        self.w.open()

    def popUpButtonCallback(self, sender):
        print "pop up button selection!", sender.get()

PopUpButtonDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the pop up button. The size of the button sould match the appropriate value for the given *sizeStyle*.

Standard Dimensions		
Regular	H	20
Small	H	17
Mini	H	15

items A list of items to appear in the pop up list.

callback The method to be called when the user selects an item in the pop up list.

sizeStyle A string representing the desired size style of the pop up button. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the index selected item in the pop up list.

getItems()
Get the list of items that appear in the pop up list.

getNSPopUpButton()
Return the *NSPopUpButton* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the index of the selected item in the pop up list.

setItems(*items*)
Set the items to appear in the pop up list.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

RadioGroup

class vanilla.**RadioGroup**(*posSize*, *titles*, *isVertical=True*, *callback=None*, *sizeStyle='regular'*)

A collection of radio buttons.:

```
from vanilla import *

class RadioGroupDemo(object):

    def __init__(self):
        self.w = Window((100, 60))
        self.w.radioGroup = RadioGroup((10, 10, -10, 40),
                                       ["Option 1", "Option 2"],
                                       callback=self.radioGroupCallback)
        self.w.open()

    def radioGroupCallback(self, sender):
        print "radio group edit!", sender.get()

RadioGroupDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the radio group.

titles A list of titles to be shown next to the radio buttons.

isVertical Boolean representing if the radio group is vertical or horizontal.

callback The method to be caled when a radio button is selected.

sizeStyle A string representing the desired size style of the radio group. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the index of the selected radio button.

getNSMatrix()
Return the *NSMatrix* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*index*)
Set the index of the seleced radio button.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

Slider

class vanilla.**Slider**(*posSize*, *minValue=0*, *maxValue=100*, *value=50*, *tickMarkCount=None*, *stopOnTickMarks=False*, *continuous=True*, *callback=None*, *sizeStyle='regular'*)
A standard slider control. Sliders can be vertical or horizontal and they can show tick marks or not show tick marks.:

```
from vanilla import *

class SliderDemo(object):

    def __init__(self):
        self.w = Window((200, 43))
        self.w.slider = Slider((10, 10, -10, 23),
                               tickMarkCount=10,
                               callback=self.sliderCallback)
        self.w.open()

    def sliderCallback(self, sender):
        print "slider edit!", sender.get()

SliderDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the slider. The size of the slider sould match the appropriate value for the given *sizeStyle*.

Standard Dimensions				
without ticks				
Regular	W	15	H	15
Small	W	12	H	11
Mini	W	10	H	10
with ticks				
Regular	W	24	H	23
Small	W	17	H	17
Mini	W	16	H	16

minValue The minimum value allowed by the slider.

maxValue The maximum value allowed by the slider.

value The initial value of the slider.

tickMarkCount The number of tick marcks to be displayed on the slider. If *None* is given, no tick marks will be displayed.

stopOnTickMarks Boolean representing if the slider knob should only stop on the tick marks.

continuous Boolean representing if the assigned callback should be called during slider editing. If *False* is given, the callback will be called after the editing has finished.

callback The method to be called when the slider has been edited.

sizeStyle A string representing the desired size style of the slider. The options are:

"regular"
"small"
"mini"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the value of the slider.

getNSSlider()
Return the *NSSlider* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

getTitle()
Get the control title.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the value of the slider.

setMaxValue(*value*)
Set the maximum value allowed by the slider.

setMinValue(*value*)
Set the minimum value allowed by the slider.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

setTickMarkCount(*value*)
Set the number of tick marks on the slider.

setTickMarkPosition(*value*)
Set the position of the tick marks on the slider.

