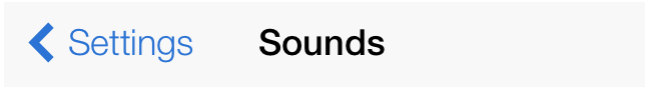
Navigation Bars

On This Page

Navigation bars allow you to present your app's content in an organized and intuitive way. A navigation bar is displayed at the top of the screen, and contains buttons for navigating through a hierarchy of screens. A navigation bar generally has a back button, a title, and a right button. The most common way to use a navigation bar is with a navigation controller. You can also use a navigation bar as a standalone object in your app.



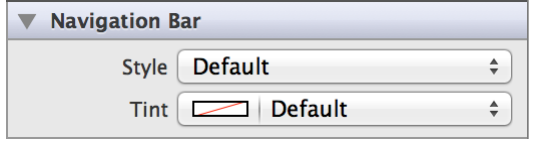
Purpose. Navigation bars allow users to:

- Navigate to the previous view
- Transition to a new view

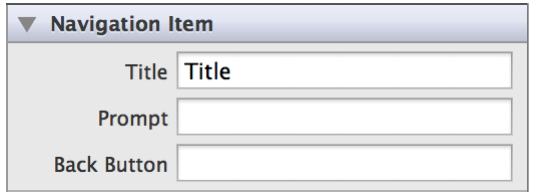
Implementation.

- Navigation bars are implemented in the `UINavigationBar` class and discussed in the *UINavigationBar Class Reference*.
- Navigation items are implemented in the `UINavigationItem` class and discussed in the *UINavigationItem Class Reference*.
- Bar button items are implemented in the `UIBarButtonItem` class and discussed in the *UIBarButtonItem Class Reference*.
- Bar items are implemented in the `UIBarItem` class and discussed in the *UIBarItem Class Reference*.

Configuration. Configure navigation bars in Interface Builder, in the Navigation Bars section of the Attributes Inspector. A few configurations cannot be made through the Attributes Inspector, so you must make them programmatically. You can set other configurations programmatically, too, if you prefer.



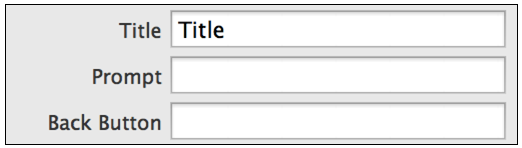
On This Page



Content of Navigation Bars

After you create a navigation bar, either in conjunction with a navigation controller or as a standalone object, you need to add content to the bar. A navigation bar can display a left button, title, prompt string, and right button.

The navigation bar displays information from a stack of `UINavigationItem` objects. At any given time, the `UINavigationItem` that is currently the `topItem` of the stack determines the title and other optional information in the navigation bar, such as the right button and prompt. The `UINavigationItem` that is immediately below the `topItem` is the `backItem`, which determines the appearance of the left or back button.



You can also add bar button items to a `UINavigationItem`. A `UIBarButtonItem` generally has a title and either a custom image or one of the system-supplied images listed in `UIBarButtonSystemItem`. It's common to have a right bar button, but you can also use a left bar button in the place of a back button.

To add any of these elements to a navigation bar, select the desired item from the Object library in Interface Builder and drag it to your storyboard. Then, you customize the contents in the Attributes Inspector as described in Images.

For more information about the elements that you add to a navigation bar, see *UINavigationItem Class Reference* and *UIBarButtonItem Class Reference*.

On This Page

Behavior of Navigation Bars

The most common way to use a navigation bar is with a `UINavigationController` object. A navigation controller manages the navigation between different screens of content for you. It also creates the navigation bar automatically, and pushes and pops navigation items as appropriate.

You can add a navigation controller to your app in Interface Builder or programmatically. To use Interface Builder to create a navigation controller, see Creating a Navigation Interface Using a Storyboard. To create a navigation controller programmatically, see Creating a Navigation Interface Programmatically.

