UIWebView Class Reference

Developer

Contents

UIWebView Class Reference

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding UIScrollViewDelegate NSCoding (UIView) UIAppearance (UIView) UIAppearanceContainer (UIView) UIDynamicItem (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iOS 2.0 and later.
Declared in	UIWebView.h
Companion guides	Text Programming Guide for iOS UIKit User Interface Catalog
Related sample code	iPhoneACFileConvertTest iPhoneMultichannelMixerTest Managed App Configuration UICatalog UIKit Printing with UIPrintInteractionController and UIViewPrintFormatter

Overview

You use the `UIWebView` class to embed web content in your application. To do so, you simply create a `UIWebView` object, attach it to a window, and send it a request to load web content. You can also use this class to move back and forward in the history of webpages, and you can even set some web content properties programmatically.

Use the `loadRequest:` (page 18) method to begin loading web content, the `stopLoading` (page 19) method to stop loading, and the `loading` (page 11) property to find out if a web view is in the process of loading.

If you allow the user to move back and forward through the webpage history, then you can use the `goBack` (page 16) and `goForward` (page 16) methods as actions for buttons. Use the `canGoBack` (page 8) and `canGoForward` (page 9) properties to disable the buttons when the user can't move in a direction.

By default, a web view automatically converts telephone numbers that appear in web content to Phone links. When a Phone link is tapped, the Phone application launches and dials the number. Set the `detectsPhoneNumbers` (page 23) property to `NO` to turn off this default behavior.

You can also use the `scalesPageToFit` (page 14) property to programmatically set the scale of web content the first time it is displayed in a web view. Thereafter, the user can change the scale using gestures.

Set the `delegate` (page 10) property to an object conforming to the `UIWebViewDelegate` protocol if you want to track the loading of web content.

> **Important:** You should not embed `UIWebView` or `UITableView` objects in `UIScrollView` objects. If you do so, unexpected behavior can result because touch events for the two objects can be mixed up and wrongly handled.

You can easily debug the HTML, CSS, and JavaScript contained inside a `UIWebView` with Web Inspector. Read "Debugging Web Content on iOS" to learn how to configure Web Inspector for iOS. Read the rest of *Safari Web Content Guide* to learn how to create web content that is optimized for Safari on iPhone and iPad.

For information about basic view behaviors, see *View Programming Guide for iOS*.

Supported File Formats

In addition to HTML content, `UIWebView` objects can be used to display other content types. For more information, see *Using UIWebView to display select document types*.

State Preservation

In iOS 6 and later, if you assign a value to this view's `restorationIdentifier` property, it attempts to preserve its URL history, the scaling and scrolling positions for each page, and information about which page is currently being viewed. During restoration, the view restores these values so that the web content appears just as it did before. For more information about how state preservation and restoration works, see *iOS App Programming Guide*.

For more information about appearance and behavior configuration, see "Web Views".

Subclassing Notes

The `UIWebView` class should not be subclassed.

Tasks

Setting the Delegate

`delegate` (page 10) *property*
The receiver's delegate.

Loading Content

– `loadData:MIMEType:textEncodingName:baseURL:` (page 16)
Sets the main page contents, MIME type, content encoding, and base URL.

– `loadHTMLString:baseURL:` (page 17)
Sets the main page content and base URL.

– `loadRequest:` (page 18)
Connects to a given URL by initiating an asynchronous client request.

`request` (page 14) *property*
The URL request identifying the location of the content to load. (read-only)

`loading` (page 11) *property*
A Boolean value indicating whether the receiver is done loading content. (read-only)

– `stopLoading` (page 19)
Stops the loading of any web content managed by the receiver.

– `reload` (page 18)
Reloads the current page.

Moving Back and Forward

`canGoBack` (page 8) *property*
A Boolean value indicating whether the receiver can move backward. (read-only)

`canGoForward` (page 9) *property*
A Boolean value indicating whether the receiver can move forward. (read-only)

– `goBack` (page 16)
Loads the previous location in the back-forward list.

– `goForward` (page 16)
Loads the next location in the back-forward list.

Setting Web Content Properties

`scalesPageToFit` (page 14) *property*

A Boolean value determining whether the webpage scales to fit the view and the user can change the scale.

`scrollView` (page 15) *property*

The scroll view associated with the web view. (read-only)

`suppressesIncrementalRendering` (page 15) *property*

A Boolean value indicating whether the web view suppresses content rendering until it is fully loaded into memory.

