UI Automation JavaScript Reference

Developer

Contents

Tables

UI Automation JavaScript Reference

Note: This document was previously titled UI Automation Reference Collection.

Use the UI Automation JavaScript library to write test scripts that exercise your app's user interface elements while the app runs on a connected device. You write the tests in JavaScript, calling the UI Automation API to simulate user interaction. The system returns log information to the host computer.

Note: UI Automation simulates all user interface actions initiated by the script. For the sake of brevity and clarity, this document describes those actions in terms of a user's perspective.

Accessing and Using User Interface Elements

In essence, your test script is an ordered set of commands, each of which accesses a user interface element in your app to perform a user action on it or to use the information associated within it. All the user interface elements in your app are represented to the script through an ordered hierarchy of objects defined by the `UIAElements` class and its subclasses. To reach a specified UI element, the script simply calls down the element hierarchy, starting with the top-level target object obtained by calling `UIATarget.localTarget()`. For example, the first button in the main window of your app might be referenced by index as follows:

```
UIATarget.localTarget().frontMostApp().mainWindow().buttons()[0]
```

If that first button is identified in your code as the Edit button, the following would also work:

```
UIATarget.localTarget().frontMostApp().mainWindow().buttons()["Edit"]
```

To tap that button, then, the script could use any of these three formats:

- `UIATarget.localTarget().frontMostApp().mainWindow().buttons()[0].tap();`
- `UIATarget.localTarget().frontMostApp().mainWindow().buttons()["Edit"].tap();`
- `var editButton=UIATarget.localTarget().frontMostApp().mainWindow().buttons()[0];`
  `editButton.tap();`

The Automation instrument maintains a complete element hierarchy that represents your app's user interface. To view that hierarchy, use the `logElementTree` method to write an outline of it to the log:

```
UIATarget.localTarget().frontMostApp().logElementTree()
```

Recording Results With the Log

To record data during its tests, the script uses `UIALogger` class methods to send messages to the Automation instrument running on the host computer. Various methods are available to assist in organizing and analyzing the recorded data. For example:

- To indicate the initiation of a specified test, use the `logStart` method:
  - `UIALogger.logStart("Test1");`
- To end a test and mark it as failed, use the `logFail` method:
  - `UIALogger.logFail("Failed to foo.");`
- To send a general-purpose debug message, use the `logDebug` method:
  - `UIALogger.logDebug("Done with level 3.");`

You view the collected data in the Detail pane of the Automation instrument using Instruments.

Handling Alerts

When UI Automation encounters an alert during the execution of your script, it calls your alert handler, passing a reference to the `UIAAlert` object representing the alert. Your script should handle the alert appropriately and return a value of `true`, upon which normal script execution continues.

To ensure that alerts don't interfere with testing, the Automation instrument also implements a simple default alert handler. If your script's alert handler returns `false`, this default handler attempts to dismiss the alert by tapping the cancel button, if it exists; otherwise, it taps the default button.

The following code implements a simple alert handler that records a message to the log and returns `false`, thereby depending on the default handler to dismiss the alert:

```
UIATarget.onAlert = function onAlert(alert) {
    var title = alert.name();

    // add a warning to the log for each alert encountered
```

```
    UIALogger.logWarning("Alert with title '" + title + "' encountered!");
    UIATarget.localTarget().captureScreenWithName("alert_" + (new Date()).UTC());

    // test if your script should handle the alert, and if so, return true

    // otherwise, return false to use the default handler
    return false;
}
```

Classes

UIAActionSheet Class Reference

Inherits from	UIAElement

Overview

The `UIAActionSheet` class allows access to, and control of, action sheets within your app.

For an explanation of how to use this and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 1-1 provides a list of methods inherited from `UIAElement`.

Table 1-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified action sheet.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified action sheet.
`buttons` (page 51)	Returns an array of buttons contained by the specified action sheet.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified action sheet.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.

Method	Description
`elements` (page 52)	Returns an array of elements contained by the specified action sheet.
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified action sheet.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified action sheet.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this action sheet.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified action sheet.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified action sheet.
`popover` (page 56)	Returns the popover object associated with the specified action sheet.

Method	Description
progressIndicators (page 57)	Returns an array of progress indicators contained by the specified action sheet.
rect (page 57)	Returns the position of the object on the main screen.
rotateWithOptions (page 57)	Perform a rotation gesture centered on the specified element.
scrollToVisible (page 58)	Scrolls until the specified element is visible in a container view.
scrollViews (page 58)	Returns an array of scroll views contained by the specified action sheet.
searchBars (page 58)	Returns an array of search bars contained by the specified action sheet.
secureTextFields (page 58)	Returns an array of secure text fields contained by the specified action sheet.
segmentedControls (page 58)	Returns an array of segmented controls contained by the specified action sheet.
sliders (page 58)	Returns an array of sliders contained by the specified action sheet.
staticTexts (page 59)	Returns an array of static texts contained by the specified action sheet.
switches (page 59)	Returns an array of switches contained by the specified action sheet.
tabBar (page 59)	Returns the specified tab bar.
tabBars (page 59)	Returns an array of tab bars contained by this action sheet.
tableViews (page 59)	Returns an array of table views contained by the specified action sheet.
tap (page 59)	Taps the specified element.
tapWithOptions (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
textFields (page 60)	Returns an array of text fields contained by the specified action sheet.

Method	Description
`textViews` (page 61)	Returns an array of text views contained by the specified action sheet.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this action sheet.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified action sheet.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Working With Buttons

`cancelButton` (page 18)

Returns the Cancel button in the action sheet.

Methods

cancelButton

Returns the Cancel button in the action sheet.

```
(UIAButton) cancelButton()
```

UIAActivityIndicator Class Reference

Inherits from	UIAElement

Overview

The `UIAActivityIndicator` class allows access to, and control of, activity indicator elements in your app.

For an explanation of how to use this and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 2-1 provides a list of methods inherited from `UIAElement`.

Table 2-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified activity indicator.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified activity indicator.
`buttons` (page 51)	Returns an array of buttons contained by the specified activity indicator.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified activity indicator.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.

Method	Description
`elements` (page 52)	Returns an array of elements contained by the specified activity indicator.
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified activity indicator.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified activity indicator.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this activity indicator.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified activity indicator.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified activity indicator.

Method	Description
popover (page 56)	Returns the popover object associated with the specified activity indicator.
progressIndicators (page 57)	Returns an array of progress indicators contained by the specified activity indicator.
rect (page 57)	Returns the position of the object on the main screen.
rotateWithOptions (page 57)	Perform a rotation gesture centered on the specified element.
scrollToVisible (page 58)	Scrolls until the specified element is visible in a container view.
scrollViews (page 58)	Returns an array of scroll views contained by the specified activity indicator.
searchBars (page 58)	Returns an array of search bars contained by the specified activity indicator.
secureTextFields (page 58)	Returns an array of secure text fields contained by the specified activity indicator.
segmentedControls (page 58)	Returns an array of segmented controls contained by the specified activity indicator.
sliders (page 58)	Returns an array of sliders contained by the specified activity indicator.
staticTexts (page 59)	Returns an array of static texts contained by the specified activity indicator.
switches (page 59)	Returns an array of switches contained by the specified activity indicator.
tabBar (page 59)	Returns the specified tab bar.
tabBars (page 59)	Returns an array of tab bars contained by this activity indicator.
tableViews (page 59)	Returns an array of table views contained by the specified activity indicator.
tap (page 59)	Taps the specified element.
tapWithOptions (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
textFields (page 60)	Returns an array of text fields contained by the specified activity indicator.

Method	Description
`textViews` (page 61)	Returns an array of text views contained by the specified activity indicator.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this activity indicator.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified activity indicator.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIAActivityView Class Reference

Inherits from	UIAElement

Overview

The `UIAActivityView` class allows access to, and control of, activity views within your app.

For an explanation of how to use this class and related classes, see the "Automating UI Testing" chapter in the *Instruments User Guide*.

Inherited Methods

Table 3-1 provides a list of methods inherited from `UIAElement`.

Table 3-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Working With Buttons

`cancelButton` (page 26)

Returns the cancel button in the activity view.

Methods

cancelButton

Returns the cancel button in the activity view.

```
(UIAButton) cancelButton()
```

UIAAlert Class Reference

Inherits from	UIAElement

Overview

The `UIAAlert` class allows access to, and control of, alerts within your app.

For an explanation of how to use this and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 4-1 provides a list of methods inherited from `UIAElement`.

Table 4-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified alert.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified alert.
`buttons` (page 51)	Returns an array of buttons contained by the specified alert.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified alert.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified alert.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified alert.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified alert.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by specified alert.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified alert.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified alert.
`popover` (page 56)	Returns the popover object associated with the specified alert.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified alert.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
rotateWithOptions (page 57)	Perform a rotation gesture centered on the specified element.
scrollToVisible (page 58)	Scrolls until the specified element is visible in a container view.
scrollViews (page 58)	Returns an array of scroll views contained by the specified alert.
searchBars (page 58)	Returns an array of search bars contained by the specified alert.
secureTextFields (page 58)	Returns an array of secure text fields contained by the specified alert.
segmentedControls (page 58)	Returns an array of segmented controls contained by the specified alert.
sliders (page 58)	Returns an array of sliders contained by the specified alert.
staticTexts (page 59)	Returns an array of static texts contained by the specified alert.
switches (page 59)	Returns an array of switches contained by the specified alert.
tabBar (page 59)	Returns the specified tab bar.
tabBars (page 59)	Returns an array of tab bars contained by this alert.
tableViews (page 59)	Returns an array of table views contained by the specified alert.
tap (page 59)	Taps the specified element.
tapWithOptions (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
textFields (page 60)	Returns an array of text fields contained by the specified alert.
textViews (page 61)	Returns an array of text views contained by the specified alert.
toolbar (page 61)	Returns the specified toolbar.
toolbars (page 61)	Returns an array of toolbars contained by this alert.
touchAndHold (page 61)	Touches the specified element and holds for the specified duration.
twoFingerTap (page 61)	Performs a two-finger (two-touch) tap on this element.
value (page 62)	Returns a string containing a value attribute specific to the type of element.

Method	Description
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified alert.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Accessing Alert Buttons

`cancelButton` (page 30)

Returns the cancel button contained in the alert.

`defaultButton` (page 30)

Returns the default button contained in the alert.

Methods

cancelButton

Returns the cancel button contained in the alert.

```
(UIAButton) cancelButton()
```

defaultButton

Returns the default button contained in the alert.

```
(UIAButton) defaultButton()
```

UIAApplication Class Reference

Overview

The `UIAApplication` class allows access to, and control of, app-level user interface elements.

