

# Toolbars

On This Page

A toolbar usually appears at the bottom of a screen, and displays one or more buttons called toolbar items. Generally, these buttons provide some sort of tool that is relevant to the screen's current content. A toolbar is often used in conjunction with a navigation controller, which manages both the navigation bar and the toolbar.



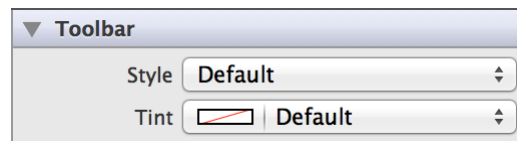
**Purpose.** Toolbars allow users to:

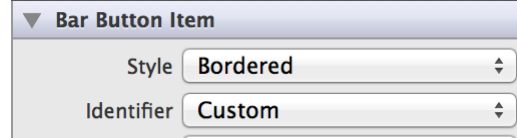
- Select one of a set of performable actions within a given view

**Implementation.**

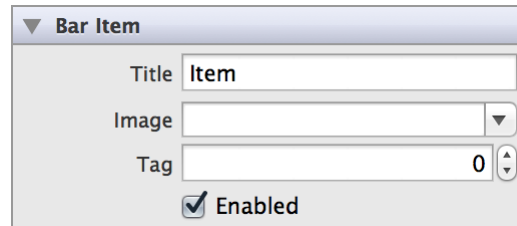
- Toolbars are implemented in the [UIToolbar](#) class and discussed in [UIToolbar Class Reference](#).
- Bar button items are implemented in the [UIBarButtonItem](#) class and discussed in [UIBarButtonItem Class Reference](#).
- Bar items are implemented in the [UIBarButtonItem](#) class and discussed in [UIBarButtonItem Class Reference](#).

**Configuration.** Configure toolbars in Interface Builder, in the Toolbar 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.





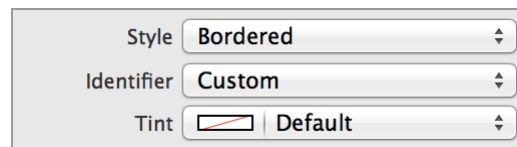
On This Page



## Content of Toolbars

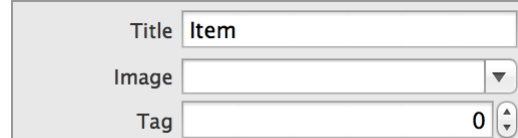
After you create a toolbar, you need to add items to the bar. Each item is a [UIBarButtonItem](#) object, which you can add to the toolbar directly in Interface Builder or in code using the [items](#) property. If you want to animate changes to your toolbar items array, use the [setItems:animated:](#) method.

You can specify the content of a particular bar button item by selecting its identifier. The identifier can either be custom or take on the value of well-know system buttons such as Edit or Done. For a list of system identifiers, see [UIBarButtonItem](#).



If you are using a bar button item with the custom identifier, you can set some of its properties at the [UIBarButtonItem](#) level. For example, you can specify either a custom title or image using the Title ([title](#)) or Image ([image](#)) fields.

You can assign a tag to your bar button item using the Tag ([tag](#)) field. This is intended to be a unique identifier for your button so you can access it in code.



On This Page

## Behavior of Toolbars

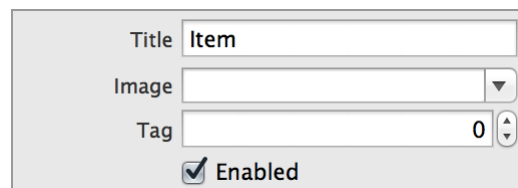
Toolbars do not need a [delegate](#) to function properly; their parent view controller can define their behavior without implementing any delegate protocols.

When a user clicks a particular button on the toolbar, you can respond by performing some corresponding action in your app, such as deleting an email. You register the [target-action](#) method for a bar button item as shown below.

```
1 self.myBarButtonItem.target = self;
2 self.myBarButtonItem.action = @selector(myAction:);
```

Alternatively, you can Control-drag the bar button item's selector from the Connections Inspector to the action method. For more information, see [Target-Action Mechanism](#).

You can disable or enable a given button on the toolbar by selecting the button in Interface Builder and toggling its Enabled ([enabled](#)) box.