For vertical sliders, the options are:

"left"
"right"

For horizontal sliders, the options are:

"top"
"bottom"

setTitle(*title*)
Set the control title.

title A string representing the title.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

Progress Indicators

ProgressBar

class vanilla.**ProgressBar**(*posSize*, *minValue=0*, *maxValue=100*, *isIndeterminate=False*, *sizeStyle='regular'*)
A standard progress bar.:

```
from vanilla import *

class ProgressBarDemo(object):

    def __init__(self):
        self.w = Window((200, 65))
        self.w.bar = ProgressBar((10, 10, -10, 16))
        self.w.button = Button((10, 35, -10, 20), "Go!",
                               callback=self.showProgress)
        self.w.open()

    def showProgress(self, sender):
        import time
        self.w.bar.set(0)
        for i in range(10):
            self.w.bar.increment(10)
            time.sleep(.2)

ProgressBarDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the progress bar. The height of the progress bar sould match the appropriate value for the given *sizeStyle*.

Standard Dimensions		
Regular	H	16
Small	H	10

minValue The minimum value of the progress bar.

maxValue The maximum value of the progress bar.

isIndeterminate Boolean representing if the progress bar is indeterminate. Determinate progress bars show how much of the task has been completed. Indeterminate progress bars simply show that the application is busy.

sizeStyle A string representing the desired size style of the pregress bar. The options are:

"regular"
"small"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

get()
Get the current value of the progress bar.

Only available in determinate progress bars.

getNSProgressIndicator()
Return the *NSProgressIndicator* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

increment(*value=1*)
Increment the progress bar by **value**.

Only available in determinate progress bars.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

set(*value*)
Set the value of the progress bar to **value**.

Only available in determinate progress bars.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

start()
Start the animation.

Only available in indeterminate progress bars.

stop()
Stop the animation.

Only available in indeterminate progress bars.

ProgressSpinner

class vanilla.**ProgressSpinner**(*posSize*, *displayWhenStopped=False*, *sizeStyle='regular'*)
An animated, spinning progress indicator.:

```
from vanilla import *

class ProgressSpinnerDemo(object):

    def __init__(self):
```

```
        self.w = Window((80, 52))
        self.w.spinner = ProgressSpinner((24, 10, 32, 32),
                                displayWhenStopped=True)
        self.w.spinner.start()
        self.w.open()