`keyboardDisplayRequiresUserAction` (page 11) *property*

A Boolean value indicating whether web content can programmatically display the keyboard.

`detectsPhoneNumbers` (page 23) *property* Deprecated in iOS 3.0

A Boolean value indicating whether telephone number detection is on. (Deprecated. Use `dataDetectorTypes` (page 9) instead.)

Running JavaScript

`- stringByEvaluatingJavaScriptFromString:` (page 19)

Returns the result of running a script.

Detecting Types of Data

`dataDetectorTypes` (page 9) *property*

The types of data converted to clickable URLs in the web view's content.

Managing Media Playback

`allowsInlineMediaPlayback` (page 8) *property*

A Boolean value that determines whether HTML5 videos play inline or use the native full-screen controller.

`mediaPlaybackRequiresUserAction` (page 12) *property*

A Boolean value that determines whether HTML5 videos can play automatically or require the user to start playing them.

`mediaPlaybackAllowsAirPlay` (page 11) *property*

A Boolean value that determines whether Air Play is allowed from this view.

Managing Pages

`gapBetweenPages` (page 10) *property*

The size of the gap, in points, between pages.

`pageCount` (page 12) *property*

The number of pages produced by the layout of the web view. (read-only)

`pageLength` (page 12) *property*

The size of each page, in points, in the direction that the pages flow.

`paginationBreakingMode` (page 13) *property*

The manner in which column- or page-breaking occurs.

`paginationMode` (page 13) *property*

The layout of content in the web view.

Properties

allowsInlineMediaPlayback

A Boolean value that determines whether HTML5 videos play inline or use the native full-screen controller.

```
@property(nonatomic) BOOL allowsInlineMediaPlayback
```

Discussion

The default value on iPhone is `NO`.

In order for video to play inline, not only does this property need to be set on the view, but the `video` element in the HTML document must also include the `webkit-playsinline` attribute.

Availability

Available in iOS 4.0 and later.

Declared in

`UIWebView.h`

canGoBack

A Boolean value indicating whether the receiver can move backward. (read-only)

```
@property(nonatomic, readonly, getter=canGoBack) BOOL canGoBack
```

Discussion
If `YES`, able to move backward; otherwise, `NO`.

Availability
Available in iOS 2.0 and later.

See Also
`@property canGoForward` (page 9)

Declared in
`UIWebView.h`

canGoForward

A Boolean value indicating whether the receiver can move forward. (read-only)

```
@property(nonatomic, readonly, getter=canGoForward) BOOL canGoForward
```

Discussion
If `YES`, able to move forward; otherwise, `NO` .

Availability
Available in iOS 2.0 and later.

See Also
`@property canGoBack` (page 8)

Declared in
`UIWebView.h`

dataDetectorTypes

The types of data converted to clickable URLs in the web view's content.

```
@property(nonatomic) UIDataDetectorTypes dataDetectorTypes
```

Discussion
You can use this property to specify the types of data (phone numbers, http links, email address, and so on) that should be automatically converted to clickable URLs in the web view. When clicked, the web view opens the application responsible for handling the URL type and passes it the URL.

Availability
Available in iOS 3.0 and later.

Declared in
`UIWebView.h`

delegate

The receiver's delegate.

```
@property(nonatomic, assign) id<UIWebViewDelegate> delegate
```

Discussion
The delegate is sent messages when content is loading. See *UIWebViewDelegate Protocol Reference* for the optional methods this delegate may implement.

> **Important:** Before releasing an instance of `UIWebView` for which you have set a delegate, you must first set its delegate property to `nil`. This can be done, for example, in your dealloc method.

Availability
Available in iOS 2.0 and later.

Declared in
`UIWebView.h`

gapBetweenPages

The size of the gap, in points, between pages.

```
@property(nonatomic) CGFloat gapBetweenPages
```

Discussion
The default value is `0`.

Availability
Available in iOS 7.0 and later.

Declared in
`UIWebView.h`

keyboardDisplayRequiresUserAction

A Boolean value indicating whether web content can programmatically display the keyboard.

```
@property(nonatomic) BOOL keyboardDisplayRequiresUserAction
```

Discussion

When this property is set to `YES`, the user must explicitly tap the elements in the web view to display the keyboard (or other relevant input view) for that element. When set to `NO`, a focus event on an element causes the input view to be displayed and associated with that element automatically.

The default value for this property is `YES`.

Availability

Available in iOS 6.0 and later.

