# Safari Extensions Reference

**User Experience**

**2010-08-03**

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



Simultaneously published in the United States and Canada.

# Contents

# Introduction

| | |
|---|---|
| **Technology area:** | Safari Extensions |
| **Companion guides** | Safari Extensions Development Guide<br>Safari Extensions Conversion Guide |

Safari extensions provide a way for you to add features to the Safari browser and extend the browsing experience. Extensions can add custom buttons to the Safari toolbar, create additional bars, add contextual menu items, display content, and inject scripts and style sheets into webpages.

Introduction

# Classes

PART I

Classes

# SafariApplication Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEventTarget` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

The `SafariApplication` class allows a Safari extension to interact with the Safari application.

## Tasks

### Creating Browser Windows

`openBrowserWindow` (page 12)
Opens a new browser window in front of all other windows.

### Accessing Browser Windows

`activeBrowserWindow` (page 11)
The active browser window.

`browserWindows` (page 12)
The current browser windows, in order from front to back.

## Properties

### activeBrowserWindow

The active browser window.

```
readonly attribute SafariBrowserWindow activeBrowserWindow
```

**Discussion**
If there are no open browser windows, the value of this attribute is `null`.

**Availability**
Available in Safari 5.0 and later.

### browserWindows

The current browser windows, in order from front to back.

```
readonly attribute array browserWindows
```

**Discussion**
If there are no browser windows, the array is empty.

Using `activeBrowserWindow` (page 11) is more efficient if you only want the frontmost window.

**Availability**
Available in Safari 5.0 and later.

## Methods

### openBrowserWindow

Opens a new browser window in front of all other windows.

```
SafariBrowserWindow openBrowserWindow (void);
```

**Return Value**
The newly opened window.

**Availability**
Available in Safari 5.0 and later.

# SafariBrowserTab Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

Instances of the `SafariBrowserTab` class represent tabs in the user's browser window.

## Tasks

### Getting Information About Browser Tabs

`browserWindow` (page 14)
The browser window containing this tab.

`page` (page 14)
A proxy object for the the web content displayed in the tab.

`title` (page 14)
The tab's current title.

`url` (page 14)
The URL loaded in this tab.

`visibleContentsAsDataURL` (page 15)
Returns a data URL for an image of the visible contents of the tab.

### Working with Browser Tabs

`activate` (page 15)
Selects the tab.

`close` (page 15)
Requests that the tab should close.

# Properties

## browserWindow

The browser window containing this tab.

```
readonly attribute SafariBrowserWindow browserWindow
```

**Availability**
Available in Safari 5.0 and later.

## page

A proxy object for the the web content displayed in the tab.

```
readonly attribute SafariWebPageProxy page
```

**Availability**
Available in Safari 5.0 and later.

## title

The tab's current title.

```
readonly attribute DOMString title
```

**Discussion**
The tab's title is the same as the title of the webpage in most cases. For example, the title of the webpage may be truncated for display, but the value of this property is not truncated.

**Availability**
Available in Safari 5.0 and later.

## url

The URL loaded in this tab.

```
attribute DOMString url
```

**Discussion**
Setting this attribute to a new value loads the page at the new URL in the tab.

**Availability**
Available in Safari 5.0 and later.

# Methods

## activate

Selects the tab.

```
void activate (void);
```

**Discussion**
Depending on the content that is currently loaded in the tab, this method may change the keyboard focus.

**Availability**
Available in Safari 5.0 and later.

## close

Requests that the tab should close.

```
void close (void);
```

**Discussion**
This method behaves like clicking the tab's close button—it does not necessarily cause the tab to close. After a tab closes, the value of all of its properties is `undefined` and all of its prototype's methods return `undefined`.

**Availability**
Available in Safari 5.0 and later.

## visibleContentsAsDataURL

Returns a data URL for an image of the visible contents of the tab.

```
DOMString visibleContentsAsDataURL (void);
```

**Return Value**
A data URL for an image of the visible contents of the tab.

**Discussion**
The image is returned as a base-64 encoded PNG.

**Availability**
Available in Safari 5.0 and later.

# SafariBrowserWindow Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

Instances of the `SafariBrowserWindow` class represent browser windows. Each window contains one or more tabs, which display web content.

## Tasks

### Getting Information About Windows

`tabs` (page 18)
The tabs in the window.

`visible` (page 18)
A Boolean value that indicates whether the window is visible.

### Interacting With Windows

`activate` (page 18)
Brings the window to the front and gives it keyboard focus.

`activeTab` (page 18)
The tab currently being displayed in the window.

`close` (page 19)
Requests that the window should close.

`openTab` (page 19)
Opens a new tab in the window.

`insertTab` (page 19)
Inserts a tab into the window.

# Properties

## activeTab

The tab currently being displayed in the window.

```
readonly attribute SafariBrowserTab activeTab
```

**Availability**
Available in Safari 5.0 and later.

## tabs

The tabs in the window.

```
readonly attribute array tabs
```

**Discussion**
The tabs are ordered in the array from left to right.

**Availability**
Available in Safari 5.0 and later.

## visible

A Boolean value that indicates whether the window is visible.

```
readonly attribute boolean visible
```

**Discussion**
This attribute is `true` if the window is being displayed, even if it is covered by other windows. It is `false` if the window has been minimized.

**Availability**
Available in Safari 5.0 and later.

# Methods

## activate

Brings the window to the front and gives it keyboard focus.

```
void activate (void);
```

**Availability**
Available in Safari 5.0 and later.

## close

Requests that the window should close.

```
void close (void);
```

**Discussion**
This method behaves like clicking the window's close button—it does not necessarily cause the window to close. After a window closes, the value of all of its properties is `undefined` and all of its prototype's methods return `undefined`.