A navigation controller automatically assigns itself as the delegate of its navigation bar object. Attempting to change the delegate raises an exception. For more information about using a navigation bar with navigation controller, see Navigation Controllers.

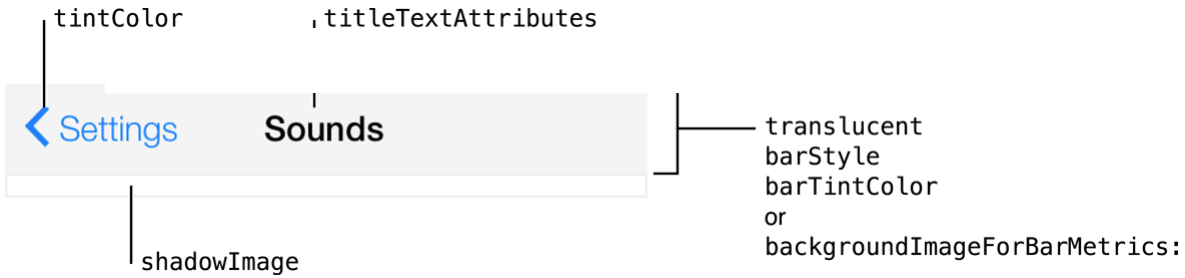
When you use a navigation bar as a standalone object, you set the initial appearance of the navigation bar by creating the appropriate `UINavigationItem` objects and adding them to the navigation bar object stack. When you create your standalone navigation bar in Interface Builder, Xcode creates the corresponding `UINavigationItem` objects for the elements you drag to the navigation bar.

You are also responsible for managing the stack of `UINavigationItem` objects when you use a navigation bar as a standalone object. You push new navigation items onto the stack using the `pushNavigationItem:animated:` method and pop items off the stack using the `popNavigationItemAnimated:` method. In addition to pushing and popping items, you can also set the contents of the stack directly using either the `items` property or the `setItems:animated:` method. You might use these methods at launch time to restore your interface to its previous state or to push or pop more than one navigation item at a time.

Assign a custom delegate object to the `delegate` property and use that object to intercept messages sent by the navigation bar. Delegate objects must conform to the `UINavigationBarDelegate` protocol. The delegate notifications let you track when navigation items are pushed or popped from the stack. You use these notifications to update the rest of your app's user interface. For more information about implementing a delegate object, see *UINavigationBarDelegate Protocol Reference*.

Appearance of Navigation Bars

You can customize the appearance of a navigation bar by setting the properties depicted below.

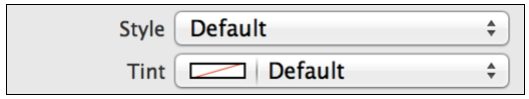


On This Page

To customize the appearance of all navigation bars in your app, use the appearance proxy (for example, `[UINavigationBar appearance]`). For more information about appearance proxies, see Appearance Proxies.

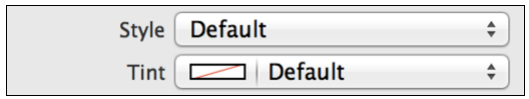
Bar Style

Navigation bars have two standard appearance styles: translucent white with dark text (default) or translucent black with light text. Use the Style (`barStyle`) field to select one of these standard styles.



Tint Color

You can specify a custom tint color for the navigation bar background using the Tint (`barTintColor`) field. The default background tint color is white.



Additionally, you can set a custom tint color for the interactive elements within a navigation bar—including button images and titles—programmatically using the `tintColor` property. The navigation bar will inherit its superview's tint color if a custom one is set, or show the default system blue color if none is set. For more information, see Tint Color.

On This Page

Images

You can set a custom background image for your navigation bar. You can do this using `setBackgroundImage:forBarMetrics:`. Note that you must specify bar metrics because navigation bars have different dimensions on different devices and orientations.

You can also use a custom shadow image for the navigation bar using the `shadowImage` property. To show a custom shadow image, you must also set a custom background image.