Declared in

`UIWebView.h`

loading

A Boolean value indicating whether the receiver is done loading content. (read-only)

```
@property(nonatomic, readonly, getter=isLoading) BOOL loading
```

Discussion

If `YES`, the receiver is still loading content; otherwise, `NO`.

Availability

Available in iOS 2.0 and later.

See Also

`@property request` (page 14)
– `stopLoading` (page 19)
– `loadRequest:` (page 18)
– `reload` (page 18)

Declared in

`UIWebView.h`

mediaPlaybackAllowsAirPlay

A Boolean value that determines whether Air Play is allowed from this view.

```
@property(nonatomic) BOOL mediaPlaybackAllowsAirPlay
```

Discussion
The default value on both iPad and iPhone is `YES`.

Availability
Available in iOS 5.0 and later.

Declared in
`UIWebView.h`

mediaPlaybackRequiresUserAction

A Boolean value that determines whether HTML5 videos can play automatically or require the user to start playing them.

```
@property(nonatomic) BOOL mediaPlaybackRequiresUserAction
```

Discussion
The default value on both iPad and iPhone is `YES`.

Availability
Available in iOS 4.0 and later.

Declared in
`UIWebView.h`

pageCount

The number of pages produced by the layout of the web view. (read-only)

```
@property(nonatomic, readonly) NSUInteger pageCount
```

Availability
Available in iOS 7.0 and later.

Declared in
`UIWebView.h`

pageLength

The size of each page, in points, in the direction that the pages flow.

```
@property(nonatomic) CGFloat pageLength
```

Discussion
When `paginationMode` (page 13) is right to left or left to right, this property represents the width of each page. When `paginationMode` (page 13) is top to bottom or bottom to top, this property represents the height of each page.

The default value is `0`, which means the layout uses the size of the viewport to determine the dimensions of the page. Adjusting the value of this property causes a relayout.

Availability
Available in iOS 7.0 and later.

Declared in
`UIWebView.h`

paginationBreakingMode

The manner in which column- or page-breaking occurs.

```
@property(nonatomic) UIWebPaginationBreakingMode paginationBreakingMode
```

Discussion
This property determines whether certain CSS properties regarding column- and page-breaking are honored or ignored. When this property is set to `UIWebPaginationBreakingModeColumn` (page 21), the content respects the CSS properties related to column-breaking in place of page-breaking.

See "UIWebPaginationBreakingMode" (page 21) for possible values. The default value is `UIWebPaginationBreakingModePage` (page 21).

Availability
Available in iOS 7.0 and later.

Declared in
`UIWebView.h`

paginationMode

The layout of content in the web view.

```
@property(nonatomic) UIWebPaginationMode paginationMode
```

Discussion
This property determines whether content in the web view is broken up into pages that fill the view one screen at a time, or shown as one long scrolling view. If set to a paginated form, this property toggles a paginated layout on the content, causing the web view to use the values of `pageLength` (page 12) and `gapBetweenPages` (page 10) to relayout its content.

See "UIWebPaginationMode" (page 21) for possible values. The default value is `UIWebPaginationModeUnpaginated` (page 22).

Availability
Available in iOS 7.0 and later.

Declared in
`UIWebView.h`

request

The URL request identifying the location of the content to load. (read-only)

```
@property(nonatomic, readonly, retain) NSURLRequest *request
```

Availability
Available in iOS 2.0 and later.

See Also
– `loadRequest:` (page 18)
– `stopLoading` (page 19)
`@property loading` (page 11)
– `reload` (page 18)

Declared in
`UIWebView.h`

scalesPageToFit

A Boolean value determining whether the webpage scales to fit the view and the user can change the scale.

```
@property(nonatomic) BOOL scalesPageToFit
```

Discussion
If `YES`, the webpage is scaled to fit and the user can zoom in and zoom out. If `NO`, user zooming is disabled. The default value is `NO`.

Availability
Available in iOS 2.0 and later.

Declared in
`UIWebView.h`

scrollView

The scroll view associated with the web view. (read-only)

```
@property(nonatomic, readonly, retain) UIScrollView *scrollView
```

Discussion
Your application can access the scroll view if it wants to customize the scrolling behavior of the web view.

Availability
Available in iOS 5.0 and later.

Declared in
`UIWebView.h`

suppressesIncrementalRendering

A Boolean value indicating whether the web view suppresses content rendering until it is fully loaded into memory.

```
@property(nonatomic) BOOL suppressesIncrementalRendering
```

Discussion
When set to `YES`, the web view does not attempt to render incoming content as it arrives. Instead, the view's current contents remain in place until all of the new content has been received, at which point the new content is rendered. This property does not affect the rendering of content retrieved after a frame finishes loading.