**Availability**
Available in Safari 5.0 and later.

## insertTab

Inserts a tab into the window.

```
void insertTab (in SafariBrowserTab  tab, in long  index);
```

**Parameters**

*tab*
: The tab being inserted.

*index*
: The location where the tab is being inserted.

**Availability**
Available in Safari 5.0 and later.

## openTab

Opens a new tab in the window.

```
SafariBrowserTab openTab (in DOMString  visibility, in long  index);
```

**Parameters**

*visibility*
: Either `foreground` if the tab should be opened in the foreground, or `background` if it should be opened in the background. Optional.

*index*
: The desired location of the new tab. Optional.

**Return Value**
A new tab.

**Discussion**
The default visibility is `foreground`, used when `visibility` is omitted or undefined. An invalid value causes an exception.

Tab indexes start at 0 on the far left and increases going to the right. The default location is at the far right. If `index` is negative, the tab is inserted at the far left; if `index` is greater than the number of tabs currently open, the tab is inserted at the far right.

**Availability**
Available in Safari 5.0 and later.

# SafariCommandEvent Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEvent` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariCommandEvent` class represent events being used to send a command.

The event type for this class is `command`.

## Tasks

### Getting the Command Identifier

`command` (page 21)
The command identifier of the target being dispatched.

## Properties

### command

The command identifier of the target being dispatched.

```
readonly attribute DOMString command
```

**Availability**
Available in Safari 5.0 and later.

# SafariContentBrowserTabProxy Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

Instances of the `SafariContentBrowserTabProxy` class serve as proxy objects, allowing scripts injected in the web content to communicate with `SafariBrowserTab` objects and the rest of the extension.

The counterpart class, `SafariWebPageProxy`, is used to pass messages from outside the web content to the scripts injected into it.

## Tasks

### Sending Messages

`canLoad` (page 23)
Dispatches a `canLoad` message to the browser window tab.

`dispatchMessage` (page 24)
Dispatches a message to the browser window tab.

`setContextMenuEventUserInfo` (page 24)
Sets the user info to the appropriate context information.

## Methods

### canLoad

Dispatches a `canLoad` message to the browser window tab.

```
any canLoad (in BeforeLoadEvent  event, in any  message);
```

**Parameters**

*event*
The before-load event you are responding to.

*message*
The body of the message. Optional.

**Discussion**
Use this method to set up an observer for before-load events and call this method from the observer. Then use the return value to respond to the before-load event, permitting or blocking the resource from being loaded.

This message is sent synchronously, unlike all other messages.

## dispatchMessage

Dispatches a message to the browser window tab.

```
void dispatchMessage (in DOMString  name, in any  message);
```

**Parameters**

*name*
The name of the message.

*message*
The body of the message. Optional.

**Discussion**
All messages except for `canLoad` messages are dispatched asynchronously. To send a `canLoad` message, you should use the `canLoad` (page 23) method.

**Availability**
Available in Safari 5.0 and later.

## setContextMenuEventUserInfo

Sets the user info to the appropriate context information.

```
void setContextMenuEventUserInfo (in MouseEvent  event, in any  userInfo);
```

**Parameters**

*event*
The event you are responding to.

*userInfo*
The context information.

**Discussion**
The context information in `userInfo` is carried along through the entire event-handling process. It enables you to provide your event handlers with contextual information, such as the user interface element with which the user interacted.

**Availability**
Available in Safari 5.0 and later.

**See Also**
`SafariExtensionContextMenuEvent`
`SafariExtensionContextMenuItemValidateEvent`

# SafariContentExtension Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

The `SafariContentExtension` class represents your extension to scripts running inside the web content; an instance of the class is accessible as `safari.extension`. Its counterpart class outside of the web content is `SafariExtension`.

## Tasks

### Getting the Base URI

`baseURI` (page 25)
The URI that corresponds to the root of the extension's bundle.

## Properties

### baseURI

The URI that corresponds to the root of the extension's bundle.

```
readonly attribute DOMString baseURI
```

**Availability**
Available in Safari 5.0 and later.

# SafariContentNamespace Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

The `SafariContentNamespace` class is a namespace that provides access to the Safari Extensions functionality inside of a web content area.

The corresponding class, `SafariNamespace`, is used outside of web content areas.

## Tasks

### Accessing Functionality

`self` (page 27)
The associated page object.

`extension` (page 27)
The current Safari extension.

## Properties

### extension

The current Safari extension.

```
readonly attribute SafariContentExtension extension
```

**Availability**
Available in Safari 5.0 and later.

### self

The associated page object.

```
readonly attribute SafariContentWebPage self
```

**Availability**
Available in Safari 5.0 and later.

# SafariContentWebPage Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEventTarget` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariContentWebPage` class represent a web content area.

## Tasks

### Working with Tabs

`tab` (page 29)
A proxy object for the tab containing the web content.

## Properties

### tab

A proxy object for the tab containing the web content.

```
readonly attribute SafariContentBrowserTabProxy tab
```

**Availability**
Available in Safari 5.0 and later.

# SafariEvent Class Reference

**Technology area:** Safari Extensions

## Overview

The `SafariEvent` class is a base class used to encapsulate events. Developers who are familiar with DOM events will notice many similarities in design and behavior.

Events are sent to their target and its ancestors in three phases. In order, they are as follows:

1. **Capturing.** The event is sent to every ancestor of the its target, starting with the most distant ancestor, and ending with the parent of the event's target.
2. **Targeting**. The event is sent to its target.
3. **Bubbling.** The event is sent to every ancestor of its target, starting with the parent of the event's target, and ending with its most distant ancestor.

You can tell which phase an event is in from its `eventPhase` property. When you register for an event notification, you specify whether you want to be notified during the capturing phase or not, so you may not always need to check what phase the event is in.

## Tasks

### Targeting Events

`type` (page 33)
The type of the event.