ProgressSpinnerDemo()
```

posSize Tuple of form *(left, top, width, height)* representing the position and size of the spinner. The size of the spinner sould match the appropriate value for the given *sizeStyle*.

Standard Dimensions				
Regular	W	32	H	32
Small	W	16	H	16

displayWhenStopped Boolean representing if the spiiner should be displayed when it is not spinning.

sizeStyle A string representing the desired size style of the spinner. The options are:

"regular"
"small"

enable(*onOff*)
Enable or disable the object. **onOff** should be a boolean.

getNSProgressIndicator()
Return the *NSProgressIndicator* that this object wraps.

getPosSize()
The position and size of the object as a tuple of form *(left, top, width, height)*.

isVisible()
Return a bool indicting if the object is visible or not.

move(*x*, *y*)
Move the object by **x** units and **y** units.

resize(*width*, *height*)
Change the size of the object to **width** and **height**.

setPosSize(*posSize*)
Set the postion and size of the object.

posSize A tuple of form *(left, top, width, height)*.

show(*onOff*)
Show or hide the object.

onOff A boolean value representing if the object should be shown or not.

start()
Start the animation.

stop()
Stop the animation.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

v

vanilla, 3

Index

A

addToolbar() (vanilla.FloatingWindow method), 9
addToolbar() (vanilla.Sheet method), 12
addToolbar() (vanilla.Window method), 6
assignToDocument() (vanilla.FloatingWindow method), 7
assignToDocument() (vanilla.Sheet method), 10
assignToDocument() (vanilla.Window method), 4

B

bind() (vanilla.Button method), 29
bind() (vanilla.FloatingWindow method), 8
bind() (vanilla.GradientButton method), 34
bind() (vanilla.HelpButton method), 35
bind() (vanilla.ImageButton method), 32
bind() (vanilla.Sheet method), 11
bind() (vanilla.SquareButton method), 31
bind() (vanilla.Window method), 5
Box (class in vanilla), 17
Button (class in vanilla), 29

C

center() (vanilla.FloatingWindow method), 8
center() (vanilla.Sheet method), 11
center() (vanilla.Window method), 5
CheckBox (class in vanilla), 46
CheckBoxListCell() (in module vanilla), 25
close() (vanilla.Drawer method), 14
close() (vanilla.FloatingWindow method), 7
close() (vanilla.Sheet method), 10
close() (vanilla.Window method), 4
ColorWell (class in vanilla), 47
ComboBox (class in vanilla), 49

D

DatePicker (class in vanilla), 43
Drawer (class in vanilla), 13

E

EditText (class in vanilla), 40
enable() (vanilla.Box method), 17
enable() (vanilla.Button method), 30
enable() (vanilla.CheckBox method), 47
enable() (vanilla.ColorWell method), 48
enable() (vanilla.ComboBox method), 49
enable() (vanilla.DatePicker method), 44
enable() (vanilla.Drawer method), 14
enable() (vanilla.EditText method), 40
enable() (vanilla.GradientButton method), 34
enable() (vanilla.Group method), 15
enable() (vanilla.HelpButton method), 36
enable() (vanilla.HorizontalLine method), 18
enable() (vanilla.ImageButton method), 33
enable() (vanilla.ImageView method), 26
enable() (vanilla.LevelIndicator method), 27
enable() (vanilla.List method), 23
enable() (vanilla.PopUpButton method), 51
enable() (vanilla.ProgressBar method), 55
enable() (vanilla.ProgressSpinner method), 57
enable() (vanilla.RadioGroup method), 52
enable() (vanilla.ScrollView method), 16
enable() (vanilla.SearchBox method), 45
enable() (vanilla.SecureEditText method), 42
enable() (vanilla.Slider method), 54
enable() (vanilla.SquareButton method), 31
enable() (vanilla.TextBox method), 39
enable() (vanilla.TextEditor method), 38
enable() (vanilla.VerticalLine method), 19

F

FloatingWindow (class in vanilla), 7

G

get() (vanilla.CheckBox method), 47
get() (vanilla.ColorWell method), 48
get() (vanilla.ComboBox method), 49
get() (vanilla.DatePicker method), 44
get() (vanilla.EditText method), 40
get() (vanilla.LevelIndicator method), 27

get() (vanilla.List method), 23
get() (vanilla.PopUpButton method), 51
get() (vanilla.ProgressBar method), 56
get() (vanilla.RadioGroup method), 52
