Class

UIWebView

You can use the `UIWebView` class to embed web content in your app. To do so, 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.

Language

Swift | Objective-C

SDK

iOS 2.0+

On This Page

Overview
Symbols
Relationships

Overview

Note

In apps that run in iOS 8 and later, use the `WKWebView` class instead of using `UIWebView`. Additionally, consider setting the `WKPreferences` property `javaScriptEnabled` to `false` if you render files that are not supposed to run JavaScript.

Important

An iOS app linked on or after iOS 10.0 must include in its `Info.plist` file the usage description keys for the types of data it needs to access or it will crash. To access a user's photo data specifically, it must include NSPhotoLibraryUsageDescription and NSCameraUsageDescription.

Use the `loadHTMLString(_:baseURL:)` method to begin loading local HTML files or the `loadRequest(_:)` method to begin loading web content. Use the `stopLoading()` method to stop loading, and the `isLoading` 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()` and `goForward()` methods as actions for buttons. Use the `canGoBack` and `canGoForward` 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 app launches and dials the number. To turn off this default behavior, set the `dataDetectorTypes` property with a `UIDataDetectorTypes` bitfield that does not contain the `phoneNumber` flag.

You can also use the `scalesPageToFit` 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` 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, such as Keynote, PDF, and Pages documents. For the best rendering of plain and rich text in your app, however, you should use `UITextView` instead.

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 App Programming Guide for iOS.

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

Subclassing Notes

The `UIWebView` class should not be subclassed.

Symbols

Setting the Delegate

```
var delegate: UIWebViewDelegate?
```

The receiver's delegate.

Loading Content

```
func load(Data, mimeType: String, textEncodingName: String, baseURL: URL)
```

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

`func loadHTMLString(String, baseURL: URL?)`

Sets the main page content and base URL.

`func loadRequest(URLRequest)`

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

`var request: URLRequest?`

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

`var isLoading: Bool`

A Boolean value indicating whether the receiver is done loading content.

`func stopLoading()`

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

`func reload()`

Reloads the current page.

Moving Back and Forward

`var canGoBack: Bool`

A Boolean value indicating whether the receiver can move backward.

`var canGoForward: Bool`

A Boolean value indicating whether the receiver can move forward.

`func goBack()`

Loads the previous location in the back-forward list.

`func goForward()`

Loads the next location in the back-forward list.

Setting Web Content Properties

`var allowsLinkPreview: Bool`

A Boolean value that determines whether pressing on a link displays a preview of the destination for the link.

`var scalesPageToFit: Bool`

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

`var scrollView: UIScrollView`

The scroll view associated with the web view.

`var suppressesIncrementalRendering: Bool`

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

`var keyboardDisplayRequiresUserAction: Bool`

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

`var dataDetectorTypes: UIDataDetectorTypes`
The types of data converted to clickable URLs in the web view’s content.

Running JavaScript

`func stringByEvaluatingJavaScript(from: String)`
Returns the result of running a JavaScript script. Although this method is not deprecated, best practice is to use the `evaluateJavaScript(_:completionHandler:)` method of the `WKWebView` class instead.

Managing Media Playback

`var allowsInlineMediaPlayback: Bool`
A Boolean value that determines whether HTML5 videos play inline or use the native full-screen controller.

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

`var mediaPlaybackAllowsAirPlay: Bool`
A Boolean value that determines whether Air Play is allowed from this view.

`var allowsPictureInPictureMediaPlayback: Bool`
A Boolean value that determines whether Picture in Picture playback is allowed from this view.

Managing Pages

`var gapBetweenPages: CGFloat`
The size of the gap, in points, between pages.

`var pageCount: Int`
The number of pages produced by the layout of the web view.

`var pageLength: CGFloat`
The size of each page, in points, in the direction that the pages flow.

`var paginationBreakingMode: UIWebPaginationBreakingMode`
The manner in which column- or page-breaking occurs.

`var paginationMode: UIWebPaginationMode`
The layout of content in the web view.

Constants

`UIWebViewNavigationType`
Constant indicating the user’s action.

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

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

Relationships

Inherits From	`UIView`
Conforms To	• `CVarArg` • `Equatable` • `Hashable` • `NSCoding` • `UIAccessibilityIdentification` • `UIScrollViewDelegate`