`target` (page 33)
The target of the event.

`currentTarget` (page 32)
The object that the event is currently being sent to.

`timeStamp` (page 33)
The time and date that the event was created.

### Interacting with Event Propagation

`eventPhase` (page 33)
: A number that indicates which event-handling phase the event is in.

`bubbles` (page 32)
: A Boolean value that indicates whether the event goes through the bubbling phase.

`cancelable` (page 32)
: A Boolean value that indicates whether the event can be canceled.

`stopPropagation` (page 34)
: Prevents the event from any further propagation.

### Preventing the Default Action

`preventDefault` (page 33)
: Prevents the browser from performing the default action for the event.

`defaultPrevented` (page 33)
: A Boolean value that indicates whether the default action has been prevented.

## Properties

### bubbles

A Boolean value that indicates whether the event goes through the bubbling phase.

```
readonly attribute boolean bubbles
```

### cancelable

A Boolean value that indicates whether the event can be canceled.

```
readonly attribute boolean cancelable
```

**See Also**
`stopPropagation` (page 34)

### currentTarget

The object that the event is currently being sent to.

```
readonly attribute SafariEventTarget currentTarget
```

**Discussion**
This attribute varies as the event progresses through the phases, changing as the event moves through the event-dispatch hierarchy.

### defaultPrevented

A Boolean value that indicates whether the default action has been prevented.

```
readonly attribute boolean defaultPrevented
```

### eventPhase

A number that indicates which event-handling phase the event is in.

```
readonly attribute unsigned short eventPhase
```

**Discussion**
The values for this property are the same as the values used by Webkit to identify the event-handling phases.

### target

The target of the event.

```
readonly attribute SafariEventTarget target
```

**Discussion**
This attribute stays the same as the event moves through the event-dispatch hierarchy. Its value is the same as the object that the event is sent to during the targeting phase.

### timeStamp

The time and date that the event was created.

```
readonly attribute DOMTimeStamp timeStamp
```

### type

The type of the event.

```
readonly attribute DOMString type
```

**Discussion**
The string used to identify a particular type of event is documented in the reference for that class.

## Methods

### preventDefault

Prevents the browser from performing the default action for the event.

```
void preventDefault()
```

**Discussion**
Use this method to indicate that your extension has already fully handled the event; you don't want the browser to do anything. Note that preventing the default action does not stop an event from propagating.

In the current implementation, there are no Safari Extensions events that have default actions associated with them.

## stopPropagation

Prevents the event from any further propagation.

```
void stopPropagation()
```

**Discussion**
Propagation can only be stopped if an event is can be canceled. After propagation is stopped, the event is not sent to any other targets.

**See Also**
`cancelable` (page 32)

# SafariEventTarget Class Reference

**Technology area:** Safari Extensions

## Overview

The `SafariEventTarget` class serves as a base class for all objects that participate in the event-dispatch hierarchy.

## Tasks

### Registering Event Listeners

`addEventListener` (page 35)
Starts listening for the specified type of event.

`removeEventListener` (page 36)
Stops listening for the specified type of event.

## Methods

### addEventListener

Starts listening for the specified type of event.

```
void addEventListener(in DOMString type, in SafariEventListener listener, in boolean
    useCapture);
```

**Parameters**

*type*
The type of event to listen for.

*listener*
The function to call when the event occurs.

*useCapture*
Pass `true` if you want to listen during the capturing phase; otherwise, `false`.

**Discussion**
For the string used to identify the type of an event, see the reference for the class used to represent that event.

## removeEventListener

Stops listening for the specified type of event.

```
void removeEventListener(in DOMString type, in SafariEventListener listener, in
    boolean useCapture);
```

### Parameters

*type*

The type of event to listen for.

*listener*

The function to call when the event occurs.

*useCapture*

Pass `true` if you want to listen during the capturing phase; otherwise, `false`.

### Discussion

For the listener to be removed, the parameters must be exactly the same as the parameters that were passed to `addEventListener` (page 35) to add the listener.

# SafariExtension Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

The `SafariExtension` class represents your extension outside of the web content; an instance of the class is accessible as `safari.extension`. Its counterpart class for scripts running within the web content is `SafariContentExtension`.

**Adding and removing style sheets and scripts.** Adding or removing a content style sheet applies to pages immediately. Adding or removing a content script applies only to pages opened or reloaded after that point. Removing a style sheet or a script that is in the `Info.plist` file only removes it from the current browser session.

**Whitelists and blacklists.** A content script or style sheet may specify a blacklist and a whitelist; both are optional. See "The Extension Builder Interface" in *Safari Extensions Development Guide* for a description of the pattern format. The script or style sheet will be applied to a page only in the following cases:

- The whitelist and blacklist are both empty.
- The whitelist is empty, and the page's URL does not match anything on the blacklist.
- The page's URL matches a pattern on the whitelist, and the blacklist is empty.
- The page's URL matches a pattern on the whitelist and it does not match anything on the blacklist.

## Tasks

### Working with Extensions

`bars` (page 38)
All of the extension's bars.

`baseURI` (page 39)
The URI that corresponds to the root of the extension's bundle.

`globalPage` (page 39)
The extension's global page, or `null` if the extension doesn't have a global page.

`toolbarItems` (page 39)
All of the extension's toolbar items.

### Adding and Removing Scripts and Style Sheets

`addContentScript` (page 40)
Adds a content script from a string.

`addContentScriptFromURL` (page 40)
Adds a content script from a URL.

`addContentStyleSheet` (page 41)
Adds a content style sheet from a string.

`addContentStyleSheetFromURL` (page 41)
Adds a content style sheet from a URL.

`removeContentScript` (page 42)
Removes the specified content script.

`removeContentScripts` (page 42)
Removes all content scripts added by this extension.

