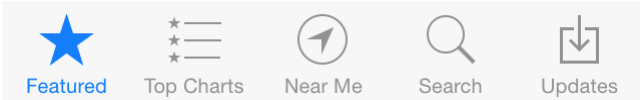
Tab Bars

On This Page

A tab bar provides easy access to different views in an app. Use a tab bar to organize information in your app by subtask. The most common way to use a tab bar is with a tab bar controller. You can also use a tab bar as a standalone object in your app.



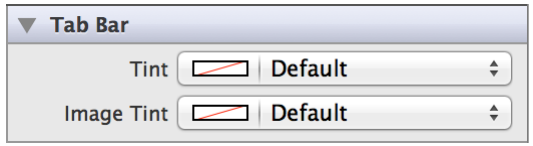
Purpose. Tab bars allow the user to:

- Quickly navigate within an app
- Get an understanding of the app's layout

Implementation.

- Tab bars are implemented in the `UITabBar` class and discussed in the *UITabBar Class Reference*.
- Tab bar items are implemented in the `UITabBarItem` class and discussed in the *UITabBarItem Class Reference*.

Configuration. Configure tab bars in Interface Builder, in the Tab Bar 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.



Content of Tab Bars

Each tab on a tab bar is represented as a `UITabBarItem` and you use the `UITabBarItem` class methods to create a tab bar

On This Page

After you create your tab bar items, add them to your tab bar with the `items` property, which is an array of `UITabBarItem` objects. If you want to animate changes to your tab bar items array, use the `setItems:animated:` method instead.

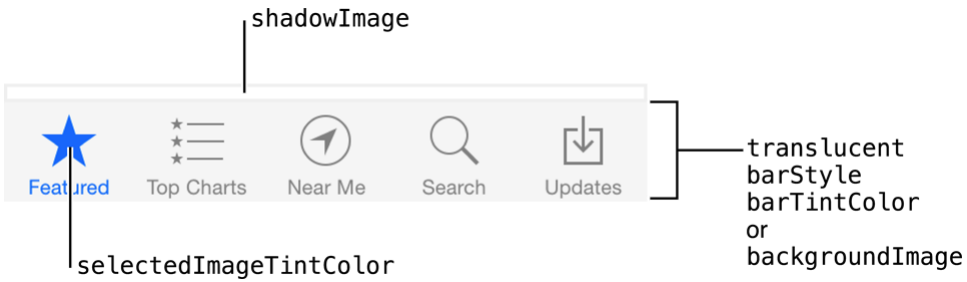
Behavior of Tab Bars (Programmatic)

You can change the contents of a tab bar at runtime, and allow users to add, remove, or reorder tabs. To present a modal view that allows users to customize a tab bar, use the `beginCustomizingItems:` method. You can also add a `UITabBarDelegate` object to your app. The tab bar delegate receives messages when the user customizes the tab bar.

The most common way to use a tab bar is in conjunction with a tab bar controller. A `UITabBarController` object manages the various tab views and view controllers, and the tab bar itself. If you use a tab bar controller, you should not use the `UITabBar` methods or properties to modify the tab bar. If you do, the system throws an exception. For more information about how to create a tab bar interface with an associated tab bar controller, see Tab Bar Controllers.

Appearance of Tab Bars

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



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

On This Page

Style

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

Tint Color

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

Use the Image Tint (`selectedImageTintColor`) field to specify the bar item's tint color when that tab is selected. By default, that color is blue.

Images

The selection indicator image is shown when a tab is selected. It is drawn on top of the tab bar, behind the bar item icon. By default, there is no selection indicator image, but you can set a custom one using the `selectionIndicatorImage` property.

By default, there is no divider image between tabs on a tab bar. You can set custom divider images for each combination of left and right tab control states using the `setDividerImage:forLeftState:rightState:` method. If you use custom dividers, make sure to set divider images all combinations of tabs states: left selected, right selected, or both unselected.

You can also set a custom background image for your entire tab bar using the `backgroundImage` property. If you set this property with a stretchable image, the image is stretched. If you use a non-stretchable image, the image is tiled.

If you want to use custom shadow image for the tab bar, set the `shadowImage` property. To show a custom shadow image, you must also set a custom background image with `backgroundImage`.

Translucency

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

On This Page

These settings a[illegible] If you prefer, you can make the tab bar opaque by setting the `translucent` property to `NO` programmatically. In this case, the bar draws an opaque background using black if the tab bar has `UIBarStyleBlack` style, white if the tab bar has `UIBarStyleDefault`, or the tab bar's `barTintColor` if a custom value is defined.

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

Tab Bar Item Icons

Each item in a tab bar can have a custom selected image and unselected image. You can specify these images when you initialize a tab bar item using the `initWithTitle:image:selectedImage:` method. Note that a tab bar item image will be automatically rendered as a template image within a tab bar, unless you explicitly set its rendering mode to `UIImageRenderingModeAlwaysOriginal`. For more information, see Template Images.

Using Auto Layout with Tab Bars

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

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

Making Tab Bars Accessible

Tab bars are accessible by default.

With VoiceOver enabled on an iOS device, when a user touches a tab in a tab bar, VoiceOver reads the title of the tab, its position in the bar, and whether it is selected. For example in the iTunes app on iPad, you might hear “Selected, Audiobooks, four of seven” or “Genius, six of seven.”

For general info

On This Page

Internationalizing Tab Bars

To internationalize a tab bar, you must provide localized strings for the tab bar item titles.

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

Elements Similar to a Tab Bar

The following classes provide similar functionality to a tab bar:

- **Segmented Control.** Similar to a tab bar, a segmented control functions as a button that shows different views. If you want to provide functionality similar to a tab bar, but don’t want those controls to be persistent, consider using a segmented control. Remember that a tab bar should be accessible from every location in an app. For more information, see Segmented Controls.
- **Navigation Bar.** A navigation bar also allows users to navigate through different content views, but it offers a linear path. With a tab bar, a user can view any other tab at any given time. For more information, see Navigation Bars.
- **Toolbar.** Both a tab bar and a toolbar are always visible onscreen. However, unlike a tab bar, which switches between views, a `UIToolbar` object contains controls that allow the user to perform actions related to objects onscreen. For more information, see Toolbars.