The value of this property is `NO` by default.

Availability
Available in iOS 6.0 and later.

Declared in
`UIWebView.h`

Instance Methods

goBack

Loads the previous location in the back-forward list.

```
– (void)goBack
```

Availability
Available in iOS 2.0 and later.

See Also
– `goForward` (page 16)

Declared in
`UIWebView.h`

goForward

Loads the next location in the back-forward list.

```
– (void)goForward
```

Availability
Available in iOS 2.0 and later.

See Also
– `goBack` (page 16)

Declared in
`UIWebView.h`

loadData:MIMEType:textEncodingName:baseURL:

Sets the main page contents, MIME type, content encoding, and base URL.

```
– (void)loadData:(NSData *)data MIMEType:(NSString *)MIMEType
textEncodingName:(NSString *)encodingName baseURL:(NSURL *)baseURL
```

Parameters

`data`

The content for the main page.

`MIMEType`

The MIME type of the content.

`encodingName`

The IANA encoding name as in `utf-8` or `utf-16`.

`baseURL`

The base URL for the content.

Availability

Available in iOS 2.0 and later.

See Also

– `loadHTMLString:baseURL:` (page 17)

Declared in

`UIWebView.h`

loadHTMLString:baseURL:

Sets the main page content and base URL.

```
- (void)loadHTMLString:(NSString *)string baseURL:(NSURL *)baseURL
```

Parameters

`string`

The content for the main page.

`baseURL`

The base URL for the content.

Availability

Available in iOS 2.0 and later.

See Also

– `loadData:MIMEType:textEncodingName:baseURL:` (page 16)

Related Sample Code

CustomHTTPProtocol

Declared in

`UIWebView.h`

loadRequest:

Connects to a given URL by initiating an asynchronous client request.

```
- (void)loadRequest:(NSURLRequest *)request
```

Parameters

`request`

A URL request identifying the location of the content to load.

Discussion

To stop this load, use the `stopLoading` (page 19) method. To see whether the receiver is done loading the content, use the `loading` (page 11) property.

Availability

Available in iOS 2.0 and later.

See Also

`@property request` (page 14)
`- stopLoading` (page 19)
`@property loading` (page 11)
`- reload` (page 18)

Declared in

`UIWebView.h`

reload

Reloads the current page.

```
- (void)reload
```

Availability

Available in iOS 2.0 and later.

See Also

`@property request` (page 14)
`@property loading` (page 11)
`- loadRequest:` (page 18)
`- stopLoading` (page 19)

Declared in

`UIWebView.h`

stopLoading

Stops the loading of any web content managed by the receiver.

```
- (void)stopLoading
```

Discussion

Stops any content in the process of being loaded by the main frame or any of its children frames. Does nothing if no content is being loaded.

Availability

Available in iOS 2.0 and later.

See Also

`@property request` (page 14)
`@property loading` (page 11)
`- loadRequest:` (page 18)
`- reload` (page 18)

Declared in

`UIWebView.h`

stringByEvaluatingJavaScriptFromString:

Returns the result of running a script.

```
- (NSString *)stringByEvaluatingJavaScriptFromString:(NSString *)script
```

Parameters

`script`

The script to run.

Return Value

The result of running `script` or `nil` if it fails.

Discussion

JavaScript execution time is limited to 10 seconds for each top-level entry point. If your script executes for more than 10 seconds, the web view stops executing the script. This is likely to occur at a random place in your code, so unintended consequences may result. This limit is imposed because JavaScript execution may cause the main thread to block, so when scripts are running, the user is not able to interact with the webpage.

JavaScript allocations are also limited to 10 MB. The web view raises an exception if you exceed this limit on the total memory allocation for JavaScript.

Availability
Available in iOS 2.0 and later.

Declared in
`UIWebView.h`

Constants

UIWebViewNavigationType

Constant indicating the user's action.

```
enum {
   UIWebViewNavigationTypeLinkClicked,
   UIWebViewNavigationTypeFormSubmitted,
   UIWebViewNavigationTypeBackForward,
   UIWebViewNavigationTypeReload,
   UIWebViewNavigationTypeFormResubmitted,
   UIWebViewNavigationTypeOther
};
typedef NSUInteger UIWebViewNavigationType;
```

Constants

`UIWebViewNavigationTypeLinkClicked`
User tapped a link.
Available in iOS 2.0 and later.
Declared in `UIWebView.h`.