`removeContentStyleSheet` (page 42)
Removes the specified content style sheet.

`removeContentStyleSheets` (page 43)
Removes all content style sheets added by this extension.

### Storing Settings

`settings` (page 39)
A storage object for the extension's normal settings.

`secureSettings` (page 39)
A storage object for the extension's secure settings.

## Properties

### bars

All of the extension's bars.

```
readonly attribute array bars
```

**Discussion**
If there are no bars, the array is empty.

If multiple windows are open, there might be duplicates. Each bar appears separately for every window it appears in. When updating a bar, you should usually update it in every window.

**Availability**
Available in Safari 5.0 and later.

## baseURI

The URI that corresponds to the root of the extension's bundle.

```
readonly attribute DOMString baseURI
```

**Availability**
Available in Safari 5.0 and later.

## globalPage

The extension's global page, or `null` if the extension doesn't have a global page.

```
readonly attribute SafariExtensionGlobalPage globalPage
```

**Availability**
Available in Safari 5.0 and later.

## secureSettings

A storage object for the extension's secure settings.

```
readonly attribute SafariExtensionSecureSettings secureSettings
```

**Discussion**
Use `settings` (page 39) for normal settings.

**Availability**
Available in Safari 5.0 and later.

## settings

A storage object for the extension's normal settings.

```
readonly attribute SafariExtensionSettings settings
```

**Discussion**
Do not store sensitive information with this method, such as passwords or credit card numbers. Use `secureSettings` (page 39) instead.

**Availability**
Available in Safari 5.0 and later.

## toolbarItems

All of the extension's toolbar items.

```
readonly attribute array toolbarItems
```

**Discussion**
If there are no toolbar items, this is an empty array.

If multiple windows are open, there might be duplicates. Each toolbar item appears separately for every window it appears in. When updating a toolbar item, you should usually update it in every window.

**Availability**
Available in Safari 5.0 and later.

# Methods

## addContentScript

Adds a content script from a string.

```
DOMString addContentScript (in DOMString  source, in array  whitelist, in array
    blacklist, in boolean  runAtEnd);
```

**Parameters**

*source*
The source code of the script.

*whitelist*
A list of patterns matching URLs of pages that the script should be run on.

*blacklist*
A list of patterns matching URLs of pages that the script should not be run on.

*runAtEnd*
If `true`, the script waits until the page is fully loaded before running.

**Return Value**
If the script was successfully added, a generated URL that can be used to remove the script; otherwise, `null`.

**Discussion**
If `runAtEnd` is `true`, the script is run as soon as the page has completely finished loading. Otherwise, it is run as soon as the DOM is ready, before loading subresources such as images and the contents of frames, which allows you to block resources from being loaded.

**Availability**
Available in Safari 5.0 and later.

## addContentScriptFromURL

Adds a content script from a URL.

```
DOMString addContentScriptFromURL (in DOMString  url, in array  whitelist, in array
     blacklist, in boolean  runAtEnd);
```

**Parameters**

*url*
The URL of the script.

*whitelist*
A list of patterns matching URLs of pages that the script should be run on.

*blacklist*
A list of patterns matching URLs of pages that the script should not be run on.

*runAtEnd*
If `true`, the script waits until the page is fully loaded before running.

**Return Value**
If the script was successfully added, the supplied URL; otherwise, `null`.

**Discussion**
If `runAtEnd` is `true`, the script is run as soon as the page has completely finished loading. Otherwise, it is run as soon as the DOM is ready, before loading subresources such as images and the contents of frames, which allows you to block resources from being loaded.

**Availability**
Available in Safari 5.0 and later.

## addContentStyleSheet

Adds a content style sheet from a string.

```
DOMString addContentStyleSheet (in DOMString  source, in array  whitelist, in array
      blacklist);
```

**Parameters**

*source*
The source code of the style sheet.

*whitelist*
A list of patterns matching URLs of pages that the script should be applied to.

*blacklist*
A list of patterns matching URLs of pages that the script should not be applied to.

**Return Value**
If the style sheet was successfully added, a generated URL that can be used to remove it; otherwise, `null`.

**Discussion**
The style sheet behaves like a user-level style sheet. It is loaded after user-level style sheets, so it can override them, as well as overriding page-level style sheets.

Pages that are already loaded are updated to use the style sheet after it is added.

**Availability**
Available in Safari 5.0 and later.

## addContentStyleSheetFromURL

Adds a content style sheet from a URL.

```
DOMString addContentStyleSheetFromURL (in DOMString  url, in array  whitelist, in
     array  blacklist);
```

**Parameters**

*url*
The URL of the style sheet.

*whitelist*
A list of patterns matching URLs of pages that the script should be applied to.

*blacklist*
A list of patterns matching URLs of pages that the script should not be applied to.

**Return Value**
If the style sheet was successfully added, the supplied URL; otherwise, `null`.

**Discussion**
The style sheet behaves like a user-level style sheet. It is loaded after user-level style sheets, so it can override them, as well as overriding page-level style sheets.

Pages that are already loaded are updated to use the style sheet after it is added.

**Availability**
Available in Safari 5.0 and later.

## removeContentScript

Removes the specified content script.

```
void removeContentScript (in DOMString  url);
```

**Parameters**

*url*
The URL of the script being removed.

**Discussion**
Content scripts specified in the `Info.plist` are removed only for the current browser session.

**Availability**
Available in Safari 5.0 and later.

## removeContentScripts

Removes all content scripts added by this extension.

```
void removeContentScripts (void);
```

**Discussion**
Content scripts specified in the `Info.plist` are removed only for the current browser session.

**Availability**
Available in Safari 5.0 and later.

## removeContentStyleSheet

Removes the specified content style sheet.

```
void removeContentStyleSheet (in DOMString  url);
```

**Parameters**

*url*

The URL of the style sheet being removed.