For an explanation of how to use this and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Tasks

Working With App-level Elements

`actionSheet` (page 32)
Returns the action sheet.

`alert` (page 32)
Returns the alert.

`bundleID` (page 32)
Returns a string containing the app's bundle ID in reverse-DNS format.

`editingMenu` (page 33)
Returns the app's edit menu.

`interfaceOrientation` (page 33)
Returns the interface orientation.

`keyboard` (page 33)
Returns the app's keyboard.

`mainWindow` (page 33)
Returns the object that represents the app's main window.

`navigationBar` (page 33)
Returns the app's navigation bar.

`preferencesValueForKey` (page 33)

Gets the value of a specified app user preference.

`setPreferencesValueForKey` (page 34)

Sets the value of a specified app user preference.

`statusBar` (page 34)

Returns the app's status bar.

`tabBar` (page 34)

Returns the app's tab bar.

`toolbar` (page 34)

Returns the app's toolbar.

`version` (page 34)

Returns a string representing the build version number of the app.

`windows` (page 35)

Returns an array of objects representing the app's windows.

Methods

actionSheet

Returns the action sheet.

```
(UIAActionSheet) actionSheet()
```

alert

Returns the alert.

```
(UIAAlert) alert()
```

bundleID

Returns a string containing the app's bundle ID in reverse-DNS format.

```
(String) bundleID()
```

editingMenu

Returns the app's edit menu.

```
(UIAEditingMenu) editingMenu()
```

interfaceOrientation

Returns the interface orientation.

```
(Number) interfaceOrientation()
```

Discussion
Interface orientation represents the rotation required to keep the interface right-side up upon device rotation. Note that in landscape mode, device orientation and interface orientation are opposite, because rotating the device requires rotating the content in the opposite direction.

keyboard

Returns the app's keyboard.

```
(UIAKeyboard) keyboard()
```

mainWindow

Returns the object that represents the app's main window.

```
(UIAWindow) mainWindow()
```

navigationBar

Returns the app's navigation bar.

```
(UIANavigationBar) navigationBar())
```

preferencesValueForKey

Gets the value of a specified app user preference.

```
(NotTyped) preferencesValueForKey(key)
```

Parameters

`key`

The key representing the preference to be set.

setPreferencesValueForKey

Sets the value of a specified app user preference.

```
(undefined) setPreferencesValueForKey(NotTyped value, String key)
```

Parameters

`value`

The new value to set for the specified preference.

`key`

The key representing the preference to be set.

statusBar

Returns the app's status bar.

```
(UIAStatusBar) statusBar()
```

tabBar

Returns the app's tab bar.

```
(UIATabBar) tabBar()
```

toolbar

Returns the app's toolbar.

```
(UIAToolbar) toolbar()
```

version

Returns a string representing the build version number of the app.

```
(String) version()
```

Discussion

The build version is represented by a string comprising three period-separated integers.

windows

Returns an array of objects representing the app's windows.

```
(UIAElementArray) windows()
```

UIAButton Class Reference

Inherits from	UIAElement

Overview

The `UIAButton` class allows access to, and control of, button elements in your app.

For an explanation of how to use this and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 6-1 provides a list of methods inherited from `UIAElement`.

Table 6-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified button.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified button.
`buttons` (page 51)	Returns an array of buttons contained by the specified button.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified button.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified button.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified button.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified button.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this button.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified button.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified button.
`popover` (page 56)	Returns the popover object associated with the specified button.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified button.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified button.
`searchBars` (page 58)	Returns an array of search bars contained by the specified button.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified button.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified button.
`sliders` (page 58)	Returns an array of sliders contained by the specified button.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified button.
`switches` (page 59)	Returns an array of switches contained by the specified button.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this button.
`tableViews` (page 59)	Returns an array of table views contained by the specified button.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified button.
`textViews` (page 61)	Returns an array of text views contained by the specified button.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this button.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.

Method	Description
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified button.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIACollectionView Class Reference

Inherits from	UIAScrollView
Availability	Available in iOS 6.0 and later

Overview

The `UIACollectionView` class allows access to, and control of, elements within a collection view in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 7-1 provides a list of methods inherited from `UIAScrollView`.

Table 7-1 Methods inherited from `UIAScrollView`

Method	Description
`scrollDown` (page 115)	Scrolls down within the specified collection view.
`scrollLeft` (page 115)	Scrolls left within the specified collection view.
`scrollRight` (page 115)	Scrolls right within the specified collection view.
`scrollToElementWithName` (page 115)	Scrolls within the collection view until the named element is displayed on the screen.
`scrollToElementWithPredicate` (page 115)	Scrolls within the collection view until the matching element is displayed on the screen.
`scrollToElementWithValueForKey` (page 116)	Scrolls within the collection view until the element with the specified value for the specified key is displayed on the screen.
`scrollUp` (page 116)	Scrolls up within the specified collection view.

Tasks

Working With Cells

`cells` (page 41)

Returns an array of elements within the collection view.

`visibleCells` (page 41)

Returns an array visible elements within the collection view.

Methods

cells

Returns an array of elements within the collection view.

```
(UIAElementArray) cells()
```

visibleCells

Returns an array visible elements within the collection view.

```
(UIAElementArray) visibleCells()
```

UIAEditingMenu Class Reference

Inherits from	UIAElement

Overview

The `UIAEditingMenu` class allows access to, and control of, your app's edit menu.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 8-1 provides a list of methods inherited from `UIAElement`.

Table 8-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified menu.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified menu.
`buttons` (page 51)	Returns an array of buttons contained by the specified menu.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified menu.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified menu.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified menu.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified menu.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this menu.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified menu.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified menu.
`popover` (page 56)	Returns the popover object associated with the specified menu.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified menu.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified menu.
`searchBars` (page 58)	Returns an array of search bars contained by the specified menu.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified menu.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified menu.
`sliders` (page 58)	Returns an array of sliders contained by the specified menu.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified menu.
`switches` (page 59)	Returns an array of switches contained by the specified menu.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this menu.
`tableViews` (page 59)	Returns an array of table views contained by the specified menu.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified menu.
`textViews` (page 61)	Returns an array of text views contained by the specified menu.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this menu.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.

Method	Description
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified menu.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIAElement Class Reference

Overview

The `UIAElement` class is the superclass for all user interface elements in the context of the Automation instrument for automating user interface testing of iOS apps.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Tasks

Determining Element Positioning

`hitpoint` (page 53)
Returns the screen position to tap for the specified element.

`rect` (page 57)
Returns the position of the object on the main screen.

Determining and Manipulating Element Hierarchy

`activityIndicators` (page 50)
Returns an array of the activity indicators contained by the specified object.

`activityView` (page 50)
Returns an object representing an activity view.

`ancestry` (page 50)
Returns an array containing the parents of the specified object.

`buttons` (page 51)
Returns an array of buttons contained by the specified object.

`collectionViews` (page 51)

Returns an array of collection views contained by the specified object.

`elements` (page 52)

Returns an array of elements contained by the specified object.

`images` (page 54)

Returns an array of images contained by the specified object.

`links` (page 55)

Returns an array of links contained by the specified object.

`navigationBar` (page 56)

Returns the app's navigation bar.

`navigationBars` (page 56)

Returns an array of navigation bar objects contained by this object.

`pageIndicators` (page 56)

Returns an array of page indicators contained by the specified object.

`parent` (page 56)

Returns the parent of the specified element.

`pickers` (page 56)

Returns an array of picker objects contained by the specified object.

`popover` (page 56)

Returns the popover object associated with the specified object, if one exists.

`progressIndicators` (page 57)

Returns an array of progress indicators contained by the specified object.

`scrollViews` (page 58)

Returns an array of scroll views contained by the specified object.

`searchBars` (page 58)

Returns an array of search bars contained by the specified object.

`secureTextFields` (page 58)

Returns an array of secure text fields contained by the specified object.

`segmentedControls` (page 58)

Returns an array of segmented controls contained by the specified object.

`sliders` (page 58)

Returns an array of sliders contained by the specified object.

`staticTexts` (page 59)

Returns an array of static texts contained by the specified object.

`switches` (page 59)
Returns an array of switches contained by the specified object.

`tabBar` (page 59)
Returns the specified tab bar.

`tabBars` (page 59)
Returns an array of tab bars contained by this object.

`tableViews` (page 59)
Returns an array of table views contained by the specified object.

`textFields` (page 60)
Returns an array of text fields contained by the specified object.

`textViews` (page 61)
Returns an array of text views contained by the specified object.

`toolbar` (page 61)
Returns the specified toolbar.

`toolbars` (page 61)
Returns an array of toolbars contained by this object.

`webViews` (page 62)
Returns an array of web views contained by the specified object.

Gestures and Actions

These methods allow you to effect the common gestures and actions a user can perform through the user interface. Options are available for use with some of these methods to give you flexibility in defining and varying the attributes of the gesture or action to be performed.

`doubleTap` (page 51)
Double-taps the specified element.

`dragInsideWithOptions` (page 51)
Drags within the bounds of an element.

`flickInsideWithOptions` (page 52)
Flicks within the bounds of an element.

`rotateWithOptions` (page 57)
Perform a rotation gesture centered on the specified element.

`scrollToVisible` (page 58)
Scrolls until the specified element is visible in a container view.

`tap` (page 59)

Taps the specified element.

`tapWithOptions` (page 60)

Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.

`touchAndHold` (page 61)

Touches the specified element and holds for the specified duration.

`twoFingerTap` (page 61)

Performs a two-finger (two-touch) tap on this element.

Determining Element State

Use these methods to determine whether an element is still valid.

`checkIsValid` (page 51)

Returns the specified element's current validity status.

`hasKeyboardFocus` (page 53)

Determines whether the specified element receives keyboard input.

`isEnabled` (page 54)

Determines whether the specified element is enabled.

`isValid` (page 54)

Returns the specified element's validity status as of the most recent access.

`isVisible` (page 54)

Determines whether the specified element is visible on the screen.

`waitForInvalid` (page 62)

Waits for the specified element to become invalid.

Identifying Elements

`label` (page 54)

Returns a string containing the label attribute of the element.

`name` (page 55)

Returns a string containing the name attribute of the element.

`value` (page 62)

Returns a string containing a value attribute specific to the type of element.

`withName` (page 62)

Returns an element whose name attribute matches a specified string.

`withPredicate` (page 62)

Returns the element matching the specified criteria.

`withValueForKey` (page 63)

Returns the element containing the specified property with the specified value.

Logging Element Information

`logElement` (page 55)

Logs information about the specified element.

`logElementTree` (page 55)

Logs information about the specified element and all of its subelements.