get() (vanilla.SearchBox method), 45
get() (vanilla.SecureEditText method), 42
get() (vanilla.Slider method), 54
get() (vanilla.TextBox method), 39
get() (vanilla.TextEditor method), 38
getCriticalValue() (vanilla.LevelIndicator method), 28
getItems() (vanilla.PopUpButton method), 51
getMaxValue() (vanilla.LevelIndicator method), 28
getMinValue() (vanilla.LevelIndicator method), 28
getNSBox() (vanilla.Box method), 17
getNSBox() (vanilla.HorizontalLine method), 18
getNSBox() (vanilla.VerticalLine method), 19
getNSButton() (vanilla.Button method), 30
getNSButton() (vanilla.CheckBox method), 47
getNSButton() (vanilla.GradientButton method), 34
getNSButton() (vanilla.HelpButton method), 36
getNSButton() (vanilla.ImageButton method), 33
getNSButton() (vanilla.SquareButton method), 31
getNSColorWell() (vanilla.ColorWell method), 48
getNSComboBox() (vanilla.ComboBox method), 49
getNSDatePicker() (vanilla.DatePicker method), 44
getNSDrawer() (vanilla.Drawer method), 14
getNSImageView() (vanilla.ImageView method), 26
getNSLevelIndicator() (vanilla.LevelIndicator method), 28
getNSMatrix() (vanilla.RadioGroup method), 52
getNSPopUpButton() (vanilla.PopUpButton method), 51
getNSProgressIndicator() (vanilla.ProgressBar method), 56
getNSProgressIndicator() (vanilla.ProgressSpinner method), 57
getNSScrollView() (vanilla.List method), 23
getNSScrollView() (vanilla.ScrollView method), 16
getNSScrollView() (vanilla.TextEditor method), 38
getNSSearchField() (vanilla.SearchBox method), 45
getNSSecureTextField() (vanilla.SecureEditText method), 42
getNSSlider() (vanilla.Slider method), 54
getNSTableView() (vanilla.List method), 23
getNSTextField() (vanilla.EditText method), 41
getNSTextField() (vanilla.SecureEditText method), 42
getNSTextField() (vanilla.TextBox method), 39
getNSTextView() (vanilla.TextEditor method), 38
getNSView() (vanilla.Group method), 15
getNSWindow() (vanilla.FloatingWindow method), 7
getNSWindow() (vanilla.Sheet method), 10
getNSWindow() (vanilla.Window method), 4
getNSWindowController() (vanilla.FloatingWindow method), 7
getNSWindowController() (vanilla.Sheet method), 10
getNSWindowController() (vanilla.Window method), 4
getPlaceholder() (vanilla.EditText method), 41
getPlaceholder() (vanilla.SecureEditText method), 42
getPosSize() (vanilla.Box method), 17
getPosSize() (vanilla.Button method), 30
getPosSize() (vanilla.CheckBox method), 47
getPosSize() (vanilla.ColorWell method), 48
getPosSize() (vanilla.ComboBox method), 49
getPosSize() (vanilla.DatePicker method), 44
getPosSize() (vanilla.Drawer method), 14
getPosSize() (vanilla.EditText method), 41
getPosSize() (vanilla.FloatingWindow method), 8
getPosSize() (vanilla.GradientButton method), 34
getPosSize() (vanilla.Group method), 15
getPosSize() (vanilla.HelpButton method), 36
getPosSize() (vanilla.HorizontalLine method), 18
getPosSize() (vanilla.ImageButton method), 33
getPosSize() (vanilla.ImageView method), 26
getPosSize() (vanilla.LevelIndicator method), 28
getPosSize() (vanilla.List method), 23
getPosSize() (vanilla.PopUpButton method), 51
getPosSize() (vanilla.ProgressBar method), 56
getPosSize() (vanilla.ProgressSpinner method), 57
getPosSize() (vanilla.RadioGroup method), 52
getPosSize() (vanilla.ScrollView method), 16
getPosSize() (vanilla.SearchBox method), 45
getPosSize() (vanilla.SecureEditText method), 42
getPosSize() (vanilla.SegmentedButton method), 36
getPosSize() (vanilla.Sheet method), 11
getPosSize() (vanilla.Slider method), 54
getPosSize() (vanilla.SquareButton method), 31
getPosSize() (vanilla.TextBox method), 39
getPosSize() (vanilla.TextEditor method), 38
getPosSize() (vanilla.VerticalLine method), 19
getPosSize() (vanilla.Window method), 4
getSelection() (vanilla.List method), 23
getTitle() (vanilla.Box method), 17
getTitle() (vanilla.Button method), 30
getTitle() (vanilla.CheckBox method), 47
getTitle() (vanilla.ComboBox method), 49
getTitle() (vanilla.DatePicker method), 44
getTitle() (vanilla.EditText method), 41
getTitle() (vanilla.FloatingWindow method), 8