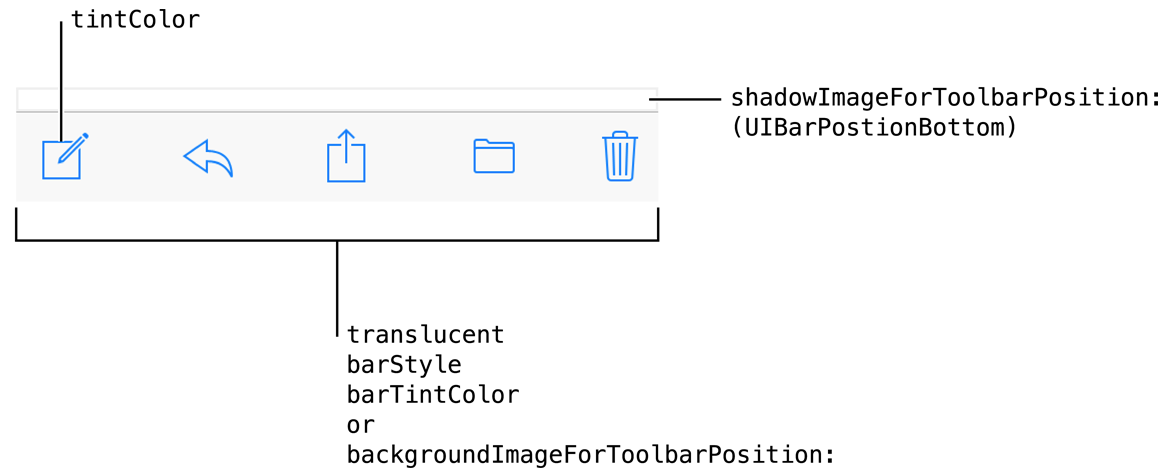


A common way to create and manage a toolbar is in conjunction with a navigation controller. The navigation controller displays the toolbar and populates it with items from the currently visible view controller. Using a navigation controller is ideal for an app design where you want to change the contents of the toolbar dynamically. However, you should not use a navigation controller if your app does not have or need a navigation bar. For more information, see [Displaying a Navigation Toolbar](#).

# Appearance of Toolbars

On This Page

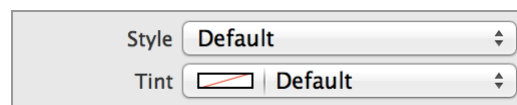
You can customize the appearance of a toolbar by setting the properties depicted below.



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

## Style

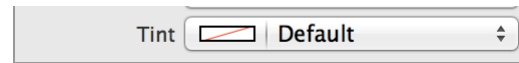
Toolbars have two standard appearance styles: translucent white with dark text (default) or translucent black with light text. Use the Style (`barStyle`) field to select one of these standard styles.



## Tint Color

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

On This Page



Additionally, you can set a custom tint color for the interactive elements within a toolbar bar—including the button images and text—programmatically using the `tintColor` property. The toolbar 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](#).

## Background Images

You can set a custom background image for your toolbar using the `backgroundImageForToolbarPosition:barMetrics:` method. The image must be the correct dimensions in order to cover the area of the toolbar correctly. Remember to set custom images for different sets of bar metrics.

If you want to use custom shadow image for the toolbar, use the `setShadowImage:forToolbarPosition:` method. To show a custom shadow image, you must also set a custom background image with `backgroundImageForToolbarPosition:barMetrics:`.

## Translucency

Toolbars are translucent by default on iOS 7. Additionally, there is a system blur applied to all toolbars. 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 toolbar opaque by setting the `translucent` property to `NO` programmatically. In this case, the bar draws an opaque background using black if the toolbar has `UIBarStyleBlack` style, white if the toolbar has `UIBarStyleDefault`, or the toolbar’s `barTintColor` if a custom value is defined.

If the toolbar 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 toolbar will be translucent by

default; if the average alpha is 1.0, the toolbar will be opaque by default. If you set the [translucent](#) property to YES on a toolbar with an opaque custom background image, the toolbar makes the image translucent. If you set the [translucent](#) property to NO on a toolbar with a translucent custom background image, the toolbar provides an opaque background image. If the toolbar has a custom background image, the toolbar has [backgroundImage](#).

On This Page

## Bar Button Item Icons

Any bar button in a toolbar 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 toolbar, unless you explicitly set its rendering mode to `UIImageRenderingModeAlwaysOriginal`. For more information, see [Template Images](#).

## Using Auto Layout with Toolbars

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

You cannot create Auto Layout constraints for individual bar button items. However, you can use bar button items with the Fixed Space and Flexible Space identifiers to determine the spacing of buttons on your toolbar.

For general information about using Auto Layout with iOS views, see [Using Auto Layout with Views](#).

## Making Toolbars Accessible

Toolbars are accessible by default.

For general information about making iOS views accessible, see [Making Views Accessible](#).

## Internationalizing Toolbars

To internationalize a toolbar, you must provide localized strings for all button titles. Remember to test all localizations, as button size may change unexpectedly when using localized strings.

For more information, see [Internationalization and Localization Guide](#).

On This Page

## Debugging Toolbars

When debugging issues with toolbars, watch for this common pitfall:

**Trying to customize the content of a non-custom bar button item.** If you try to set a custom title or image—at the `UIBarButtonItem` level—for a bar button item with a non-custom identifier, the bar button item’s identifier will automatically switch to the custom type in Interface Builder.

## Elements Similar to a Toolbar

The following elements provide similar functionality to a toolbar:

- **Tab Bar.** A toolbar is most similar to a tab bar—both can appear at the bottom of the screen. Use a toolbar to display controls that perform specific functions, and use a tab bar to allow the user to switch between different views or subtasks. For more information, see [Tab Bars](#).
- **Navigation Bar.** For more information, see [Navigation Bars](#).