Methods

activityIndicators

Returns an array of the activity indicators contained by the specified object.

```
(UIAElementArray) activityIndicators()
```

activityView

Returns an object representing an activity view.

```
(UIAActivityView) activityView()
```

ancestry

Returns an array containing the parents of the specified object.

```
(UIAElementArray) ancestry()
```

buttons

Returns an array of buttons contained by the specified object.

```
(UIAElementArray) buttons()
```

checkIsValid

Returns the specified element's current validity status.

```
(Boolean) checkIsValid()
```

Discussion
Use this method to determine whether the user interface element represented by the specified UIAElement currently exists. You should use `checkIsValid`, for example, if you're referencing an element after having performed some action that may have changed the UI state of that element in some way. This requires a call to the underlying Accessibility framework to ensure the validity of the result.

See Also
`isValid`

collectionViews

Returns an array of collection views contained by the specified object.

```
(UIAElementArray collectionViews()
```

doubleTap

Double-taps the specified element.

```
(undefined) doubleTap()
```

dragInsideWithOptions

Drags within the bounds of an element.

```
(undefined) dragInsideWithOptions(Object options)
```

Parameters

`options`

A dictionary that specifies characteristics of the gesture. Valid keys are as follows:

`touchCount`	The number of touches to use in the specified gesture. (Effectively, the number of fingers a user would use to make the specified gesture.) The default touch count value is 1.
`duration`	The length of hold time for the specified gesture. The default duration value for a tap is 0. The default value for touch-and-hold gestures (such as drag, pinch open, and pinch close) is 1.
`startOffset`	The first offset to use for a multiple-point gesture. The default value is {x:0.0, y:0.0}. See the discussion for details.
`endOffset`	The last offset to use for a multiple-point gesture. The default value is {x:0.0, y:0.0}. See the discussion for details.

Discussion

You can use offsets to achieve finer precision in specifying the hitpoint within the `rect` for the specified element. The offset comprises a pair of x and y values, each ranging from 0.0 to 1.0. These values represent, respectively, relative horizontal and vertical positions within the `rect`, with {x:0.0, y:0.0} as the top left and {x:1.0, y:1.0} as the bottom right. Thus, {x:0.3, y:0.6} specifies a position just below and to the left of center, and {x:1.0, y:0.5} specifies a position centered vertically at the far right.

This example performs a slow drag within the target element from left edge to right edge, just below the top:

```
target.dragInsideWithOptions({startOffset:{x:0.0, y:0.1}, endOffset:{x:1.0, y:0.1},
duration:1.5});
```

elements

Returns an array of elements contained by the specified object.

```
(UIAElementArray) elements()
```

flickInsideWithOptions

Flicks within the bounds of an element.

```
(undefined) flickInsideWithOptions(Object options)
```

Parameters

`options`

A dictionary that specifies characteristics of the gesture. Valid keys are as follows:

`touchCount`	The number of touches to use in the specified gesture. (Effectively, the number of fingers a user would use to make the specified gesture.) The default touch count value is 1.
`startOffset`	The first offset to use for a multiple-point gesture. The default value is {x:0.0, y:0.0}. See the discussion for details.
`endOffset`	The last offset to use for a multiple-point gesture. The default value is {x:0.0, y:0.0}. See the discussion for details.

Discussion

You can use offsets to achieve finer precision in specifying the hitpoint within the `rect` for the specified element. The offset comprises a pair of x and y values, each ranging from 0.0 to 1.0. These values represent, respectively, relative horizontal and vertical positions within the `rect`, with {x:0.0, y:0.0} as the top left and {x:1.0, y:1.0} as the bottom right. Thus, {x:0.3, y:0.6} specifies a position just below and to the left of center, and {x:1.0, y:0.5} specifies a position centered vertically at the far right.

This example performs a flick just above the bottom edge of the target element, from center to right edge:

```
target.flickInsideWithOptions({startOffset:{x:0.5, y:0.9}, endOffset:{x:1.0,
y:0.9}});
```

hasKeyboardFocus

Determines whether the specified element receives keyboard input.

```
(Number) hasKeyboardFocus()
```

Return Value

Returns 1 if the specified element is the receiver of keyboard input, 0 if not. If the status is not available, it returns null.

hitpoint

Returns the screen position to tap for the specified element.

```
(Point) hitpoint()
```

images

Returns an array of images contained by the specified object.

```
(UIAElementArray) images()
```

isEnabled

Determines whether the specified element is enabled.

```
(Number) isEnabled()
```

Return Value
Returns 1 if the specified element is enabled, 0 if not. If the status is not available, it returns null.

isValid

Returns the specified element's validity status as of the most recent access.

```
(Boolean) isValid()
```

Discussion
Use this method to determine whether the user interface element represented by the specified UIAElement existed as of the last attempt to access it. To be certain that the element exists, use `checkIsValid` instead.

See Also
`checkIsValid`

isVisible

Determines whether the specified element is visible on the screen.

```
(Number) isVisible()
```

Return Value
Returns 1 if the user interface element represented by the specified element is visible on screen, 0 if not. If the status is not available, it returns null.

label

Returns a string containing the label attribute of the element.

```
(String) label()
```

Discussion
This method always returns the label attribute string. (Contrast with the `name` (page 55) method.)

links

Returns an array of links contained by the specified object.

```
(UIAElementArray) links()
```

logElement

Logs information about the specified element.

```
(undefined) logElement()
```

Discussion
This method can be used with any element.

logElementTree

Logs information about the specified element and all of its subelements.

```
(undefined) logElementTree()
```

Discussion
This method can be used with any element.

name

Returns a string containing the name attribute of the element.

```
(String) name()
```

Discussion
The element name is derived from the accessibility attribute of the underlying view. If an identifier attribute string is specified, that string is used as the name; otherwise, the label attribute string is used as the name. Contrast with the `label` (page 54) method.

For more information, see *UIAccessibilityIdentification Protocol Reference* .

navigationBar

Returns the app's navigation bar.

```
(UIAElement) navigationBar()
```

Discussion
This method has been moved up to this class from the UIAWindow Class.

navigationBars

Returns an array of navigation bar objects contained by this object.

```
(UIAElementArray) navigationBars()
```

Discussion
This method has been moved up to this class from the UIAWindow Class.

pageIndicators

Returns an array of page indicators contained by the specified object.

```
(UIAElementArray) pageIndicators()
```

parent

Returns the parent of the specified element.

```
(UIAElement) parent()
```

pickers

Returns an array of picker objects contained by the specified object.

```
(UIAElementArray) pickers()
```

popover

Returns the popover object associated with the specified object, if one exists.

```
(UIAPopover) popover()
```

progressIndicators

Returns an array of progress indicators contained by the specified object.

```
(UIAElementArray) progressIndicators()
```

rect

Returns the position of the object on the main screen.

```
(Rect) rect()
```

Discussion

Your script should treat the `rect` object as a generic JavaScript object whose properties for origin, x, y, size, width, and height correspond to those of the analogous CGRect Cocoa structure. The rect object has the form `{origin:{x:`*xposition*`,y:`*yposition*`}, size:{width:`*widthvalue*`, height:`*heightvalue*`}}`. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

rotateWithOptions

Perform a rotation gesture centered on the specified element.

```
(undefined) rotateWithOptions(Object options)
```

Parameters

`options`

A dictionary that specifies characteristics of the rotation gesture. Valid keys are as follows:

`centerOffset`	The offset to use for the center of the rotate gesture. The default offset value is {x:0.0, y:0.0}.
`duration`	The length of hold time for the specified gesture, in seconds. The default duration value is 1.
`radius`	The distance in points from the center to the edge of the circular path.
`rotation`	The length of rotation in radians. The default is pi (ffl).
`touchCount`	The number of touches to use in the specified gesture. (Effectively, the number of fingers a user would use to make the specified gesture.) Valid values are 1 to 5. The default is 2.

Discussion

This gesture is generated such that each touch is equidistant from the others.

scrollToVisible

Scrolls until the specified element is visible in a container view.

```
(undefined) scrollToVisible()
```

Discussion
Use this method with tables and web views.

scrollViews

Returns an array of scroll views contained by the specified object.

```
(UIAElementArray) scrollViews()
```

searchBars

Returns an array of search bars contained by the specified object.

```
(UIAElementArray) searchBars()
```

secureTextFields

Returns an array of secure text fields contained by the specified object.

```
(UIAElementArray) secureTextFields()
```

segmentedControls

Returns an array of segmented controls contained by the specified object.

```
(UIAElementArray) segmentedControls()
```

sliders

Returns an array of sliders contained by the specified object.

```
(UIAElementArray) sliders()
```

staticTexts

Returns an array of static texts contained by the specified object.

```
(UIAElementArray) staticTexts()
```

switches

Returns an array of switches contained by the specified object.

```
(UIAElementArray) switches()
```

tabBar

Returns the specified tab bar.

```
(UIAElement) tabBar()
```

Discussion
This method has been moved up to this class from the UIAWindow Class.

tabBars

Returns an array of tab bars contained by this object.

```
(UIAElementArray) tabBars()
```

Discussion
This method has been moved up to this class from the UIAWindow Class.

tableViews

Returns an array of table views contained by the specified object.

```
(UIAElementArray) tableViews()
```

tap

Taps the specified element.

```
(undefined) tap()
```

tapWithOptions

Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.

```
(undefined) tapWithOptions(Object options)
```

Parameters

`options`

A dictionary that specifies characteristics of the gesture. Valid keys are as follows:

`tapCount`	The number of taps that compose the specified gesture. The default value is 1 (single tap).
`touchCount`	The number of touches to use in the specified gesture. (Effectively, the number of fingers a user would use to make the specified gesture.) The default touch count value is 1.
`duration`	The length of hold time for the specified gesture. The default duration value for a tap is 0. The default value for touch-and-hold gestures (such as drag, pinch open, and pinch close) is 1.
`tapOffset`	The offset to use for the specified tap gesture. The default offset value is {x:0.0, y:0.0}. See the discussion for details.