**Discussion**

Content style sheets specified in the `Info.plist` are removed only for the current browser session.

**Availability**

Available in Safari 5.0 and later.

## removeContentStyleSheets

Removes all content style sheets added by this extension.

```
void removeContentStyleSheets (void);
```

**Discussion**

Content style sheets specified in the `Info.plist` are removed only for the current browser session.

**Availability**

Available in Safari 5.0 and later.

# SafariExtensionBar Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEventTarget` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionBar` class represent a bar that your extension provides. An extension can add any number of bars to Safari. Adding bars is optional.

Safari creates a separate instance of the bars from your extensions for every window. Thus, if the user opens multiple Safari windows, there are multiple `SafariExtensionBar` objects representing the same bar — one in each window. Also, if the user doesn't have any Safari windows open, there are no `SafariExtensionBar` objects.

To access the array of all bars added by your extension, use the `SafariExtension` method `bars` (page 38).

## Tasks

### Bar Visibility

`visible` (page 47)
A Boolean value that indicates whether the bar is visible.

`hide` (page 47)
Hides the bar.

`show` (page 47)
Shows the bar.

### Accessing Windows

`browserWindow` (page 46)
The browser window containing the bar.

`contentWindow` (page 46)
The DOM window object of the bar.

### Identifying Bars

`identifier` (page 46)
The unique identifier of the bar.

`label` (page 46)
The title of the bar.

## Properties

### browserWindow

The browser window containing the bar.

```
readonly attribute SafariBrowserWindow browserWindow
```

**Availability**
Available in Safari 5.0 and later.

### contentWindow

The DOM window object of the bar.

```
readonly attribute DOMWindow contentWindow
```

**Availability**
Available in Safari 5.0 and later.

### identifier

The unique identifier of the bar.

```
readonly attribute DOMString identifier
```

**Availability**
Available in Safari 5.0 and later.

### label

The title of the bar.

```
attribute DOMString label
```

**Discussion**
Setting an empty string, `null`, or `undefined` has no effect.

**Availability**
Available in Safari 5.0 and later.

### visible

A Boolean value that indicates whether the bar is visible.

```
readonly attribute boolean visible
```

**Availability**
Available in Safari 5.0 and later.

## Methods

### hide

Hides the bar.

```
void hide (in boolean  doNotRemember);
```

**Parameters**

*doNotRemember*

If `true`, new bars with the same identifier should be also be hidden in the future. Defaults to `false`.

**Discussion**
If the bar is already hidden, does nothing.

**Availability**
Available in Safari 5.0 and later.

### show

Shows the bar.

```
void show (in boolean  doNotRemember);
```

**Parameters**

*doNotRemember*

If `true`, new bars with the same identifier should also be shown. Defaults to `false`.

**Discussion**
If the bar is already being shown, does nothing.

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionContextMenu Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

Instances of the `SafariExtensionContextMenu` class represent a contextual menu that is populated by the extension. Safari displays the contents of this menu as part of its contextual menu.

Each time a contextual menu is about to be displayed, Safari creates a new menu, populated with the default menu items from your extension's `Info.plist` file. Next, a `SafariExtensionContextMenuEvent` event is sent, and its listeners add menu items to the menu. Then a `SafariExtensionContextMenuItemValidateEvent` is sent for each menu item, and its listeners have the opportunity to prevent items in the context menu from being displayed by marking them as disabled.

Your extension has access only to context menu items that it has added; there is no way to change or count menu items from Safari or other extensions. Therefore, it is important to limit the menu items that your extension adds to a reasonable number, to avoid displaying an unusably long contextual menu.

## Tasks

### Adding Menu Items

`appendContextMenuItem` (page 50)
Appends a menu item to the contextual menu.

`insertContextMenuItem` (page 50)
Inserts a menu item at a specific index in the contextual menu.

### Accessing Menu Items

`contextMenuItems` (page 50)
Returns a list of the context menu items from this extension.

# Properties

## contextMenuItems

Returns a list of the context menu items from this extension.

```
readonly attribute array contextMenuItems
```

**Discussion**
Only menu items from your extension are returned.

**Availability**
Available in Safari 5.0 and later.

# Methods

## appendContextMenuItem

Appends a menu item to the contextual menu.

```
SafariExtensionContextMenuItem appendContextMenuItem (in DOMString  identifier, in
    DOMString  title, in DOMString  command);
```

**Parameters**

*identifier*
The unique identifier of the menu item.

*title*
The title of the menu item.

*command*
The command identifier that the context menu item sends when activated. Optional.

**Return Value**
The context menu item that was appended.

**Discussion**
If another menu item with the same identifier already exists, it is removed before appending the menu item. If `command` is not supplied, `identifier` is used as the command identifier.

**Availability**
Available in Safari 5.0 and later.

## insertContextMenuItem

Inserts a menu item at a specific index in the contextual menu.

```
SafariExtensionContextMenuItem insertContextMenuItem (in unsigned int  index, in
    DOMString  identifier, in DOMString  title, in DOMString  command);
```

**Parameters**

*index*

The index where the menu item is being inserted.

*identifier*

The unique identifier of the menu item.

*title*

The title of the menu item.

*command*

The command identifier that the context menu item sends when activated. Optional.

**Return Value**

The context menu item that was inserted.

**Discussion**

If another menu item with the same identifier already exists, it is removed before appending the menu item. If `command` is not supplied, `identifier` is used as the command identifier.

**Availability**

Available in Safari 5.0 and later.

# SafariExtensionContextMenuEvent Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEvent` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionContextMenuEvent` class are used to notify listeners when a context menu is about to be displayed.