getTitle() (vanilla.GradientButton method), 34
getTitle() (vanilla.HelpButton method), 36
getTitle() (vanilla.HorizontalLine method), 18
getTitle() (vanilla.ImageButton method), 33
getTitle() (vanilla.LevelIndicator method), 28
getTitle() (vanilla.PopUpButton method), 51
getTitle() (vanilla.RadioGroup method), 52
getTitle() (vanilla.SearchBox method), 45
getTitle() (vanilla.SecureEditText method), 42
getTitle() (vanilla.SegmentedButton method), 37
getTitle() (vanilla.Sheet method), 11

getTitle() (vanilla.Slider method), 54
getTitle() (vanilla.SquareButton method), 31
getTitle() (vanilla.TextBox method), 39
getTitle() (vanilla.VerticalLine method), 19
getTitle() (vanilla.Window method), 4
getWarningValue() (vanilla.LevelIndicator method), 28
GradientButton (class in vanilla), 34
Group (class in vanilla), 15

H

HelpButton (class in vanilla), 35
hide() (vanilla.FloatingWindow method), 7
hide() (vanilla.Sheet method), 11
hide() (vanilla.Window method), 4
HorizontalLine (class in vanilla), 18

I

ImageButton (class in vanilla), 32
ImageView (class in vanilla), 25
increment() (vanilla.ProgressBar method), 56
isOpen() (vanilla.Drawer method), 14
isVisible() (vanilla.Box method), 17
isVisible() (vanilla.Button method), 30
isVisible() (vanilla.CheckBox method), 47
isVisible() (vanilla.ColorWell method), 48
isVisible() (vanilla.ComboBox method), 49
isVisible() (vanilla.DatePicker method), 44
isVisible() (vanilla.Drawer method), 14
isVisible() (vanilla.EditText method), 41
isVisible() (vanilla.FloatingWindow method), 8
isVisible() (vanilla.GradientButton method), 34
isVisible() (vanilla.Group method), 15
isVisible() (vanilla.HelpButton method), 36
isVisible() (vanilla.HorizontalLine method), 18
isVisible() (vanilla.ImageButton method), 33
isVisible() (vanilla.ImageView method), 26
isVisible() (vanilla.LevelIndicator method), 28
isVisible() (vanilla.List method), 23
isVisible() (vanilla.PopUpButton method), 51
isVisible() (vanilla.ProgressBar method), 56
isVisible() (vanilla.ProgressSpinner method), 57
isVisible() (vanilla.RadioGroup method), 52
isVisible() (vanilla.ScrollView method), 16
isVisible() (vanilla.SearchBox method), 45
isVisible() (vanilla.SecureEditText method), 42
isVisible() (vanilla.SegmentedButton method), 37
isVisible() (vanilla.Sheet method), 11
isVisible() (vanilla.Slider method), 54
isVisible() (vanilla.SquareButton method), 31
isVisible() (vanilla.TextBox method), 39
isVisible() (vanilla.TextEditor method), 38
isVisible() (vanilla.VerticalLine method), 19
isVisible() (vanilla.Window method), 4

L

LevelIndicator (class in vanilla), 26
LevelIndicatorListCell() (in module vanilla), 24
List (class in vanilla), 20

M

makeKey() (vanilla.FloatingWindow method), 7
makeKey() (vanilla.Sheet method), 11
makeKey() (vanilla.Window method), 4
makeMain() (vanilla.FloatingWindow method), 8
makeMain() (vanilla.Sheet method), 11
makeMain() (vanilla.Window method), 4
move() (vanilla.Box method), 18
move() (vanilla.Button method), 30
move() (vanilla.CheckBox method), 47
move() (vanilla.ColorWell method), 48
move() (vanilla.ComboBox method), 50
move() (vanilla.DatePicker method), 44
move() (vanilla.Drawer method), 14
move() (vanilla.EditText method), 41
move() (vanilla.FloatingWindow method), 8
move() (vanilla.GradientButton method), 34
move() (vanilla.Group method), 15
move() (vanilla.HelpButton method), 36
move() (vanilla.HorizontalLine method), 18
move() (vanilla.ImageButton method), 33
move() (vanilla.ImageView method), 26
move() (vanilla.LevelIndicator method), 28
move() (vanilla.List method), 23
move() (vanilla.PopUpButton method), 51
move() (vanilla.ProgressBar method), 56
move() (vanilla.ProgressSpinner method), 57
move() (vanilla.RadioGroup method), 52
move() (vanilla.ScrollView method), 16
move() (vanilla.SearchBox method), 46
move() (vanilla.SecureEditText method), 43
move() (vanilla.SegmentedButton method), 37
move() (vanilla.Sheet method), 11
move() (vanilla.Slider method), 54
move() (vanilla.SquareButton method), 31
move() (vanilla.TextBox method), 39