Discussion

For example, you could specify a triple tap with two fingers at the center of the screen (on an iPhone in portrait orientation), as follows:

```
element.tapWithOptions({touchCount:2, tapCount:3});

element.tapWithOptions({touchCount:2, tapCount:3, tapOffset:{x:0.75, y:0.25}});
```

You can use offsets to achieve finer precision in specifying the hitpoint within the `rect` for the specified element. The offset comprises a pair of x and y values, each ranging from 0.0 to 1.0. These values represent, respectively, relative horizontal and vertical positions within the `rect`, with {x:0.0, y:0.0} as the top left and {x:1.0, y:1.0} as the bottom right. Thus, {x:0.3, y:0.6} specifies a position just below and to the left of center, and {x:1.0, y:0.5} specifies a position centered vertically at the far right.

textFields

Returns an array of text fields contained by the specified object.

```
(UIAElementArray) textFields()
```

textViews

Returns an array of text views contained by the specified object.

```
(UIAElementArray) textViews()
```

toolbar

Returns the specified toolbar.

```
(UIAElement) toolbar()
```

Discussion
This method has been moved up to this class from the UIAWindow Class.

toolbars

Returns an array of toolbars contained by this object.

```
(UIAElementArray) toolbars()
```

Discussion
This method has been moved up to this class from the UIAWindow Class.

touchAndHold

Touches the specified element and holds for the specified duration.

```
(undefined) touchAndHold(Number duration)
```

Parameters
`duration`
The length of time to hold the touch on the element, in seconds.The default duration value for a tap is 0. The default value for touch-and-hold gestures (such as drag, pinch open, and pinch close) is 1.

twoFingerTap

Performs a two-finger (two-touch) tap on this element.

```
(undefined) twoFingerTap()
```

value

Returns a string containing a value attribute specific to the type of element.

```
(String) value()
```

Discussion
For example, a switch has a value of `1` for ON an `0` for OFF.

waitForInvalid

Waits for the specified element to become invalid.

```
(Boolean) waitForInvalid()
```

Discussion
Waits for the user interface element represented by the specified UIAElement to become invalid. Uses the current `timeout` value for the wait time interval.

webViews

Returns an array of web views contained by the specified object.

```
(UIAElementArray) webViews()
```

withName

Returns an element whose name attribute matches a specified string.

```
(UIAElement) withName(String name)
```

Parameters
`name`

A string containing the name to test for.

Discussion
Tests if the name attribute of the element has the given string value. If the match fails, the test is retried until the current `timeout` expires.

withPredicate

Returns the element matching the specified criteria.

```
(UIAElement) withPredicate(PredicateString predicateString)
```

Parameters

`predicateString`

A string specifying the match criteria.

Discussion

Uses the specified predicate string to test for a match. If the match fails, the test is retried until the current `timeout` expires. See *Predicate Programming Guide* for information about using predicates.

withValueForKey

Returns the element containing the specified property with the specified value.

```
(UIAElement) withValueForKey(NotTyped value, String key)
```

Parameters

`value`

A string specifying the value that the specified property, if it exists, should match.

`key`

A string specifying the property to test for.

Discussion

Tests if the element has a specified property with the specified value. If the match fails, the test is retried until the current `timeout` expires.

UIAElementArray Class Reference

Overview

The `UIAElementArray` class supports operations with arrays containing multiple UIAElement objects. You can search the array by name or key/value pairs, or by custom criteria that you specify using a predicate. For detailed information about using predicates, see *Predicates Programming Guide*.

UIAElementArray supports the traditional shorthand syntax for accessing items in native JavaScript objects:

- dot reference by element name, for example:

  ```
  var okButton = buttons.OK;
  ```

- bracket reference by index, for example:

  ```
  var firstElement = elements[0];
  ```

- bracket reference by element name, for example:

  ```
  var helloWorldText = staticTexts["Hello World"];
  ```

It is important to note one potentially confusing limitation in using bracket references by element name. If the name of an element happened to be a number, JavaScript would interpret that name as an index, likely yielding incorrect results. For example, consider this array:

```
var elements = ["DoIt", "Cancel", "1"];
```

If you attempt to access the third element by name with the bracket reference syntax:

```
elements()["1"]
```

it is actually treated as a reference to the first element, yielding the first element (named “DoIt”) instead. In cases where this problem might occur, you should use the dot reference syntax instead:

```
elements().firstWithName("1")
```

For an explanation of how to use this class and related classes, see “Automating UI Testing” in *Instruments User Guide*.

Tasks

Working With Arrays

`length` (page 65)
Returns the number of user interface elements in the array.

`firstWithName` (page 65)
Returns the first element in the array with this name.

`firstWithPredicate` (page 66)
Returns the first element in the array matching the given criteria.

`firstWithValueForKey` (page 66)
Returns the first element in the array with a value that matches the property key.

`toArray` (page 66)
Converts the array into a standard JavaScript array.

`withName` (page 66)
Returns all elements in the array with this name.

`withPredicate` (page 66)
Returns all elements in the array matching the given.

`withValueForKey` (page 67)
Returns all elements in the array with a value that matches the property key.

Properties

length

Returns the number of user interface elements in the array.

```
(Number) length
```

Methods

firstWithName

Returns the first element in the array with this name.

```
(UIAElement) firstWithName(String name)
```

Parameters

`name`

A string whose value is the name of the element match on.

firstWithPredicate

Returns the first element in the array matching the given criteria.

```
(UIAElement) firstWithPredicate(PredicateString predicateString)
```

Parameters

`predicateString`

A predicate specifying the criteria to match.

Discussion

For detailed information about predicate matching, see *Predicate Programming Guide*.

firstWithValueForKey

Returns the first element in the array with a value that matches the property key.

```
(UIAElement) firstWithValueForKey(NotTyped value, String key)
```

toArray

Converts the array into a standard JavaScript array.

```
(Array) toArray()
```

withName

Returns all elements in the array with this name.

```
(UIAElementArray) withName(String name)
```

withPredicate

Returns all elements in the array matching the given.

```
(UIAElementArray) withPredicate(PredicateString predicateString)
```

Discussion

Predicate matching follows the same rules as `NSPredicate`.

withValueForKey

Returns all elements in the array with a value that matches the property key.

```
(UIAElementArray) withValueForKey(NotTyped value, String key)
```

Parameters

`value`

A string specifying the value that the specified property, if it exists, should match.

`key`

A string specifying the property to test for.

Constants

UIAElementNil

Constants

`UIAElementNil`

Returned by a function with return type `UIAElement` or `UIAElementArray` if the requested element is not available after the `timeout` grace period you specify. This mechanism allows your scripting expressions to complete even if an intermediate function in the expression fails temporarily during the grace period. For example, the following code does not raise an exception if the navigation bar does not exist; instead it returns `UIAElementNil`.

```
var backButton =
UIATarget.localTarget().frontMostApp().navigationBar().buttons()["Back"];
if (backButton.isValid()) {
    backButton.tap();
} else {
    UIALogger.logError("Could not find 'Back' button!");
}
```

UIAHost Class Reference

Overview

The `UIAHost` class allows your script to exercise limited control over the Automation instrument process running on the host computer.

For an explanation of how to use this and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Tasks

Performing a Task on the Host Computer

`performTaskWithPathArgumentsTimeout` (page 68)

Executes a task from the Automation instrument process running on the host.

Methods

performTaskWithPathArgumentsTimeout

Executes a task from the Automation instrument process running on the host.

```
(object) performTaskWithPathArgumentsTimeout(path, args, timeout)
```

Parameters

`path`

The pathname of the code to run, relative to the root level of the host's boot drive.

`args`

An array that specifies the arguments for the code to be run.

`timeout`

The length, in seconds, of the grace period in which the task is expected to execute, before script execution resumes.

Discussion

The process executes from within the context of the Instruments application parent process. The code below runs the `echo` command to display "Hello World" with a grace period of 5 seconds.

The returned object contains the properties `exitCode`, `stdout`, and `stderr`. The example uses these properties to capture the exit code, standard output stream, and standard error stream, displaying each in a log message with a `debug` severity level.

```
var target = UIATarget.localTarget();
var host = target.host();


var result = host.performTaskWithPathArgumentsTimeout("/usr/bin/echo", ["Hello
World"], 5);


UIALogger.logDebug("exitCode: " + result.exitCode);
UIALogger.logDebug("stdout: " + result.stdout);
UIALogger.logDebug("stderr: " + result.stderr);
```

UIAKey Class Reference

Inherits from	UIAElement
Availability	Available in iOS 4.0 and later.

Overview

The `UIAKey` class allows access to, and control, of key elements within your app's keyboard.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 12-1 provides a list of methods inherited from `UIAElement`.

Table 12-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.

Method	Description
`elements` (page 52)	Returns an array of elements contained by the specified object.
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.

Method	Description
`rect` (page 57)	Returns the position of the object on the main screen.
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.

Method	Description
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIAKeyboard Class Reference

Inherits from	UIAElement

Overview

The `UIAKeyboard` class allows access to, and control of, elements within your app's keyboard.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 13-1 provides a list of methods inherited from `UIAElement`.

Table 13-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Retrieving Keyboard Information

`keys` (page 77)

Returns an array representing the keys of the specified keyboard.

Exercising the Keyboard

`typeString` (page 78)

Taps the keys of the specified keyboard as required to generate the specified string.

Methods

keys

Returns an array representing the keys of the specified keyboard.

```
(UIAElementArray) keys()
```

typeString

Taps the keys of the specified keyboard as required to generate the specified string.

```
(undefined) typeString(String string)
```

Parameters

`string`

The string to be typed on the keyboard.

UIALink Class Reference

Inherits from	UIAElement

Overview

The `UIALink` class allows access to, and control of, link elements.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 14-1 provides a list of methods inherited from `UIAElement`.

Table 14-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Retrieving Link Information

`url` (page 82)
Returns a string containing a URL.

Methods

url

Returns a string containing a URL.

```
(String) url()
```

UIALogger Class Reference

Overview

The `UIALogger` class provides test and error information on retrieval functionality.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Tasks

Logging With Test Status

`logFail` (page 84)
Logs a message and indicates a test has completed unsuccessfully.

`logIssue` (page 84)
Logs a message and indicates a test has terminated abnormally.

`logPass` (page 85)
Logs a message and indicates a test has completed successfully.

`logStart` (page 85)
Logs a message and indicates a test has started.

Logging With Severity Levels

These methods log a message and set a severity level to support filtering in the detail pane.

`logDebug` (page 84)
Logs the specified message and sets the severity level to `debug`.

`logError` (page 84)
Logs the specified message and sets the severity level to `error`.

`logMessage` (page 85)

Logs the specified message and sets the severity level to `message`.

`logWarning` (page 85)

Logs the specified message and sets the severity level to `warning`.

Methods

logDebug

Logs the specified message and sets the severity level to `debug`.

```
(undefined) logDebug(String message)
```

Parameters

`message`

A string containing the message to log.

logError

Logs the specified message and sets the severity level to `error`.

```
(undefined) logError(String message)
```

Parameters

`message`

A string containing the message to log.

logFail

Logs a message and indicates a test has completed unsuccessfully.

```
(undefined) logFail(String message)
```

logIssue

Logs a message and indicates a test has terminated abnormally.

```
(undefined) logIssue(String message)
```

Parameters

`message`

A string containing the message to log.

logMessage

Logs the specified message and sets the severity level to `message`.

```
(undefined) logMessage(String message)
```

Parameters

`message`

A string containing the message to log.

logPass

Logs a message and indicates a test has completed successfully.

```
(undefined) logPass(String message)
```

Parameters

`message`

A string containing the message to log.

logStart

Logs a message and indicates a test has started.

```
(undefined) logStart(String message)
```

Parameters

`message`

A string containing the message to log.

logWarning

Logs the specified message and sets the severity level to `warning`.

```
(undefined) logWarning(String message)
```

Parameters

`message`

A string containing the message to log.