Each time a contextual menu is about to be displayed, Safari creates a new menu, populated with the default menu items from your extension's `Info.plist` file. Next, a `SafariExtensionContextMenuEvent` event is sent, and its listeners add menu items to the menu. Then a `SafariExtensionContextMenuItemValidateEvent` event is sent for each menu item, and its listeners have the opportunity to prevent items in the context menu from being displayed by marking them as disabled.

The event type for this class is `contextmenu`.

## Tasks

### Determining the Context

`userInfo` (page 54)
Information about the current context menu event.

`contextMenu` (page 53)
The context menu being built up.

## Properties

### contextMenu

The context menu being built up.

```
readonly attribute SafariExtensionContextMenu contextMenu
```

**Availability**
Available in Safari 5.0 and later.

## userInfo

Information about the current context menu event.

```
readonly attribute any userInfo
```

**Discussion**
From within the web content area, you listen for a DOM context menu event, and then call the `setContextMenuEventUserInfo` (page 24) method. The value you provide is used as the `userInfo` property by the `SafariExtensionContextMenuEvent` and `SafariExtensionContextMenuItemValidateEvent` events when they are sent.

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionContextMenuItem Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEventTarget` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionContextMenuItem` class represent individual menu items within an extension's context menu.

## Tasks

### Setting Up a Menu Item

`command` (page 55)
The command identifier that the context menu item sends when activated.

`disabled` (page 56)
A Boolean value that indicates whether a context menu item is disabled.

`identifier` (page 56)
The unique identifier of the context menu item.

`title` (page 56)
The title displayed in the context menu.

## Properties

### command

The command identifier that the context menu item sends when activated.

```
attribute DOMString command
```

**Discussion**
Setting an empty string, `null`, or `undefined` has no effect.

This attribute is optional; the value defaults to the value of `identifier`.

**Availability**
Available in Safari 5.0 and later.

## disabled

A Boolean value that indicates whether a context menu item is disabled.

```
attribute boolean disabled
```

**Discussion**
Disabled menu items are not displayed in the context menu.

**Availability**
Available in Safari 5.0 and later.

## identifier

The unique identifier of the context menu item.

```
readonly attribute DOMString identifier
```

**Discussion**
This attribute is required.

**Availability**
Available in Safari 5.0 and later.

## title

The title displayed in the context menu.

```
attribute DOMString title
```

**Discussion**
This attribute is required.

Setting an empty string, `null`, or `undefined` has no effect.

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionContextMenuItemCommandEvent Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEvent` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionContextMenuItemCommandEvent` class are used to send command events from a context menu item.

The event type for this class is `command`.

## Tasks

### Getting Context Information

`userInfo` (page 57)
The user info object for this context menu event.

## Properties

### userInfo

The user info object for this context menu event.

```
readonly attribute any userInfo
```

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionContextMenuItemValidateEvent Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEvent` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionContextMenuItemValidateEvent` class are used to notify listeners when a context menu is about to be displayed.

Each time a contextual menu is about to be displayed, Safari creates a new menu, populated with the default menu items from your extension's `Info.plist` file. Next, a `SafariExtensionContextMenuEvent` event is sent, and its listeners add menu items to the menu. Then a `SafariExtensionContextMenuItemValidateEvent` event is sent for each menu item, and its listeners have the opportunity to prevent items in the context menu from being displayed by marking them as disabled.

The event type for this class is `validate`.

## Tasks

### Determining the Context

`userInfo` (page 59)
Information about the current context menu event.

## Properties

### userInfo

Information about the current context menu event.

```
readonly attribute any userInfo
```

**Discussion**
From within the web content area, you listen for a DOM context menu event, and then call the `setContextMenuEventUserInfo` (page 24) method. The value you provide is used as the `userInfo` property by the `SafariExtensionContextMenuEvent` and `SafariExtensionContextMenuItemValidateEvent` events when they are sent.

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionGlobalPage Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

Safari extensions have a global page, represented by the `SafariExtensionGlobalPage` class. The global page is loaded once when the extension is loaded, but it is never displayed to the user. It can contain resources such as scripts, images, and HTML content.

The global page is a good place to run scripts that that should be run only once for all of Safari and to store data that is displayed in multiple locations. For example, a stock ticker extension could use a script on its global page fetch and store the stock prices. This makes it easier to keep the user interface consistant by allowing the extension to maintain only one source data; this also uses less memory than making a copy of the data for each window.

## Tasks

### Accessing the Global Page

`contentWindow` (page 61)
The DOM window object of the extension's global page.

## Properties

### contentWindow

The DOM window object of the extension's global page.

```
readonly attribute DOMWindow contentWindow
```

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionMessageEvent Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEvent` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionMessageEvent` encapsulate a message being passed. Message passing is the only way to communicate information from inside the web content area to scripts running outside of it, and vice versa.

The event type for this class is `message`.

## Tasks

### Working with Messages

`name` (page 64)
The name of the message.

`message` (page 63)
The message data.

## Properties

### message

The message data.

```
attribute any message
```

**Discussion**
This attribute is optional; if you don't need it you can set it to `undefined` or `null`, or not set any value.

The data can be any scalar value, or any object that conforms to the W3C standard for safe passing of structured cloned data.

## name

The name of the message.

```
readonly attribute DOMString name
```

**Discussion**
There is no fixed meaning to the name of a message; it is up to the programmer to decide how messages should be named within an extension.

A message's name must not be the empty string. The message name `canLoad` is used by the `SafariContentBrowserTabProxy` method `canLoad` (page 23); you should not send messages with this name.

# SafariExtensionSecureSettings Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEventTarget` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

The `SafariExtensionSecureSettings` class provides a place for your extension to securely store settings that should persist between sessions. You interact with it in exactly the same way as you interact with the `SafariExtensionSettings` class.

## Tasks

### Reading and Writing Settings

`getItem` (page 66)
Returns the current value of a key.