move() (vanilla.TextEditor method), 38
move() (vanilla.VerticalLine method), 19
move() (vanilla.Window method), 5

O

open() (vanilla.Drawer method), 14
open() (vanilla.FloatingWindow method), 7
open() (vanilla.Sheet method), 10
open() (vanilla.Window method), 4

P

PopUpButton (class in vanilla), 50

ProgressBar (class in vanilla), 55
ProgressSpinner (class in vanilla), 56

R

RadioGroup (class in vanilla), 52
resize() (vanilla.Box method), 18
resize() (vanilla.Button method), 30
resize() (vanilla.CheckBox method), 47
resize() (vanilla.ColorWell method), 48
resize() (vanilla.ComboBox method), 50
resize() (vanilla.DatePicker method), 44
resize() (vanilla.Drawer method), 14
resize() (vanilla.EditText method), 41
resize() (vanilla.FloatingWindow method), 8
resize() (vanilla.GradientButton method), 34
resize() (vanilla.Group method), 15
resize() (vanilla.HelpButton method), 36
resize() (vanilla.HorizontalLine method), 18
resize() (vanilla.ImageButton method), 33
resize() (vanilla.ImageView method), 26
resize() (vanilla.LevelIndicator method), 28
resize() (vanilla.List method), 24
resize() (vanilla.PopUpButton method), 51
resize() (vanilla.ProgressBar method), 56
resize() (vanilla.ProgressSpinner method), 57
resize() (vanilla.RadioGroup method), 52
resize() (vanilla.ScrollView method), 17
resize() (vanilla.SearchBox method), 46
resize() (vanilla.SecureEditText method), 43
resize() (vanilla.SegmentedButton method), 37
resize() (vanilla.Sheet method), 11
resize() (vanilla.Slider method), 54
resize() (vanilla.SquareButton method), 31
resize() (vanilla.TextBox method), 39
resize() (vanilla.TextEditor method), 38
resize() (vanilla.VerticalLine method), 19
resize() (vanilla.Window method), 5

S

scrollToSelection() (vanilla.List method), 24
ScrollView (class in vanilla), 16
SearchBox (class in vanilla), 45
SecureEditText (class in vanilla), 41
SegmentedButton (class in vanilla), 36
select() (vanilla.FloatingWindow method), 8
select() (vanilla.Sheet method), 11
select() (vanilla.Window method), 4
selectAll() (vanilla.EditText method), 41
selectAll() (vanilla.SecureEditText method), 43
selectAll() (vanilla.TextEditor method), 38
set() (vanilla.CheckBox method), 47
set() (vanilla.ColorWell method), 48
set() (vanilla.ComboBox method), 50
set() (vanilla.DatePicker method), 44
set() (vanilla.EditText method), 41
set() (vanilla.LevelIndicator method), 28
set() (vanilla.List method), 24
set() (vanilla.PopUpButton method), 51
set() (vanilla.ProgressBar method), 56
set() (vanilla.RadioGroup method), 52
set() (vanilla.SearchBox method), 46
set() (vanilla.SecureEditText method), 43
set() (vanilla.Slider method), 54
set() (vanilla.TextBox method), 39
set() (vanilla.TextEditor method), 38
setBackgroundColor() (vanilla.ScrollView method), 17
setCriticalValue() (vanilla.LevelIndicator method), 28
setDefaultButton() (vanilla.FloatingWindow method), 8
setDefaultButton() (vanilla.Sheet method), 11
setDefaultButton() (vanilla.Window method), 5
setImage() (vanilla.GradientButton method), 34
setImage() (vanilla.ImageButton method), 33
setImage() (vanilla.ImageView method), 26
setItems() (vanilla.ComboBox method), 50
setItems() (vanilla.PopUpButton method), 51
setMaxValue() (vanilla.LevelIndicator method), 28
setMaxValue() (vanilla.Slider method), 54
setMinValue() (vanilla.LevelIndicator method), 28
setMinValue() (vanilla.Slider method), 54
setPlaceholder() (vanilla.EditText method), 41
setPlaceholder() (vanilla.SecureEditText method), 43
setPosSize() (vanilla.Box method), 18
setPosSize() (vanilla.Button method), 30
setPosSize() (vanilla.CheckBox method), 47
setPosSize() (vanilla.ColorWell method), 48
setPosSize() (vanilla.ComboBox method), 50
setPosSize() (vanilla.DatePicker method), 44
setPosSize() (vanilla.Drawer method), 14
setPosSize() (vanilla.EditText method), 41
setPosSize() (vanilla.FloatingWindow method), 8
setPosSize() (vanilla.GradientButton method), 35
setPosSize() (vanilla.Group method), 15
setPosSize() (vanilla.HelpButton method), 36
setPosSize() (vanilla.HorizontalLine method), 19