UIANavigationBar Class Reference

Inherits from	UIAElement

Overview

The `UIANavigationBar` class allows access to, and control of, buttons in your app's navigation bar.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 16-1 provides a list of methods inherited from `UIAElement`.

Table 16-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Accessing Buttons

`leftButton` (page 90)
: Returns the left button in the navigation bar.

`rightButton` (page 90)
: Returns the right button in the navigation bar.

Methods

leftButton

Returns the left button in the navigation bar.

```
(UIAButton) leftButton()
```

rightButton

Returns the right button in the navigation bar.

```
(UIAButton) rightButton()
```

UIAPageIndicator Class Reference

Inherits from	UIAElement

Overview

The `UIAPageIndicator` class allows access to, and control of, page indicator elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 17-1 provides a list of methods inherited from `UIAElement`.

Table 17-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Handling Page Navigation

`goToNextPage` (page 94)
: Goes to the next (logically, to the right) open view .

`goToPreviousPage` (page 95)
: Goes to the previous (logically, to the left) open view .

`pageCount` (page 95)
: Returns the number of open views.

`pageIndex` (page 95)
: Returns the index of the currently open view.

`selectPage` (page 95)
: Goes to the page specified by the index value.

Methods

goToNextPage

Goes to the next (logically, to the right) open view .

```
(undefined) goToNextPage()
```

goToPreviousPage

Goes to the previous (logically, to the left) open view .

```
(undefined) goToPreviousPage()
```

pageCount

Returns the number of open views.

```
(Number) pageCount()
```

pageIndex

Returns the index of the currently open view.

```
(Number) pageIndex()
```

selectPage

Goes to the page specified by the index value.

```
(undefined) selectPage(Number index)
```

Parameters

`index`

The value identifying the selected page.

UIAPicker Class Reference

Inherits from	UIAElement

Overview

The `UIAPicker` class allows access to, and control of, wheel elements within a picker.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 18-1 provides a list of methods inherited from `UIAElement`.

Table 18-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
value (page 62)	Returns a string containing a value attribute specific to the type of element.
waitForInvalid (page 62)	Waits for the specified element to become invalid.
webViews (page 62)	Returns an array of web views contained by the specified object.
withName (page 62)	Returns an element whose name attribute matches a specified string.
withPredicate (page 62)	Returns the element matching the specified criteria.
withValueForKey (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Manipulating Pickers

wheels (page 99)

Returns an array representing the wheels of the specified picker.

Methods

wheels

Returns an array representing the wheels of the specified picker.

```
(UIAElementArray) wheels()
```

UIAPickerWheel Class Reference

Inherits from	UIAPicker

Overview

The `UIAPickerWheel` class allows access to, and control of, wheel elements within a picker.

For an explanation of how to use this class and related classes, see the UI Automation section of "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 19-1 provides a list of methods inherited from `UIAPicker`.

Table 19-1 Methods inherited from `UIAPicker`

Method	Description
`wheels` (page 99)	Returns an array representing the wheels of the specified picker.

Tasks

Manipulating Wheels

`selectValue` (page 101)
Drags the wheel to the first row with the specified value.

`values` (page 101)
Returns an array representing the possible item values to select for the wheel.

Methods

selectValue

Drags the wheel to the first row with the specified value.

```
(undefined) selectValue()
```

Special Considerations

This method is unsupported for UIAPickerWheel objects backed by a UIADatePicker view in iOS 5 and earlier. Since picker wheels representing years and eras have hundreds of thousands of values, you can only set values that are between the min and max supported values.

values

Returns an array representing the possible item values to select for the wheel.

```
(Array) values()
```

Special Considerations

This method is unsupported for UIAPickerWheel objects backed by a UIADatePicker view in iOS 5 and earlier; in such cases, it returns nil. Since picker wheels representing years and eras have hundreds of thousands of values, only the min and max supported values are returned in these cases.

UIAPopover Class Reference

Inherits from	UIAElement
Availability	Available in iOS 4.0 and later.

Overview

The `UIAPopover` class provides methods for accessing and manipulating popovers and the elements they contain.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 20-1 provides a list of methods inherited from `UIAElement`.

Table 20-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.

Method	Description
`elements` (page 52)	Returns an array of elements contained by the specified object.
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.

Method	Description
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Retrieving Popover Contents

`actionSheet` (page 105)
Returns the action sheet contained by the popover.

`navigationBar` (page 106)
Returns the navigation bar contained by the popover.

`tabBar` (page 106)
Returns the tab bar contained by the popover.

`toolbar` (page 106)
Returns the toolbar contained by the popover.

Dismissing the Popover

`dismiss` (page 106)
Dismisses a popover by tapping outside the popover and within the region defined for dismissal.

Methods

actionSheet

Returns the action sheet contained by the popover.

```
(UIAActionSheet) actionSheet()
```

dismiss

Dismisses a popover by tapping outside the popover and within the region defined for dismissal.

```
(void) dismiss()
```

navigationBar

Returns the navigation bar contained by the popover.

```
(UIANavigationBar) navigationBar()
```

tabBar

Returns the tab bar contained by the popover.

```
(UIATabBar) tabBar()
```

toolbar

Returns the toolbar contained by the popover.

```
(UIAToolbar) toolbar()
```

UIAProgressIndicator Class Reference

Inherits from	UIAElement

Overview

The `UIAProgressIndicator` class allows access to, and control of, progress indicator elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 21-1 provides a list of methods inherited from `UIAElement`.

Table 21-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIAScrollView Class Reference

Inherits from	UIAElement

Overview

The `UIAScrollView` class allows access to, and control of, the elements of a scroll view.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 22-1 provides a list of methods inherited from `UIAElement`.

Table 22-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Scrolling

`scrollUp` (page 116)
Scrolls up within the specified scroll view.

`scrollDown` (page 115)
Scrolls down within the specified scroll view.

`scrollLeft` (page 115)
Scrolls left within the specified scroll view.

`scrollRight` (page 115)
Scrolls right within the specified scroll view.

`scrollToElementWithName` (page 115)
Scrolls within the specified scroll view until the named element is displayed on the screen.

`scrollToElementWithPredicate` (page 115)
Scrolls within the specified scroll view until the matching element is displayed on the screen.

`scrollToElementWithValueForKey` (page 116)
Scrolls within the specified scroll view until the element with the specified value for the specified key is displayed on the screen.

Methods

scrollDown

Scrolls down within the specified scroll view.

```
(undefined) scrollDown()
```

scrollLeft

Scrolls left within the specified scroll view.

```
(undefined) scrollLeft()
```

scrollRight

Scrolls right within the specified scroll view.

```
(undefined) scrollRight()
```

scrollToElementWithName

Scrolls within the specified scroll view until the named element is displayed on the screen.

```
(UIAElement) scrollToElementWithName(String name)
```

Parameters

`name`

The name of the element to scroll to.

scrollToElementWithPredicate

Scrolls within the specified scroll view until the matching element is displayed on the screen.

```
(UIAElement) scrollToElementWithPredicate(PredicateString predicateString)
```

Parameters

`predicateString`

The predicate to define the match criteria.

scrollToElementWithValueForKey

Scrolls within the specified scroll view until the element with the specified value for the specified key is displayed on the screen.

```
(UIAElement) scrollToElementWithValueForKey(NotTyped value, String key)
```

Parameters

`value`

The value for the specified key.

`key`

The key for the specified value.

scrollUp

Scrolls up within the specified scroll view.

```
(undefined) scrollUp()
```

UIASearchBar Class Reference

Inherits from	UIATextField

Overview

The `UIASearchBar` class allows access to, and control of, search bar elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 23-1 provides a list of methods inherited from `UIATextField`.

Table 23-1 Methods inherited from `UIATextField`

Method	Description
`setValue` (page 175)	Sets the specified text field to the specified value.

UIASecureTextField Class Reference

Inherits from	UIATextField

Overview

The `UIASecureTextField` class allows access to, and control of, secure text field elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 24-1 provides a list of methods inherited from `UIATextField`.

Table 24-1 Methods inherited from `UIATextField`

Method	Description
`setValue` (page 175)	Sets the specified text field to the specified value.

UIASegmentedControl Class Reference

Inherits from	UIAElement

Overview

The `UIASegmentedControl` class allows access to, and control of, elements within segmented controls in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 25-1 provides a list of methods inherited from `UIAElement`.

Table 25-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Accessing Buttons

`selectedButton` (page 122)

Returns the currently selected button within the segmented control.

Methods

selectedButton

Returns the currently selected button within the segmented control.

```
(UIAElement) selectedButton()
```

UIASlider Class Reference

Inherits from	UIAElement

Overview

The `UIASlider` class allows access to, and control of, slider elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 26-1 provides a list of methods inherited from `UIAElement`.

Table 26-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Setting Slider Values

`dragToValue` (page 126)

Drags the slider to the specified value.

Methods

dragToValue

Drags the slider to the specified value.

```
(undefined) dragToValue(Number value)
```

Parameters

`value`

The desired decimal value from 0 to 1, inclusive. A 0 value represents far left and a value of 1 represents far right.

UIAStaticText Class Reference

Inherits from	UIAElement

Overview

The `UIAStaticText` class allows access to, and control of, static text views.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 27-1 provides a list of methods inherited from `UIAElement`.

Table 27-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIAStatusBar Class Reference

Inherits from	UIAElement

Overview

The `UIAStatusBar` class allows access to, and control of, your app's status bar.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 28-1 provides a list of methods inherited from `UIAElement`.

Table 28-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIASwitch Class Reference

Inherits from	UIAElement

Overview

The `UIASwitch` class allows access to, and control of, switch elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 29-1 provides a list of methods inherited from `UIAElement`.

Table 29-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Setting Switch Values

`setValue` (page 138)
Sets the specified switch to the specified value.

Methods

setValue

Sets the specified switch to the specified value.

```
(undefined) setValue(Boolean value)
```

Parameters

`value`
A boolean value to represent the desired state, `true` for On, `false` for Off.

UIATabBar Class Reference

Inherits from	UIAElement

Overview

The `UIAElement` class allows access to, and control of, elements within your app's tab bar.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 30-1 provides a list of methods inherited from `UIAElement`.

Table 30-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Selecting a Button

`selectedButton` (page 142)

Returns the button currently selected in the tab bar.

Methods

selectedButton

Returns the button currently selected in the tab bar.

```
(UIAButton) selectedButton()
```

UIATableCell Class Reference

Inherits from	UIAElement
Availability	Available in iOS 4.0 and later.

