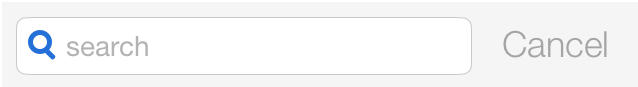
Search Bars

On This Page

A search bar provides an interface for text-based searches with a text box and buttons such as search and cancel. A search bar accepts text from users, which can be used as input for a search (shown here with placeholder text). A scope bar, which is available only in conjunction with a search bar—allows users to define the scope of a search (shown here below a search bar).

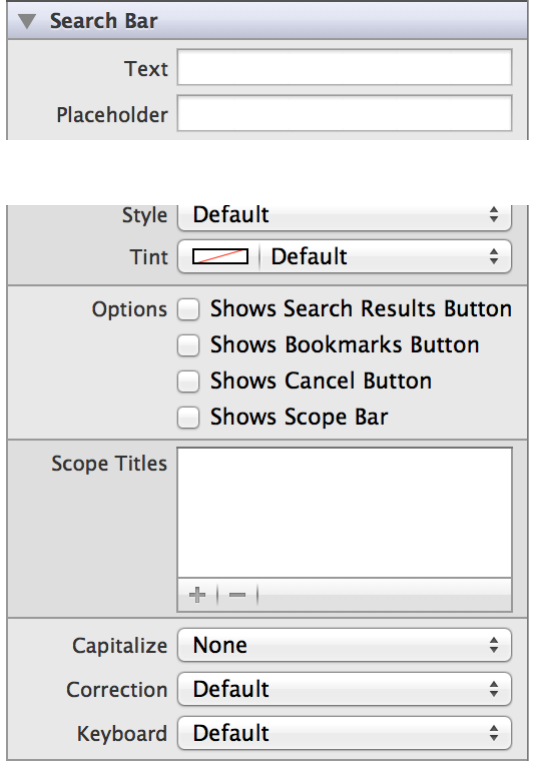


Purpose. Search bars allow users to:

- Quickly find a value in a large collection
- Create a scope filter

Implementation. Search bars are implemented in the `UISearchBar` class and discussed in the *UISearchBar Class Reference*.

Configuration. Configure search bars in Interface Builder, in the Search 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.



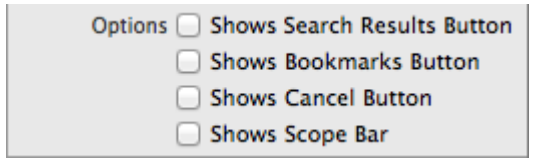
On This Page

Content of Search Bars

The Text (`text`) field contains the current search text; you can use it to set an initial search. Don't use it to provide a description of the search; use placeholder text instead. Placeholder text is specified in the Placeholder (`placeholder`) field, and is visible only when there is no other text in the search field. Placeholder text is styled differently to communicate its different meaning to the user and it is automatically cleared when the user starts typing. It is suitable for very short descriptions of what the user should enter in the search field.

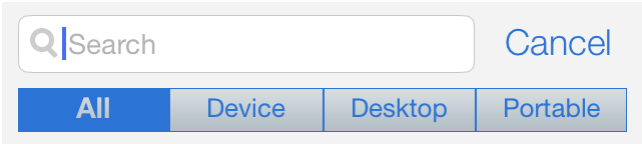
The prompt text is specified in the Prompt (`prompt`) field. It appears directly above the search bar. Unlike the placeholder text, the prompt text is visible whether or not the user has entered text in the search field, so it is suitable for longer descriptions or directions.

On This Page



Search bars can display a number of different buttons. The Cancel button is intended to terminate a search operation; you can display this button by selecting the Shows Cancel Button checkbox. The Search Results and Bookmarks buttons appear in the right side of search bar, and can be toggled to display those respective views. You can display one of these buttons by selecting either the Shows Search Results Button (`showsSearchResultsButton`) or Shows Bookmarks Button (`showsBookmarkButton`) checkbox. Note that you cannot display both of these buttons simultaneously; if both properties are enabled, only the Search Results button is visible.

> NOTE
>
> These buttons are merely user interface elements and have no functionality. You must implement the appropriate functionality yourself using the corresponding `UISearchBarDelegate` methods.



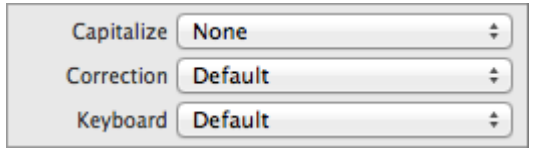
A search bar can also display a scope bar, which lets users limit the scope of a search. For example, when searching in an email app, the user could restrict the search to the Inbox or to a particular folder. To display a scope bar, check the Shows Scope Bar (`showsScopeBar`) box and add an array of scope bar titles as strings to the Scope Titles (`scopeButtonTitles`) field.