setPosSize() (vanilla.ImageButton method), 33
setPosSize() (vanilla.ImageView method), 26
setPosSize() (vanilla.LevelIndicator method), 28
setPosSize() (vanilla.List method), 24
setPosSize() (vanilla.PopUpButton method), 51
setPosSize() (vanilla.ProgressBar method), 56
setPosSize() (vanilla.ProgressSpinner method), 57
setPosSize() (vanilla.RadioGroup method), 52
setPosSize() (vanilla.ScrollView method), 17
setPosSize() (vanilla.SearchBox method), 46
setPosSize() (vanilla.SecureEditText method), 43
setPosSize() (vanilla.SegmentedButton method), 37
setPosSize() (vanilla.Sheet method), 11
setPosSize() (vanilla.Slider method), 54

setPosSize() (vanilla.SquareButton method), 31
setPosSize() (vanilla.TextBox method), 39
setPosSize() (vanilla.TextEditor method), 38
setPosSize() (vanilla.VerticalLine method), 19
setPosSize() (vanilla.Window method), 5
setSelection() (vanilla.List method), 24
setTickMarkCount() (vanilla.Slider method), 54
setTickMarkPosition() (vanilla.Slider method), 54
setTitle() (vanilla.Box method), 18
setTitle() (vanilla.Button method), 30
setTitle() (vanilla.CheckBox method), 47
setTitle() (vanilla.ComboBox method), 50
setTitle() (vanilla.DatePicker method), 44
setTitle() (vanilla.EditText method), 41
setTitle() (vanilla.FloatingWindow method), 8
setTitle() (vanilla.GradientButton method), 35
setTitle() (vanilla.HelpButton method), 36
setTitle() (vanilla.HorizontalLine method), 19
setTitle() (vanilla.ImageButton method), 33
setTitle() (vanilla.LevelIndicator method), 28
setTitle() (vanilla.PopUpButton method), 51
setTitle() (vanilla.RadioGroup method), 53
setTitle() (vanilla.SearchBox method), 46
setTitle() (vanilla.SecureEditText method), 43
setTitle() (vanilla.SegmentedButton method), 37
setTitle() (vanilla.Sheet method), 11
setTitle() (vanilla.Slider method), 55
setTitle() (vanilla.SquareButton method), 31
setTitle() (vanilla.TextBox method), 39
setTitle() (vanilla.VerticalLine method), 20
setTitle() (vanilla.Window method), 4
setWarningValue() (vanilla.LevelIndicator method), 28
Sheet (class in vanilla), 10
show() (vanilla.Box method), 18
show() (vanilla.Button method), 30
show() (vanilla.CheckBox method), 47
show() (vanilla.ColorWell method), 48
show() (vanilla.ComboBox method), 50
show() (vanilla.DatePicker method), 44
show() (vanilla.Drawer method), 15
show() (vanilla.EditText method), 41
show() (vanilla.FloatingWindow method), 7
show() (vanilla.GradientButton method), 35
show() (vanilla.Group method), 16
show() (vanilla.HelpButton method), 36
show() (vanilla.HorizontalLine method), 19
show() (vanilla.ImageButton method), 34
show() (vanilla.ImageView method), 26
show() (vanilla.LevelIndicator method), 28
show() (vanilla.List method), 24
show() (vanilla.PopUpButton method), 51
show() (vanilla.ProgressBar method), 56
show() (vanilla.ProgressSpinner method), 57
show() (vanilla.RadioGroup method), 53
show() (vanilla.ScrollView method), 17
show() (vanilla.SearchBox method), 46
show() (vanilla.SecureEditText method), 43
show() (vanilla.SegmentedButton method), 37
show() (vanilla.Sheet method), 11
show() (vanilla.Slider method), 55
show() (vanilla.SquareButton method), 32
show() (vanilla.TextBox method), 39
show() (vanilla.TextEditor method), 38
show() (vanilla.VerticalLine method), 20
show() (vanilla.Window method), 4
Slider (class in vanilla), 53
SliderListCell() (in module vanilla), 25
SplitView (in module vanilla), 17
SquareButton (class in vanilla), 30
start() (vanilla.ProgressBar method), 56
start() (vanilla.ProgressSpinner method), 57
stop() (vanilla.ProgressBar method), 56
stop() (vanilla.ProgressSpinner method), 57

T

TextBox (class in vanilla), 38
TextEditor (class in vanilla), 37
toggle() (vanilla.CheckBox method), 47
toggle() (vanilla.Drawer method), 15

U

unbind() (vanilla.FloatingWindow method), 9
unbind() (vanilla.Sheet method), 12
unbind() (vanilla.Window method), 5

V

vanilla (module), 3, 7, 10, 13, 15–20, 25, 26, 29, 30, 32, 34–38, 40, 41, 43, 45–47, 49, 50, 52, 53, 55, 56
VerticalLine (class in vanilla), 19

W

Window (class in vanilla), 3