`UIWebViewNavigationTypeFormSubmitted`
User submitted a form.
Available in iOS 2.0 and later.
Declared in `UIWebView.h`.

`UIWebViewNavigationTypeBackForward`
User tapped the back or forward button.
Available in iOS 2.0 and later.
Declared in `UIWebView.h`.

`UIWebViewNavigationTypeReload`
User tapped the reload button.
Available in iOS 2.0 and later.
Declared in `UIWebView.h`.

`UIWebViewNavigationTypeFormResubmitted`

User resubmitted a form.

Available in iOS 2.0 and later.

Declared in `UIWebView.h`.

`UIWebViewNavigationTypeOther`

Some other action occurred.

Available in iOS 2.0 and later.

Declared in `UIWebView.h`.

Availability

Available in iOS 2.0 and later.

Declared in

`UIWebView.h`

UIWebPaginationBreakingMode

The manner in which column- or page-breaking occurs.

```
typedef NS_ENUM(NSInteger,
   UIWebPaginationBreakingMode) {
   UIWebPaginationBreakingModePage,
   UIWebPaginationBreakingModeColumn
};
```

Constants

`UIWebPaginationBreakingModePage`

Content respects CSS properties related to page-breaking.

Available in iOS 7.0 and later.

Declared in `UIWebView.h`.

`UIWebPaginationBreakingModeColumn`

Content respects CSS properties related to column-breaking.

Available in iOS 7.0 and later.

Declared in `UIWebView.h`.

UIWebPaginationMode

The layout of content in the web view, which determines the direction that the pages flow.

```
typedef NS_ENUM(NSInteger,
   UIWebPaginationMode) {
   UIWebPaginationModeUnpaginated,
   UIWebPaginationModeLeftToRight,
   UIWebPaginationModeTopToBottom,
   UIWebPaginationModeBottomToTop,
   UIWebPaginationModeRightToLeft
};
```

Constants

`UIWebPaginationModeUnpaginated`

Content appears as one long scrolling view with no distinct pages.

Available in iOS 7.0 and later.

Declared in `UIWebView.h`.

`UIWebPaginationModeLeftToRight`

Content is broken up into pages that flow from left to right.

Available in iOS 7.0 and later.

Declared in `UIWebView.h`.

`UIWebPaginationModeTopToBottom`

Content is broken up into pages that flow from top to bottom.

Available in iOS 7.0 and later.

Declared in `UIWebView.h`.

`UIWebPaginationModeBottomToTop`

Content is broken up into pages that flow from bottom to top.

Available in iOS 7.0 and later.

Declared in `UIWebView.h`.

`UIWebPaginationModeRightToLeft`

Content is broken up into pages that flow from right to left.

Available in iOS 7.0 and later.

Declared in `UIWebView.h`.

Deprecated UIWebView Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in iOS 3.0

detectsPhoneNumbers

A Boolean value indicating whether telephone number detection is on. (Deprecated in iOS 3.0. Use `dataDetectorTypes` *(page 9) instead.)*

```
@property(nonatomic) BOOL detectsPhoneNumbers
```

Discussion
If `YES`, telephone number detection is on; otherwise, `NO`. If a webpage contains numbers that can be interpreted as phone numbers, but are not phone numbers, you can turn off telephone number detection by setting this property to `NO`. The default value is `YES` on devices that have phone capabilities.

Special Considerations
The functionality provided by this property has been superseded by the `dataDetectorTypes` (page 9) property.

Availability
Available in iOS 2.0 and later.

Deprecated in iOS 3.0.

Declared in
`UIWebView.h`

Document Revision History

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

Date	Notes
2013-09-18	Updated for iOS 7.
2013-04-23	Added information in the introduction about inspecting web content inside web views.
2012-09-19	Added new methods introduced in iOS 6.
2011-10-12	Updated for iOS 5.
2010-11-15	Added UITableView to the warning about embedding UIWebView objects in scrolls views.
2010-08-03	Adding warning about embedding web views in scroll views.
2010-06-14	Describes the allowsInlineMediaPlayback and mediaPlaybackRequiresUserAction properties introduced in iOS 4.0.
2009-06-04	Fixed broken link. Corrected description of JavaScript execution time.
2009-03-11	Updated for iOS 3.0.
2008-10-15	Added important release information to delegate.
2008-09-09	Removed text about delegate being informed of scale changes.
2008-05-30	New document that describes the class for embedding web content in an application.

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