Translucency

Navigation bars are translucent by default in iOS 7. Additionally, there is a system blur applied to all navigation bars. This allows your content to show through underneath the bar.

These settings automatically apply when you set any style for `barStyle` or any custom color for `barTintColor`. If you prefer, you can make the navigation bar opaque by setting the `translucent` property to `NO` programmatically. In this case, the bar draws an opaque background using black if the navigation bar has `UIBarStyleBlack` style, white if the navigation bar has `UIBarStyleDefault`, or the navigation bar's `barTintColor` if a custom value is defined.

If the navigation bar has a custom background image, the default translucency is automatically inferred from the average alpha values of the image. If the average alpha is less than 1.0, the navigation bar will be translucent by default; if the average alpha is 1.0, the search bar will be opaque by default. If you set the `translucent` property to `YES` on a navigation bar with an opaque custom background image, the navigation bar makes the image translucent. If you set the `translucent` property to `NO` on a navigation bar with a translucent custom background image, the navigation bar provides an opaque background for the image using black if the navigation bar has `UIBarStyleBlack` style, white if the navigation bar has `UIBarStyleDefault`, or the navigation bar's `barTintColor` if a custom value is defined.

Title Attributes

The `titleTextAttributes` property specifies the attributes for displaying the bar's title text. You can specify the font, text color, text shadow color, and text shadow offset for the title in the text attributes dictionary, using the text attribute keys described in *NSString UIKit Additions Reference*.

You can adjust t
`setTitleVerticalPositionAdjustment:forBarMetrics:` method. Note that you must specify bar metrics because navigation bars have different dimensions for different devices and screen orientations.

On This Page

Bar Button Item Icons

Any bar button in a navigation bar can have a custom image instead of text. You can provide this image to your bar button item during initialization. Note that a bar button image will be automatically rendered as a template image within a navigation bar, unless you explicitly set its rendering mode to `UIImageRenderingModeAlwaysOriginal`. For more information, see Template Images.

Using Auto Layout with Navigation Bars

You can create Auto Layout constraints between a navigation bar and other user interface elements. You can create any type of constraint for a navigation bar besides a baseline constraint.

For general information about using Auto Layout with iOS views, see Using Auto Layout with Views.

Making Navigation Bars Accessible

Navigation bars are accessible by default. The default accessibility trait for a navigation bar is User Interaction Enabled.

With VoiceOver enabled on an iOS device, after the user navigates to a new view in the hierarchy, VoiceOver reads the navigation bar's title, followed by the name of the left bar button item. When the user taps on an element in a navigation bar, VoiceOver reads the name and the type of the element, such as, "General back button," "Keyboard heading," and "Edit button."

For general information about making iOS views accessible, see Making Views Accessible.

Internationalizing Navigation Bars

For more information, see *Internationalization and Localization Guide*.

On This Page

Debugging Navigation Bars

When debugging issues with navigation bars, watch for this common pitfall:

Specifying conflicting appearance settings. When customizing navigation bar appearance with a style or color, use one option or the other, but not both. Conflicting settings for navigation bar appearance will be resolved in favor of the most recently set value. For example, setting a new style clears any custom tint color you have set. Similarly, setting a custom tint color overrides any style you have set.

Elements Similar to a Navigation Bar

The following classes provide similar functionality to a navigation bar:

- **Toolbar.** A navigation controller can also manage a toolbar. On iPhone, this toolbar always appears at the bottom edge of the screen, but on iPad a toolbar can appear at the top of the screen. You can create a toolbar with a navigation controller, or as a standalone object. Unlike a navigation bar, which contains controls for navigating through a hierarchy of screens, a toolbar contains controls that perform actions related to the contents of the screen. For example, a toolbar might contain Share button and a Search button. For more information about toolbars, see Toolbars.
- **Tab Bar.** Similar to a navigation bar, a tab bar allows the user to switch between different views. However, a tab bar is persistent, which means that the user can select any tab from any other tab. By contrast, a navigation bar presents a linear path through various screens. For more information about tab bars, see Tab Bars.