Overview

The `UIATableCell` class allows access to, and control of, table cell elements within a table view.

> **Important:** In certain test scenarios, particularly with new off-screen table cell elements, UI Automation is unable to immediately derive a label for an element, rendering that element inaccessible to your script. To avoid resultant problems with automated tests, your application should always set the `accessibilityLabel` property for new table cell elements. See *UIAccessibilityElement Class Reference* for more information.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide* .

Inherited Methods

Table 31-1 provides a list of methods inherited from `UIAElement`.

Table 31-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.

Method	Description
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.

Method	Description
pickers (page 56)	Returns an array of picker objects contained by the specified object.
popover (page 56)	Returns the popover object associated with the specified object.
progressIndicators (page 57)	Returns an array of progress indicators contained by the specified object.
rect (page 57)	Returns the position of the object on the main screen.
rotateWithOptions (page 57)	Perform a rotation gesture centered on the specified element.
scrollToVisible (page 58)	Scrolls until the specified element is visible in a container view.
scrollViews (page 58)	Returns an array of scroll views contained by the specified object.
searchBars (page 58)	Returns an array of search bars contained by the specified object.
secureTextFields (page 58)	Returns an array of secure text fields contained by the specified object.
segmentedControls (page 58)	Returns an array of segmented controls contained by the specified object.
sliders (page 58)	Returns an array of sliders contained by the specified object.
staticTexts (page 59)	Returns an array of static texts contained by the specified object.
switches (page 59)	Returns an array of switches contained by the specified object.
tabBar (page 59)	Returns the specified tab bar.
tabBars (page 59)	Returns an array of tab bars contained by this object.
tableViews (page 59)	Returns an array of table views contained by the specified object.
tap (page 59)	Taps the specified element.
tapWithOptions (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
textFields (page 60)	Returns an array of text fields contained by the specified object.
textViews (page 61)	Returns an array of text views contained by the specified object.

Method	Description
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIATableGroup Class Reference

Inherits from	UIAElement

Overview

The `UIATableGroup` class allows access to, and control of, group elements within a table view.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 32-1 provides a list of methods inherited from `UIAElement`.

Table 32-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIATableView Class Reference

Inherits from	UIAScrollView

Overview

The `UIATableView` class allows access to, and control of, elements within a table view in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 33-1 provides a list of methods inherited from `UIAScrollView`.

Table 33-1 Methods inherited from `UIAScrollView`

Method	Description
`scrollDown` (page 115)	Scrolls down within the specified table view.
`scrollLeft` (page 115)	Scrolls left within the specified table view.
`scrollRight` (page 115)	Scrolls right within the specified table view.
`scrollToElementWithName` (page 115)	Scrolls within the table view until the named element is displayed on the screen.
`scrollToElementWithPredicate` (page 115)	Scrolls within the table view until the matching element is displayed on the screen.
`scrollToElementWithValueForKey` (page 116)	Scrolls within the table view until the element with the specified value for the specified key is displayed on the screen.
`scrollUp` (page 116)	Scrolls up within the specified table view.

Tasks

Retrieving Information

`cells` (page 152)

Returns an array of the cells within the table view.

`groups` (page 152)

Returns an array of the groups within the table view.

`visibleCells` (page 152)

Returns an array of the cells that are visible within the table view.

Methods

cells

Returns an array of the cells within the table view.

```
(UIAElementArray) cells()
```

groups

Returns an array of the groups within the table view.

```
(UIAElementArray) groups()
```

visibleCells

Returns an array of the cells that are visible within the table view.

```
(UIAElementArray) visibleCells()
```

UIATarget Class Reference

Availability	Available in iOS 4.0 and later.

Overview

The `UIATarget` class represents high-level user interface elements of the system under test (SUT)—that is, your app, the iOS, and the connected device on which they're running. Your test scripts, written in JavaScript and running in conjunction with the UI Automation instrument, use this class and related UI Automation classes to exercise the SUT and log results.

For the sake of simplicity and consistency with other Apple documentation, this document describes device operations and user interface actions as though they were performed by a user. In practice, the Automation instrument simulates these operations and actions.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Tasks

Getting the Base Target and Host Objects

`host` (page 161)
Returns an object representing the machine that is host to the current target.

`localTarget` (page 161)
Returns an object representing the system under test (SUT).

Managing Your App

`deactivateAppForDuration` (page 158)
Renders your app inactive for the specified duration.

`frontMostApp` (page 161)
Returns an object representing your app.

Obtaining Device Property Information

Use these methods to obtain information specific to the device, such as assigned name, device model, and operating-system name and version.

`model` (page 162)
Returns the device model.

`name` (page 162)
Returns the device name.

`rect` (page 164)
Returns the rectangle surrounding the device's main screen.

`systemName` (page 168)
Returns the name of the operating system running on the device.

`systemVersion` (page 168)
Returns the current version of the operating system running on the device.

Determining and Changing Device Orientation

`deviceOrientation` (page 159)
Returns the current orientation of the device.

`setDeviceOrientation` (page 165)
Changes the device orientation to the specified new `deviceOrientation` value.

Changing the Device Location

`setLocation` (page 166)
Specifies a change in device's latitude and longitude.

`setLocationWithOptions` (page 166)
Specifies a change in the device's latitude, longitude, and other characteristics.

Device Controls and Actions

`clickVolumeDown` (page 158)
Presses the volume down hardware button.

`clickVolumeUp` (page 158)
Presses the volume up hardware button.

`holdVolumeDown` (page 161)
Holds down the volume down hardware button for the specified duration.

`holdVolumeUp` (page 161)
Presses and holds the volume up hardware button for the specified duration.

`lockForDuration` (page 162)
Locks the device, using a lock event, for the specified duration.

`lock` (page 161)
Locks the device, using a lock event.

`shake` (page 167)
Simulates a shake action on the device.

`unlock` (page 170)
Unlocks the device using an unlock event followed by a drag of the slider.

Interacting with the Screen

The `rect` and `point` objects used with these screen interaction methods have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

`dragFromToForDuration` (page 160)
Drags from a specified starting screen location to a specified ending screen location, for a specified length of time.

`doubleTap` (page 159)
Double-taps the specified element or at the specified screen location.

`flickFromTo` (page 160)
Flicks from the specified starting screen location to the specified ending screen location.

`pinchCloseFromToForDuration` (page 162)

Pinches (performs a pinch-close gesture) from a specified starting screen location to a specified ending screen location, for a specified length of time.

`pinchOpenFromToForDuration` (page 163)

Stretches (performs a pinch-open gesture) from a specified starting screen location to a specified ending screen location, for a specified length of time.

`rotateWithOptions` (page 164)

Performs a rotation gesture at the specified location.

`tap` (page 168)

Taps the specified element or the specified screen location.

`tapWithOptions` (page 168)

Taps the specified element with the specified options.

`touchAndHold` (page 169)

Touches the specified element, or the specified screen location, and holds for the specified duration.

Capturing Screen Images

These methods allow you to record the appearance of the screen (or some portion of it). Such images can be helpful in tracking progress in a test and in diagnosing problems.

`captureRectWithName` (page 157)

Takes a screen shot of the specified rectangular portion of the device screen.

`captureScreenWithName` (page 157)

Takes a screen shot of the entire device screen.

Manipulating Timeouts

`popTimeout` (page 163)

Retrieves the previous timeout value from a stack, restores it as the current timeout value, and returns it.

`pushTimeout` (page 164)

Stores the current timeout value on a stack and sets a new timeout value.

`setTimeout` (page 167)

Sets a new timeout value.

`timeout` (page 169)

Returns the current timeout value.

Miscellaneous

`delay` (page 158)

Delays script execution for the specified time.

Methods

captureRectWithName

Takes a screen shot of the specified rectangular portion of the device screen.

```
(undefined) captureRectWithName(Rect rect, String imageName)
```

Parameters

`rect`

The `rect` that defines the area of the screen to capture.

`imageName`

A string to use as the name for the resultant image file.

Discussion

Your script should treat the `rect` object as a generic JavaScript object whose properties for origin, x, y, size, width, and height correspond to those of the analogous CGRect Cocoa structure. The rect object has the form `{origin:{x:`*xposition*`,y:`*yposition*`}, size:{width:`*widthvalue*`, height:`*heightvalue*`}}`. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

The image is saved as a file in .PNG graphic format, with the specified name, in the log.

captureScreenWithName

Takes a screen shot of the entire device screen.

```
(undefined) captureScreenWithName(String imageName)
```

Parameters

`imageName`

A string to use as the name for the resultant image file.

Discussion
The image is saved as a file in .PNG graphic format, with the specified name, in the log.

clickVolumeDown

Presses the volume down hardware button.

```
(undefined) clickVolumeDown()
```

clickVolumeUp

Presses the volume up hardware button.

```
(undefined) clickVolumeUp()
```

deactivateAppForDuration

Renders your app inactive for the specified duration.

```
(Boolean) deactivateApp(Number duration)
```

Parameters
`duration`
The time, in seconds, for the app to remain inactive.

Discussion
Use this method to test shifting your app to and from the background execution context. Note that apps built using iOS SDK 4.0 or later and running in iOS 4.0 and later aren't necessarily terminated when the user presses the Home button. See *iOS App Programming Guide* for details of multitasking and background execution context.

delay

Delays script execution for the specified time.

```
(Boolean) delay(Number timeInterval)
```

Parameters
`timeInterval`
The time to delay, in seconds.

Discussion
You can use this method to provide enough time for lengthy operations to complete.

deviceOrientation

Returns the current orientation of the device.

```
(Number deviceOrientation) deviceOrientation()
```

Discussion
The returned value is a constant that represents the physical orientation of the device and may be different from the current orientation of your app's user interface. The possible values are as follows:

```
UIA_DEVICE_ORIENTATION_UNKNOWN
UIA_DEVICE_ORIENTATION_PORTRAIT
UIA_DEVICE_ORIENTATION_PORTRAIT_UPSIDEDOWN
UIA_DEVICE_ORIENTATION_LANDSCAPELEFT
UIA_DEVICE_ORIENTATION_LANDSCAPERIGHT
UIA_DEVICE_ORIENTATION_FACEUP
UIA_DEVICE_ORIENTATION_FACEDOWN
```

See the Constants section for descriptions of these values.

doubleTap

Double-taps the specified element or at the specified screen location.

```
(undefined) doubleTap(Object tapPointObject)
```

Parameters
`tapPointObject`
A `rect`, `point`, or `UIAElement`.

Discussion
The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

dragFromToForDuration

Drags from a specified starting screen location to a specified ending screen location, for a specified length of time.

```
(undefined) dragFromToForDuration(fromPointObject, toPointObject, Number duration)
```

Parameters

`fromPointObject`

The `rect` or `point` from which the drag action is to begin.