On This Page

Behavior of Search Bars

Search bars need a delegate to handle user interaction. You implement the `UISearchBarDelegate` protocol on a delegate object to respond to user actions—for example, performing the search. Every search bar needs a delegate object that implements the `UISearchBarDelegate` protocol. The delegate is responsible for taking actions in response to user input such as editing the search text, starting or canceling a search, and tapping in the scope bar. At the very minimum, the delegate needs to perform a search after text is entered in the text field.

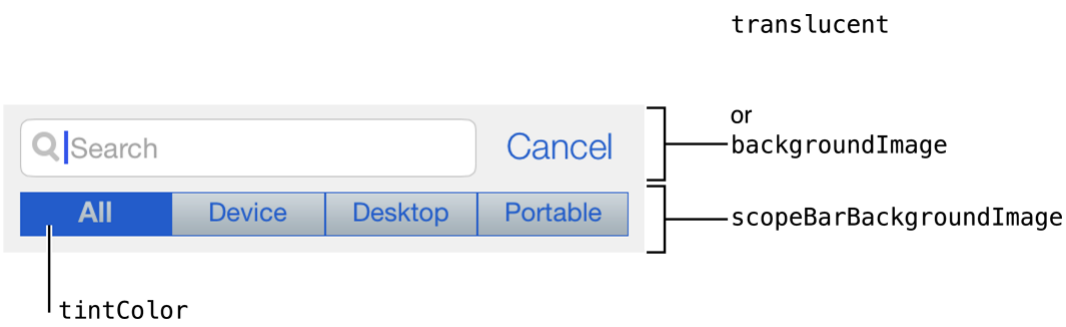
A user types content into a search bar using a keyboard, which has a number of customization options:



- **Keyboard layout.** The Keyboard field allows you to select from a number of different keyboard layouts. Match the keyboard layout to the purpose and scope of the search bar. The default keyboard layout is an alphanumeric keyboard in the device's default language. For a list of possible keyboard types, see `UIKeyboardType`.
- **Capitalization scheme.** The Capitalization field specifies how text should be capitalized in the search bar: no capitalization, every word, every sentence, or every character. The no capitalization scheme is selected by default.
- **Auto-correction.** The Correction field simply disables or enables auto-correct in the search bar.

Appearance of Search Bars

You can customize the appearance of a search bar by setting the following properties:

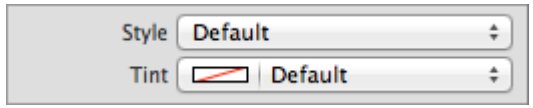


On This Page

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

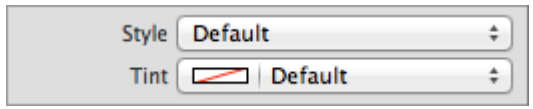
Style

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 search 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 search bar—including the scope bar, cancel button, and cursor—programmatically using the `tintColor` property. The search 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

Background Images

A search bar can have a background image that covers the area behind the search field. Use the `backgroundImage` property to set a background image for your search bar. You can also set the background image for a search bar's scope bar using the `scopeBarBackgroundImage` property. Single-pixel images or stretchable images are stretched; otherwise, the image is tiled. If you set one of these background image properties, you should also set the other to give your app interface a consistent look.

Translucency

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

If the search 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 search 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 search bar with an opaque custom background image, the search bar makes the image translucent. If you set the `translucent` property to `NO` on a search bar with a translucent custom background image, the search bar provides an opaque background for the image using black if the search bar has `UIBarStyleBlack` style, white if the search bar has `UIBarStyleDefault`, or the search bar's `barTintColor` if a custom value is defined.

Layout

You can also control certain aspects of the search bar's layout by providing position adjustments: for icons using the `positionAdjustmentForSearchBarIcon:` method, for the background image using the `searchFieldBackgroundPositionAdjustment` property, and for search text using the `searchTextPosi`

On This Page

Using Auto Layout with Search Bars

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

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

Making Search Bars Accessible

Search bars are accessible by default.

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

Internationalizing Search Bars

To internationalize a search bar, you must provide localized strings for the following properties:

- `placeholder`
- `prompt`
- `text`
- `scopeButtonTitles`

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

Debugging Navigation Bars

When debugging issues with navigation bars, watch for these common pitfalls:

- **Specifying conflicting appearance settings.** When customizing search bar appearance with a style or color, use one option or the other, but not both. Conflicting settings for search bar appearance will be resolved i
color you have set. Similarly, setting a custom tint color overrides any style you have set.
- **Performance issues.** If search operations can be carried out very rapidly, it is possible to update the search results as the user is typing by implementing the `searchBar:textDidChange:` method on the delegate object. However, if a search operation takes more time, you should wait until the user taps the Search button before beginning the search in the `searchBarSearchButtonClicked:` method. Always perform search operations a background thread to avoid blocking the main thread. This keeps your app responsive to the user while the search is running and provides a better user experience.

On This Page

Elements Similar to a Search Bar

The following element provides similar functionality to a search bar:

Toolbar. A toolbar object contains controls that allow the user to perform actions related to objects onscreen. For more information, see Toolbars.