`setItem` (page 66)
Sets the value of a key.

### Removing Settings

`removeItem` (page 66)
Removes a key.

`clear` (page 65)
Removes all key-value pairs.

## Methods

### clear

Removes all key-value pairs.

```
void clear (void);
```

**Discussion**
If there is nothing to remove, this method does nothing.

On success, this method dispatches a `SafariExtensionSettingsChangeEvent` event describing the change.

**Availability**
Available in Safari 5.0 and later.

## getItem

Returns the current value of a key.

```
any getItem (in DOMString  key);
```

**Parameters**

*key*

The key whose value is being returned.

**Return Value**
If the key exists, its current value; otherwise, `null`.

**Availability**
Available in Safari 5.0 and later.

## removeItem

Removes a key.

```
void removeItem (in DOMString  key);
```

**Parameters**

*key*

The key being removed.

**Discussion**
If there is no value with the given key, this method does nothing.

On success, this method dispatches a `SafariExtensionSettingsChangeEvent` event describing the change.

**Availability**
Available in Safari 5.0 and later.

## setItem

Sets the value of a key.

```
voidsetItem (in DOMString key, in any value);
```

**Parameters**

*key*

The key whose value is being set.

*value*

The value being set.

**Discussion**
The behavior of this method is undefined if `value` cannot be serialized to a JSON value.

On success, this method dispatches a `SafariExtensionSettingsChangeEvent` event describing the change.

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionSettings Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEventTarget` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

The `SafariExtensionsSettings` class provides a place for your extension to store settings that should persist between sessions.

**Important:** Use `SafariExtensionSecureSettings` to store sensitive settings, such as passwords or credit card numbers.

You can access settings using the provided methods or using JavaScript's built-in syntax, as follows:

```
// Get a value.
safari.extension.settings.someProperty
safari.extension.settings["someProperty"]
safari.extension.settings.getItem("someProperty")

// Set a value.
safari.extension.settings.someProperty = 42;
safari.extension.settings["someProperty"] = 42;
safari.extension.settings.setItem("someProperty", 42");

// Delete a value.
delete safari.extension.someProperty;
delete safari.extension["someProperty"];
safari.extension.removeItem("someProperty");
```

Note that there is a minor difference for properties that don't exist: JavaScript's build-in syntax returns `undefined`, but the `getItem` method returns `null`.

## Tasks

### Reading and Writing Settings

`getItem` (page 70)
Returns the current value of a key.

`setItem` (page 71)
Sets the value of a key.

### Removing Settings

`removeItem` (page 70)
Removes a key.

`clear` (page 70)
Removes all key-value pairs.

# Methods

## clear

Removes all key-value pairs.

```
void clear (void);
```

**Discussion**
If there is nothing to remove, this method does nothing.

On success, dispatches a `SafariExtensionSettingsChangeEvent` event describing the change.

**Availability**
Available in Safari 5.0 and later.

## getItem

Returns the current value of a key.

```
any getItem (in DOMString  key);
```

**Parameters**

*key*
The key whose value is being returned.

**Return Value**
If the key exists, its current value; otherwise, `null`.

**Availability**
Available in Safari 5.0 and later.

## removeItem

Removes a key.

```
void removeItem (in DOMString  key);
```

**Parameters**

*key*

The key being removed.

**Discussion**

If there is no value with the given key, this method does nothing.

On success, this method dispatches a `SafariExtensionSettingsChangeEvent` event describing the change.

**Availability**

Available in Safari 5.0 and later.

## setItem

Sets the value of a key.

```
void setItem (in DOMString  key, in any  value);
```

**Parameters**

*key*

The key whose value is being set.

*value*

The value being set.

**Discussion**

The behavior of this method is undefined if `value` cannot be serialized to a JSON value.

On success, this method dispatches a `SafariExtensionSettingsChangeEvent` event describing the change.

**Availability**

Available in Safari 5.0 and later.

# SafariExtensionSettingsChangeEvent Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEvent` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionsSettingsChangeEvent` class are used to provide a description of a change made to the extension's settings. A settings change event is sent when your extension changes a setting as well as when a setting is changed outside your extension (for example, a setting changed in the Safari preferences).

These events do not bubble; their target is the settings or secure settings object that changed.

The event type for this class is `change`.

**Note:** Take special care to leave the settings unchanged while handling to a settings-change event. Otherwise you can create an infinite loop, in which each settings-change event triggers another settings-change event.

## Tasks

### Getting Information About Changes to Settings

`key` (page 74)
The key identifier of the setting that was changed.

`newValue` (page 74)
The value after the settings change.

`oldValue` (page 74)
The value before the settings change.

# Properties

## key

The key identifier of the setting that was changed.

```
readonly attribute DOMString key
```

**Discussion**
If all of the settings have been removed, this value is `null`.

**Availability**
Available in Safari 5.0 and later.

## newValue

The value after the settings change.

```
readonly attribute any newValue
```

**Discussion**
If the key was removed or all settings have been removed, this value is `null`.

**Availability**
Available in Safari 5.0 and later.

## oldValue

The value before the settings change.

```
readonly attribute any oldValue
```

**Discussion**
If the key was added or all settings have been removed, this value is `null`.

**Availability**
Available in Safari 5.0 and later.

# SafariExtensionToolbarItem Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEventTarget` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariExtensionToolbarItem` represent items that your extension adds to the Safari toolbar. Users can select which toolbar items are shown by selecting Customize Toolbar in the same way they can add and remove toolbar items that are provided by Safari.

## Tasks

### Setting Up the Appearance

`badge` (page 76)
The current badge number.

`image` (page 77)
The current image URL.

`label` (page 77)
The label of the toolbar item, as shown in the toolbar's overflow menu.

`paletteLabel` (page 77)
The label of the toolbar item, as shown in the Customize palette.