`toPointObject`

The `rect` or `point` at which the drag action is to end.

`duration`

The length of time, in seconds, between starting and stopping the gesture.

Discussion

The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

flickFromTo

Flicks from the specified starting screen location to the specified ending screen location.

```
(undefined) flickFromTo(fromPointObject, toPointObject)
```

Parameters

`fromPointObject`

The `rect` or `point` from which the flick action is to begin.

`toPointObject`

The `rect` or `point` at which the flick action is to end.

Discussion

The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

frontMostApp

Returns an object representing your app.

```
(UIAApplication) frontMostApp()
```

Discussion
This `UIAApplication` object is the centralized point of control and coordination for your app.

holdVolumeDown

Holds down the volume down hardware button for the specified duration.

```
(undefined) holdVolumeDown(Number duration)
```

holdVolumeUp

Presses and holds the volume up hardware button for the specified duration.

```
(undefined) holdVolumeUp(Number duration)
```

host

Returns an object representing the machine that is host to the current target.

```
(UIAHost) host()
```

localTarget

Returns an object representing the system under test (SUT).

```
(UIATarget) localTarget()
```

lock

Locks the device, using a lock event.

```
(undefined) lock()
```

Special Considerations
This method, and its counterpart, `unlock`, are deprecated. Use `lockForDuration` instead.

lockForDuration

Locks the device, using a lock event, for the specified duration.

```
(undefined) lockForDuration(Number duration)
```

Parameters

`Duration`

The length of time, in seconds, for the lock to persist.

Discussion

This method replaces the deprecated `lock` and `unlock` methods.

model

Returns the device model.

```
(String) model()
```

Discussion

Examples of model strings are `iPhone` and `iPod touch`.

name

Returns the device name.

```
(String) name()
```

Discussion

The device name is an arbitrary string specified for the device by the user. On an iPhone, for example, you can see the name on the device in the General > About settings or in iTunes on the Summary > iPhone tab.

pinchCloseFromToForDuration

Pinches (performs a pinch-close gesture) from a specified starting screen location to a specified ending screen location, for a specified length of time.

```
(undefined) pinchCloseFromToForDuration(fromPointObject, toPointObject, Number duration)
```

Parameters

`fromPointObject`

The `rect` or `point` from which the pinch-close action is to begin.

`toPointObject`

The `rect` or `point` at which the pinch-close action is to end.

`duration`

The length of time, in seconds, between starting and stopping the gesture.

Discussion

The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

pinchOpenFromToForDuration

Stretches (performs a pinch-open gesture) from a specified starting screen location to a specified ending screen location, for a specified length of time.

```
(undefined) pinchOpenFromToForDuration(fromPointObject, toPointObject, Number duration)
```

Parameters

`fromPointObject`

The `rect` or `point` from which the pinch-open action is to begin.

`toPointObject`

The `rect` or `point` at which the pinch-open action is to end.

`duration`

The length of time, in seconds, between starting and stopping the gesture.

Discussion

The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

popTimeout

Retrieves the previous timeout value from a stack, restores it as the current timeout value, and returns it.

```
(Number) popTimeout()
```

Return Value

The timeout value last stored on the stack with `pushTimeout`.

Discussion

Use this method to revert to the previous grace period duration.

If an object representing a UI element becomes available within the grace period, an attempt is made to instantiate that object from information retained by the instrument.

pushTimeout

Stores the current timeout value on a stack and sets a new timeout value.

```
(undefined) pushTimeout(timeoutValue)
```

Parameters

`timeout`

The length of the grace period, in seconds.

Discussion

This method, in conjunction with `popTimeout`, allows you to temporarily change the duration of the grace period for object resolution. This code changes the timeout period to 2 seconds before attempting to access an element, then restores the previous timeout period.

```
target = UIATarget.localTarget();

target.pushTimeout(2);
    // attempt element access
target.popTimeout();
```

If an object representing a UI element becomes available within the grace period, an attempt is made to instantiate that object from information retained by the instrument.

rect

Returns the rectangle surrounding the device's main screen.

```
(Rect) rect()
```

rotateWithOptions

Performs a rotation gesture at the specified location.

```
(undefined) rotateWithOptions(Object location, Object options)
```

Parameters

`location`

The `point` object at center of the rotation gesture, with properties for `x` and `y`, corresponding to the analogous `CGPoint` Cocoa structure. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

`options`

A dictionary that specifies characteristics of the rotation gesture. Valid keys are as follows:

`duration`	The length of hold time, in seconds, for the specified gesture. The default duration value is `1`.
`radius`	The distance in points from the center to the edge of the circular path.
`rotation`	The length of rotation in radians. The default is `pi` (ffl).
`touchCount`	The number of touches to use in the specified gesture (effectively, the number of fingers a user would use to make the specified gesture.) Valid values are `1` to `5`. The default is `2`.

Discussion

This gesture is generated such that each touch is equidistant from the others.

setDeviceOrientation

Changes the device orientation to the specified new `deviceOrientation` value.

```
(undefined) setDeviceOrientation(Number deviceOrientation)
```

Discussion

The specified `deviceOrientation` value must be one of the following constants:

```
UIA_DEVICE_ORIENTATION_UNKNOWN
UIA_DEVICE_ORIENTATION_PORTRAIT
UIA_DEVICE_ORIENTATION_PORTRAIT_UPSIDEDOWN
UIA_DEVICE_ORIENTATION_LANDSCAPELEFT
UIA_DEVICE_ORIENTATION_LANDSCAPERIGHT
UIA_DEVICE_ORIENTATION_FACEUP
UIA_DEVICE_ORIENTATION_FACEDOWN
```

See the “Constants” section for descriptions of these values.

setLocation

Specifies a change in device’s latitude and longitude.

```
(boolean) setLocation(coordinates)
```

Parameters

`coordinates`

A dictionary that specifies the new location. Valid keys are as follows:

`latitude`	The latitude in degrees. Positive values indicate latitudes north of the equator. Negative values indicate latitudes south of the equator.
`longitude`	The longitude in degrees. Measurements are relative to the zero meridian, with positive values extending east of the meridian and negative values extending west of the meridian.

setLocationWithOptions

Specifies a change in the device’s latitude, longitude, and other characteristics.

```
(boolean) setLocationWithOptions(coordinates, options)
```

Parameters

`coordinates`

A dictionary that specifies the new location. Valid keys are as follows:

`latitude`	The latitude in degrees. Positive values indicate latitudes north of the equator. Negative values indicate latitudes south of the equator.
`longitude`	The longitude in degrees. Measurements are relative to the zero meridian, with positive values extending east of the meridian and negative values extending west of the meridian.

`options`

A dictionary that specifies additional characteristics of the location change. Valid keys are as follows:

`altitude`	The height, in meters, relative to sea level. Positive values indicate altitudes above sea level. Negative values indicate altitudes below sea level.
`horizontalAccuracy`	The radius, in meters, of the horizontal circle of uncertainty centered at the specified location. Negative values are invalid.
`verticalAccuracy`	The radius, in meters, of the horizontal circle of uncertainty centered at the specified location. Negative values are invalid.
`course`	The direction in which the device is moving, regardless of the device orientation.
`speed`	The speed, in meters per second, at which the device is moving.

setTimeout

Sets a new timeout value.

```
(undefined) setTimeout(Number timeout)
```

Parameters

`timeout`

A number representing the length,in seconds, of the grace period.

Discussion

The timeout value establishes a grace period for object resolution. If an object representing a UI element becomes available within the grace period, an attempt is made to instantiate that object from information retained by the instrument.

shake

Simulates a shake action on the device.

```
(undefined) shake()
```

Discussion

The shake action triggers a `UIEvent` of type `UIEventSubtypeMotionShake`, but does not affect the accelerometer itself.

systemName

Returns the name of the operating system running on the device.

```
(String) systemName()
```

systemVersion

Returns the current version of the operating system running on the device.

```
(String) systemVersion()
```

Discussion

An example of a system version string is `1.2`.

tap

Taps the specified element or the specified screen location.

```
(undefined) tap(Object tapPointObject)
```

Parameters

`tapPointObject`

A `rect`, `point`, or `UIAElement`.

Discussion

The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

tapWithOptions

Taps the specified element with the specified options.

```
(undefined) tapWithOptions(Object tapPointObject, Object options)
```

Parameters

`tapPointObject`

A `rect`, `point`, or `UIAElement`.

`options`

A dictionary that specifies characteristics of the gesture. Valid keys are as follows:

`tapCount`	The number of taps that compose the specified gesture. The default value is `1` (single tap).
`touchCount`	The number of touches to use in the specified gesture. (Effectively, the number of fingers a user would use to make the specified gesture.) The default touch count value is `1`.
`duration`	The length of hold time for the specified gesture. The default duration value for a tap is `0`.

Discussion

The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

timeout

Returns the current timeout value.

```
(Number) timeout()
```

Discussion

The timeout value establishes a grace period for object resolution. If an object representing a UI element becomes available within the grace period, an attempt is made to instantiate that object from information retained by the instrument.

touchAndHold

Touches the specified element, or the specified screen location, and holds for the specified duration.

```
(undefined) touchAndHold(Object tapPointObject, Number duration)
```

Parameters

`tapPointObject`

A `rect`, `point`, or `UIAElement`.

`duration`

The length of time, in seconds, to hold the touch.

Discussion
The `rect` and `point` objects have properties for `origin`, `size`, `x`, `y`, `height`, and `width` corresponding to the analogous `CGRect`, `CGPoint`, and `CGSize` Cocoa structures. Your script should treat methods with `rect`, `point`, or `size` arguments or return types as JavaScript objects with those properties defined. The relevant coordinates are screen-relative and are adjusted to account for device orientation.

unlock

Unlocks the device using an unlock event followed by a drag of the slider.

```
(undefined) unlock()
```

Discussion
Simulating passcode entry is currently unsupported. Set the Settings > General > Passcode Lock feature to Off prior to running your tests.

Special Considerations
This method, and its counterpart, `lock`, are deprecated. Use `lockForDuration` instead.

Event Handlers by Task

Handling Alerts

`onAlert` (page 170)
Called by UI Automation to allow your script to respond to alerts.

Event Handlers

onAlert

Called by UI Automation to allow your script to respond to alerts.

```
(Boolean) onAlert(UIAAlert alert)
```

Parameters
`alert`
An object representing the alert encountered.

Return Value

Returns `true` if successful. Returns `false` to cause the default alert handler to run.

Discussion

Your `onAlert` handler is called if an alert is encountered at any time during the execution of the script. If you do not have a declared `onAlert` handler, the UI Automation default alert handler runs instead.

This default handler attempts to dismiss the alert by first tapping the cancel button, if the button exists, then tapping the default button, if one is identifiable. If the alert is still not dismissed, an exception is thrown.

Returning `false` from your own handler also causes the default handler to run. For cursory tests, the script handler might only log an alert message and return `false` to let the default handler dismiss the alert.

Constants

Constants

`UIA_DEVICE_ORIENTATION_UNKNOWN`
: The orientation of the device cannot be determined.

`UIA_DEVICE_ORIENTATION_PORTRAIT`
: The device is in portrait mode, with the device upright and the home button at the bottom.

`UIA_DEVICE_ORIENTATION_PORTRAIT_UPSIDEDOWN`
: The device is in portrait mode but upside down, with the device upright and the home button at the top.

`UIA_DEVICE_ORIENTATION_LANDSCAPELEFT`
: The device is in landscape mode, with the device upright and the home button on the right side.

`UIA_DEVICE_ORIENTATION_LANDSCAPERIGHT`
: The device is in landscape mode, with the device upright and the home button on the left side.

`UIA_DEVICE_ORIENTATION_FACEUP`
: The device is parallel to the ground with the screen facing upward.

`UIA_DEVICE_ORIENTATION_FACEDOWN`
: The device is parallel to the ground with the screen facing downward.

UIATextField Class Reference

Inherits from	UIAElement

Overview

The `UIATextField` class allows access to, and control of, text field elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 35-1 provides a list of methods inherited from `UIAElement`.

Table 35-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Setting Text Field Values

`setValue` (page 175)

Sets the specified text field to the specified value.

Methods

setValue

Sets the specified text field to the specified value.

```
(undefined) setValue(String value)
```

Parameters

`value`

A string containing the text to populate the text field.

UIATextView Class Reference

Inherits from	UIAElement

Overview

The `UIATextView` class allows access to, and control of, text view elements in your app.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 36-1 provides a list of methods inherited from `UIAElement`.

Table 36-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Setting Text View Values

`setValue` (page 179)

Sets the specified text view to the specified value.

Methods

setValue

Sets the specified text view to the specified value.

```
(undefined) setValue(String value)
```

Parameters

`value`

A string containing the text to populate the text view.

UIAToolbar Class Reference

Inherits from	UIAElement

Overview

The `UIAToolbar` class allows access to, and control of, your app's toolbar.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 37-1 provides a list of methods inherited from `UIAElement`.

Table 37-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`navigationBar` (page 56)	Returns the app's navigation bar.
`navigationBars` (page 56)	Returns an array of navigation bar objects contained by this object.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.

Method	Description
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.
`scrollViews` (page 58)	Returns an array of scroll views contained by the specified object.
`searchBars` (page 58)	Returns an array of search bars contained by the specified object.
`secureTextFields` (page 58)	Returns an array of secure text fields contained by the specified object.
`segmentedControls` (page 58)	Returns an array of segmented controls contained by the specified object.
`sliders` (page 58)	Returns an array of sliders contained by the specified object.
`staticTexts` (page 59)	Returns an array of static texts contained by the specified object.
`switches` (page 59)	Returns an array of switches contained by the specified object.
`tabBar` (page 59)	Returns the specified tab bar.
`tabBars` (page 59)	Returns an array of tab bars contained by this object.
`tableViews` (page 59)	Returns an array of table views contained by the specified object.
`tap` (page 59)	Taps the specified element.
`tapWithOptions` (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
`textFields` (page 60)	Returns an array of text fields contained by the specified object.
`textViews` (page 61)	Returns an array of text views contained by the specified object.
`toolbar` (page 61)	Returns the specified toolbar.
`toolbars` (page 61)	Returns an array of toolbars contained by this object.
`touchAndHold` (page 61)	Touches the specified element and holds for the specified duration.
`twoFingerTap` (page 61)	Performs a two-finger (two-touch) tap on this element.

Method	Description
`value` (page 62)	Returns a string containing a value attribute specific to the type of element.
`waitForInvalid` (page 62)	Waits for the specified element to become invalid.
`webViews` (page 62)	Returns an array of web views contained by the specified object.
`withName` (page 62)	Returns an element whose name attribute matches a specified string.
`withPredicate` (page 62)	Returns the element matching the specified criteria.
`withValueForKey` (page 63)	Returns the element containing the specified property with the specified value.

UIAWebView Class Reference

Inherits from	UIAScrollView

Overview

The `UIAWebView` class allows access to, and control of, web views.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 38-1 provides a list of methods inherited from `UIAScrollView`.

Table 38-1 Methods inherited from `UIAScrollView`

Method	Description
`scrollDown` (page 115)	Scrolls down within the specified collection view.
`scrollLeft` (page 115)	Scrolls left within the specified collection view.
`scrollRight` (page 115)	Scrolls right within the specified collection view.
`scrollToElementWithName` (page 115)	Scrolls within the collection view until the named element is displayed on the screen.
`scrollToElementWithPredicate` (page 115)	Scrolls within the collection view until the matching element is displayed on the screen.
`scrollToElementWithValueForKey` (page 116)	Scrolls within the collection view until the element with the specified value for the specified key is displayed on the screen.
`scrollUp` (page 116)	Scrolls up within the specified collection view.

UIAWindow Class Reference

Inherits from	UIAElement

Overview

The `UIAWindow` class allows access to, and control of, your app's window elements.

For an explanation of how to use this class and related classes, see "Automating UI Testing" in *Instruments User Guide*.

Inherited Methods

Table 39-1 provides a list of methods inherited from `UIAElement`.

Table 39-1 Methods inherited from `UIAElement`

Method	Description
`activityIndicators` (page 50)	Returns an array of the activity indicators contained by the specified object.
`activityView` (page 50)	Returns an object representing an activity view.
`ancestry` (page 50)	Returns an array containing the parents of the specified object.
`buttons` (page 51)	Returns an array of buttons contained by the specified object.
`checkIsValid` (page 51)	Returns the specified element's current validity status.
`collectionViews` (page 51)	Returns an array of collection views contained by the specified object.
`doubleTap` (page 51)	Double-taps the specified element.
`dragInsideWithOptions` (page 51)	Drags within the bounds of an element.
`elements` (page 52)	Returns an array of elements contained by the specified object.

Method	Description
`flickInsideWithOptions` (page 52)	Flicks within the bounds of an element.
`hasKeyboardFocus` (page 53)	Determines whether specified element receives keyboard input.
`hitpoint` (page 53)	Returns the screen position to tap for the specified element.
`images` (page 54)	Returns an array of images contained by the specified object.
`isEnabled` (page 54)	Determines whether the specified element is enabled.
`isValid` (page 54)	Returns the specified element's validity status as of the most recent access.
`isVisible` (page 54)	Determines whether the specified element is visible on the screen.
`label` (page 54)	Returns a string containing the label attribute of the element.
`links` (page 55)	Returns an array of links contained by the specified object.
`logElement` (page 55)	Logs information about the specified element.
`logElementTree` (page 55)	Logs information about the specified element and all of its subelements.
`name` (page 55)	Returns a string containing the name attribute of the element.
`pageIndicators` (page 56)	Returns an array of page indicators contained by the specified object.
`parent` (page 56)	Returns the parent of the specified element.
`pickers` (page 56)	Returns an array of picker objects contained by the specified object.
`popover` (page 56)	Returns the popover object associated with the specified object.
`progressIndicators` (page 57)	Returns an array of progress indicators contained by the specified object.
`rect` (page 57)	Returns the position of the object on the main screen.
`rotateWithOptions` (page 57)	Perform a rotation gesture centered on the specified element.
`scrollToVisible` (page 58)	Scrolls until the specified element is visible in a container view.

Method	Description
scrollViews (page 58)	Returns an array of scroll views contained by the specified object.
searchBars (page 58)	Returns an array of search bars contained by the specified object.
secureTextFields (page 58)	Returns an array of secure text fields contained by the specified object.
segmentedControls (page 58)	Returns an array of segmented controls contained by the specified object.
sliders (page 58)	Returns an array of sliders contained by the specified object.
staticTexts (page 59)	Returns an array of static texts contained by the specified object.
switches (page 59)	Returns an array of switches contained by the specified object.
tableViews (page 59)	Returns an array of table views contained by the specified object.
tap (page 59)	Taps the specified element.
tapWithOptions (page 60)	Performs the specified gesture on the specified element using a dictionary to specify gesture attributes.
textFields (page 60)	Returns an array of text fields contained by the specified object.
textViews (page 61)	Returns an array of text views contained by the specified object.
touchAndHold (page 61)	Touches the specified element and holds for the specified duration.
twoFingerTap (page 61)	Performs a two-finger (two-touch) tap on this element.
value (page 62)	Returns a string containing a value attribute specific to the type of element.
waitForInvalid (page 62)	Waits for the specified element to become invalid.
webViews (page 62)	Returns an array of web views contained by the specified object.
withName (page 62)	Returns an element whose name attribute matches a specified string.
withPredicate (page 62)	Returns the element matching the specified criteria.
withValueForKey (page 63)	Returns the element containing the specified property with the specified value.

Tasks

Working with Window-level Elements

`contentArea` (page 188)

Returns the content area of the window (below the navigation bar and above the tab bar or toolbar.

`navigationBar` (page 188)

Returns the the app's navigation bar.

`navigationBars` (page 188)

Returns an array of the navigation bars contained by this object.

`tabBar` (page 189)

Returns the app's tab bar.

`tabBars` (page 189)

Returns an array of tab bars contained by this object.

`toolbar` (page 189)

Returns the the app's toolbar.

`toolbars` (page 189)

Returns an array of toolbars contained by this object.

Methods

contentArea

Returns the content area of the window (below the navigation bar and above the tab bar or toolbar.

```
(Rect) contentArea()
```

navigationBar

Returns the the app's navigation bar.

```
(UIANavigationBar) navigationBar()
```

navigationBars

Returns an array of the navigation bars contained by this object.

```
(UIAElementArray) navigationBars()
```

tabBar

Returns the app's tab bar.

```
(UIATabBar) tabBar()
```

tabBars

Returns an array of tab bars contained by this object.

```
(UIAElementArray) tabBars()
```

toolbar

Returns the the app's toolbar.

```
(UIAToolbar) toolbar()
```

toolbars

Returns an array of toolbars contained by this object.

```
(UIAElementArray) toolbars()
```

Document Revision History

This table describes the changes to *UI Automation JavaScript Reference*.

Date	Notes
2012-09-19	Added links to two new classes. Formerly titled UI Automation Reference Collection.
2012-02-16	Fixed a broken link.
2011-10-12	Added new UIAHost class, described new features.
2010-11-15	Updated to include new UIAPopover Class Reference.
2010-09-01	Updated to include minor corrections.
2010-05-27	New document that describes the JavaScript classes used to support automated iPhone application user interface testing.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010