`toolTip` (page 78)
The tooltip (help tag) of the toolbar item.

### Handling Click Events

`browserWindow` (page 76)
The containing browser window.

`command` (page 76)
The command identifier that the toolbar item sends when it is clicked.

`disabled` (page 76)
A Boolean value that indicates whether the toolbar item is disabled.

`identifier` (page 77)
The unique identifier of the toolbar item.

`validate` (page 78)
Dispatches a validate event for this toolbar item.

# Properties

## badge

The current badge number.

```
attribute long badge
```

**Discussion**
The default value is 0, which hides the badge. If you set a value that is too long, the beginning and end of the value will be shown with an ellipsis in the middle.

Setting a value of `NaN` or `Infinity` is treated as 0. Non-integer values are rounded to the nearest integer.

**Availability**
Available in Safari 5.0 and later.

## browserWindow

The containing browser window.

```
readonly attribute SafariBrowserWindow browserWindow
```

**Availability**
Available in Safari 5.0 and later.

## command

The command identifier that the toolbar item sends when it is clicked.

```
attribute DOMString command
```

**Discussion**
This attribute is optional; its value defaults to the value of `identifier`.

Setting a value of `null`, `undefined`, or the empty string has no effect.

**Availability**
Available in Safari 5.0 and later.

## disabled

A Boolean value that indicates whether the toolbar item is disabled.

```
attribute boolean disabled
```

**Discussion**
The default value is `false`. Nothing happens when the user tries to interact with a toolbar item that is disabled.

**Availability**
Available in Safari 5.0 and later.

## identifier

The unique identifier of the toolbar item.

```
readonly attribute DOMString identifier
```

**Discussion**
This attribute is required.

**Availability**
Available in Safari 5.0 and later.

## image

The current image URL.

```
attribute DOMString image
```

**Discussion**
This attribute can be changed to the URL of an image in the extension bundle. Setting a value of `null`, `undefined`, or the empty string has no effect.

**Availability**
Available in Safari 5.0 and later.

## label

The label of the toolbar item, as shown in the toolbar's overflow menu.

```
attribute DOMString label
```

**Discussion**
This attribute is required. Setting a value of `null`, `undefined`, or the empty string has no effect.

**Availability**
Available in Safari 5.0 and later.

## paletteLabel

The label of the toolbar item, as shown in the Customize palette.

```
readonly attribute DOMString paletteLabel
```

**Discussion**
This attribute is optional; its value defaults to the value of `label`.

**Availability**
Available in Safari 5.0 and later.

### toolTip

The tooltip (help tag) of the toolbar item.

```
attribute DOMString toolTip
```

**Discussion**
This attribute is optional; its value defaults to the value of `label`.

Setting a value of `null`, `undefined`, or the empty string has no effect.

**Availability**
Available in Safari 5.0 and later.

## Methods

### validate

Dispatches a validate event for this toolbar item.

```
void validate (void);
```

**Discussion**
You should call this method after a state change occurs that is relevant to your validate-event listeners. Safari also automatically sends validate events for many common browser actions, such as switching tabs and navigating to a new page.

**Availability**
Available in Safari 5.0 and later.

# SafariNamespace Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

The `SafariNamespace` class is a namespace that provides access to the Safari Extensions functionality outside of the web content area.

The corresponding class, `SafariContentNamespace`, is used from inside a web content area.

## Tasks

### Accessing Functionality

`application` (page 79)
The Safari application.

`extension` (page 80)
The current Safari extension.

`self` (page 80)
The `SafariExtensionGlobalPage` or `SafariExtensionBar` object that owns the DOM window that this instance of the `SafariNamespace` class was accessed from.

## Properties

### application

The Safari application.

```
readonly attribute SafariApplication application
```

**Availability**
Available in Safari 5.0 and later.

## extension

The current Safari extension.

```
readonly attribute SafariExtension extension
```

**Availability**
Available in Safari 5.0 and later.

## self

The `SafariExtensionGlobalPage` or `SafariExtensionBar` object that owns the DOM window that this instance of the `SafariNamespace` class was accessed from.

```
readonly attribute any self
```

**Discussion**
For example, the following is always true:

```
window.safari.self.contentWindow === window;
```

**Availability**
Available in Safari 5.0 and later.

# SafariValidateEvent Class Reference

| | |
|---|---|
| **Inherits from** | `SafariEvent` |
| **Technology area:** | Safari Extensions |
| **Availability** | Available in Safari 5.0 and later. |

## Overview

Instances of the `SafariValidateEvent` class encapsulate a validate event. User interface elements such as contextual menu items and toolbar buttons send a validate event when they are about to display. You can prevent the element from displaying by listening for validate events and marking the element as disabled.

The event type for this class is `validate`.

## Tasks

### Working with Validation Events

`command` (page 81)
The command identifier of the target being validated.

## Properties

### command

The command identifier of the target being validated.

```
readonly attribute DOMString command
```

**Availability**
Available in Safari 5.0 and later.

# SafariWebPageProxy Class Reference

**Technology area:** Safari Extensions

**Availability** Available in Safari 5.0 and later.

## Overview

Instances of the `SafariWebPageProxy` class serve as a proxy objects, allowing objects outside of the web content to communicate with the `SafariContentWebPage` object.

The counterpart class, `SafariContentBrowserTabProxy`, is used to pass messages from the web content to objects outside of the web content.

## Tasks

### Dispatching Messages

`dispatchMessage` (page 83)
Dispatches a message to the web content area.

## Methods

### dispatchMessage

Dispatches a message to the web content area.

```
void dispatchMessage (in DOMString  name, in any  message);
```

**Parameters**

*name*
The name of the message.

*message*
The body of the message. Optional.

**Availability**
Available in Safari 5.0 and later.